МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського”

«Інформатика та обчислювальна техніка» – ІОТ-2017

Матеріали наукової конференції студентів, магістрантів та аспірантів

25 – 27 квітня 2017 року

(кафедра «Автоматизовані системи обробки інформації і управління»)

Київ 2017

ЗМІСТ

УДК 004.942:519.876.2

ЛЕЩЕНКО К. С.,
СТЕЦЕНКО І. В.

МЕТОДИ ВІЗУАЛЬНОГО ПРОГРАМУВАННЯ СТОХАСТИЧНИХ МЕРЕЖ ПЕТРІ

Дану роботу присвячено аналізу методів візуального програмування стохастичних мереж Петрі, а також розробці компонента візуального програмування стохастичних мереж Петрі програмного забезпечення Петрі-об'єктного моделювання дискретно-подійних систем. В рамках роботи реалізовано механізм збереження імітаційних моделей, побудованих за допомогою маніпулювання графічними об'єктами, у вигляді програмного коду мовою Java, а також відновлення візуальних моделей з програмного коду з автоматичним розташуванням елементів мережі Петрі так, щоб максимізувати легкість сприйняття моделей користувачем і швидкість їх побудови.

This work is dedicated to the analysis of the stochastic Petri nets visual programming methods and to the development of a stochastic Petri nets visual programming component of the Petri-object modeling software for discrete-event systems. A mechanism has been created to save graphically created models in the form of program code in Java and restore visual models from the program code with automatic placement of Petri nets' elements in a way that maximizes the ease of model perception and the speed of model construction.

1. Вступ

З розвитком інформаційних технологій зростає потреба в імітаційних моделях складних систем, що використовуються для відшукання оптимальних параметрів управління та прийняття рішень. Підвищення складності систем, поведінку яких необхідно моделювати, пов'язане з тенденцією до зростання кількості їх структурних елементів, а також кількості взаємозв'язків між ними. В зв'язку з цим відбувається зростання вимог до засобів моделювання, зокрема постає задача забезпечення високої швидкості побудови імітаційних моделей дискретно-подійних систем, а також представлення моделей у вигляді, що максимально полегшив би їхнє сприйняття користувачем. Крім того, набуває важливості швидкість модифікації моделей в зв'язку зі структурними та функціональними змінами. Все це призводить до актуальності розробки такого програмного забезпечення, яке дозволило б імітувати поведінку складних дискретно-подійних систем, швидко будувати та модифікувати їх моделі за допомогою візуального програмування та забезпечувало зручність представлення імітаційних моделей користувачу.

У багатьох наукових роботах відзначається ефективність використання формалізму мереж Петрі для опису та дослідження паралельних процесів та процесів управління. Застосування мереж Петрі для моделювання комунікаційних систем розглядається в роботі [1]. Приклади моделювання та аналізу продуктивності складних стохастичних систем мережею Петрі наведені в роботі [2]. Використання мережі Петрі для моделювання бізнес-процесів описано в роботі [3].

Ми пропонуємо вирішувати задачу забезпечення високої швидкості побудови імітаційних моделей та зручного їх представлення за допомогою технології Петрі-об'єктного

моделювання. Вона ґрунтується на концепції, що модель системи конструюється з елементів, динаміка яких задана стохастичною мережею Петрі. Об’єкти із заданою динамікою тиражуються у заданій кількості і з заданими параметрами, можуть успадковувати динаміку від інших об’єктів. Доведено, що динаміка Петрі-об’єктної моделі в цілому теж описується стохастичною мережею. Цей факт є важливим, оскільки гарантує коректність та обчислюваність моделі [4]. Ефективність та поліноміальна обчислювальна складність алгоритму імітації Петрі-об’єктної моделі доведені в роботі [5].

Одним із важливих питань, що постають при розробці програмного забезпечення імітаційного моделювання систем, є питання про формат зберігання розроблених за його допомогою імітаційних моделей. Побудова моделей засобами візуального програмування означає автоматичну генерацію програмного коду, що відповідає за створення моделі, на основі маніпуляцій користувача з графічними об’єктами, які представляють структурні елементи імітаційної моделі. Візуальне представлення моделі системи, реалізація можливості створювати моделі за допомогою зручного графічного інтерфейсу прискорюють процес конструювання складних моделей, зменшують кількість помилок при введенні моделей, значно спрощують процес їх корегування та модифікації. Проте збереження графічних образів моделей для складних систем може призвести до значних витрат ресурсів комп’ютерної пам’яті. В той же час зберігання програмного коду є в десятки разів менш витратним. Отже, застосовуючи візуальне програмування імітаційних моделей, можна вирішити проблему досягнення компромісу між зручністю використання моделі та витратами ресурсів на її зберігання.

2. Огляд існуючих методів і засобів візуального програмування

Візуальне програмування пройшло довгий шлях від перших спроб у 1970-х роках [6]. Воно надає можливість генерувати програмний код за рахунок використання графічних та символьних елементів, якими можна інтерактивно маніпулювати відповідно до певних правил. Існує багато мов візуального програмування, в яких процес розробки базується на ідеї "фігур і ліній": суб'єкти представляються у вигляді фігур (наприклад, прямокутників), що поєднуються між собою лініями (дугами, стрілками тощо), яким відповідають відношення між суб’єктами.

Однією з перших галузей, в яких мови візуального програмування зазнали успіху, є навчання, оскільки вони здатні допомагати користувачам вчитись програмуванню [6]. Велику кількість мов візуального програмування розроблено з метою зменшити кількість труднощів, з якими стикаються новачки, починаючи програмувати. Прикладом такої мови може слугувати Scratch. Граматика Scratch та похідних від неї мов – це по суті граматика класичної імперативної мови програмування, однак графічні підказки дозволяють зробити очевидними правила комбінування базових елементів, таких як змінні, умовні оператори та оператори циклів. Всередині блоків для їхнього опису використовується текст, а для призначення імен та значень змінних використовуються текстові поля. У порівнянні з іншими мовами візуального програмування даний формат дуже близький до вихідного коду імперативної мови, але використовує візуальні блоки, допомагаючи користувачу відкривати для себе синтаксис і запобігаючи помилкам [6].

Деякі мови візуального програмування для опису потоку керування використовують візуальне представлення, що нагадує представлення блок-схем. Незважаючи

на те, що даний формат є легким для розуміння, такі мови мають суттєвий недолік: логічні конструкції, що можуть бути створені безпосередньо за допомогою мови візуального програмування, обмежені; багато залежить від того, що знаходиться всередині блоків, а це часто неможливо змінити за допомогою графічного інтерфейсу [6].

Серед відомих підходів до візуального програмування слід відзначити також програмування потоків даних, скінченні автомати, правила, засновані на подіях, та дерева поведінки.

Незважаючи на те, що мови візуального програмування досі не вийшли на рівень широкого професійного застосування, існують сфери, в яких вони достатньо розвинені та являють собою ефективні засоби, адаптовані до конкретних цілей. Це пов'язано з перевагами мов візуального програмування в порівнянні з класичним програмуванням, що в деяких сферах грають вирішальну роль. Основні з цих сфер - це навчання, мультимедіа, автоматизація, робототехніка, імітаційне моделювання.

3. Проектування компоненту візуального програмування стохастичних мереж Петрі

На базі існуючого програмного засобу Петрі-об’єктного моделювання дискретно-подійних систем [7] ми розробили компонент візуального програмування стохастичних мереж Петрі з часовими затримками, багатоканальними переходами та інформаційними зв’язками. Він має такі функціональні можливості:

- зберігання мережі Петрі, побудованої за допомогою маніпулювання графічними об'єктами (позиціями, переходами, вхідними та вихідними дугами), у вигляді програмного коду мовою Java, а саме у вигляді статичного методу спеціально призначеного для цього класу;
- відновлення візуальної моделі дискретно-подійної системи з програмного коду методу;
- автоматичне розташування елементів мережі, що забезпечує легкість сприйняття моделі користувачем (мінімізує кількість перетинів між дугами та випадків, коли будь-які елементи мережі частково перекривають один одного).

При збереженні візуальної імітаційної моделі у програмний метод послідовно аналізуються властивості всіх позицій, переходів і дуг, після чого генеруються та зберігаються відповідні вирази мовою Java.

Відновлення візуальної моделі з методу відбувається в два етапи. На першому етапі в результаті аналізу коду формується об'єкт класу PetriNet. На другому - на його основі будується об'єкт класу GraphPetriNet, що містить інформацію про розміщення елементів мережі у графічному просторі.

Алгоритм, розроблений для другого етапу візуалізації мережі Петрі, полягає в наступному (тут під вертикальним набором розуміється абстракція, що являє собою набір елементів, тобто позицій або переходів, які буде розміщено одним стовпцем, тобто один під одним; кожен такий набір може містити або лише позиції, або лише переходи):

1. Побудувати список вертикальних наборів позицій та переходів:

1.1. Вилучити будь-який перехід зі списку наявних переходів та помістити його в перший вертикальний набір (тип набору: для переходів).

1.2. Доки є елементи в списку наявних позицій/переходів, повторювати:

1.2.1. Обнулити посилання на останній вертикальний набір та його індекс у списку вертикальних наборів.

1.2.2. Знайти будь-який вертикальний набір, який не позначено "готовим", та вважати його останнім (встановити відповідні значення посилання та індексу).

1.2.3. Якщо всі вертикальні набори "готові", припинити роботу.

1.2.4. Побудувати множини вхідних і вихідних переходів/позицій (відносно елементів останнього набору), які не перетинаються.

1.2.5. За необхідності додати новий порожній набір до початку або кінця списку наборів.

1.2.6. Для всіх вхідних, а потім вихідних елементів (переходів/позицій):

1.2.6.1. Додати цей елемент до вертикального набору, що знаходиться: перед останнім - якщо елемент вхідний, після останнього - якщо вихідний.

1.2.6.2. Позначити набір, до якого додано елемент, як “не готовий”.

1.2.6.3. Вилучити елемент зі списку наявних переходів/позицій.

1.2.7. Позначити останній вертикальний набір як “готовий”.

2. На основі побудованого списку згенерувати об’єкт класу GraphPetriNet:

2.1. Обнулити координати x та y.

2.2. Для кожного вертикального набору (починаючи з першого) виконати:

2.2.1. Збільшити x на фіксовану кількість пікселів та присвоїти значення y (залежить від кількості елементів у вертикальному наборі).

2.2.2. Для кожного елементу вертикального набору виконати:

2.2.2.1. Збільшити y на фіксовану кількість пікселів.

2.2.2.2. Створити на основі даного елементу об'єкт класу GraphPetriPlace / GraphPetriTransition (графічний аналог позиції або переходу відповідно) з центром у точці (x, y).

2.2.2.3. Додати даний об’єкт до списку графічних аналогів позицій/переходів.

2.3. З’єднати графічні аналоги позицій та переходів дугами.

2.4. Створити об’єкт класу GraphPetriNet, передавши у конструктор побудовані списки графічних аналогів позицій, переходів і дуг.

2.5. Перемістити даний об’єкт до центру видимого візуального простору.

4. Результати роботи розробленого компоненту

Результат генерування методу на основі візуальної моделі, побудованої за допомогою графічного інтерфейсу, показано на рисунку 1.

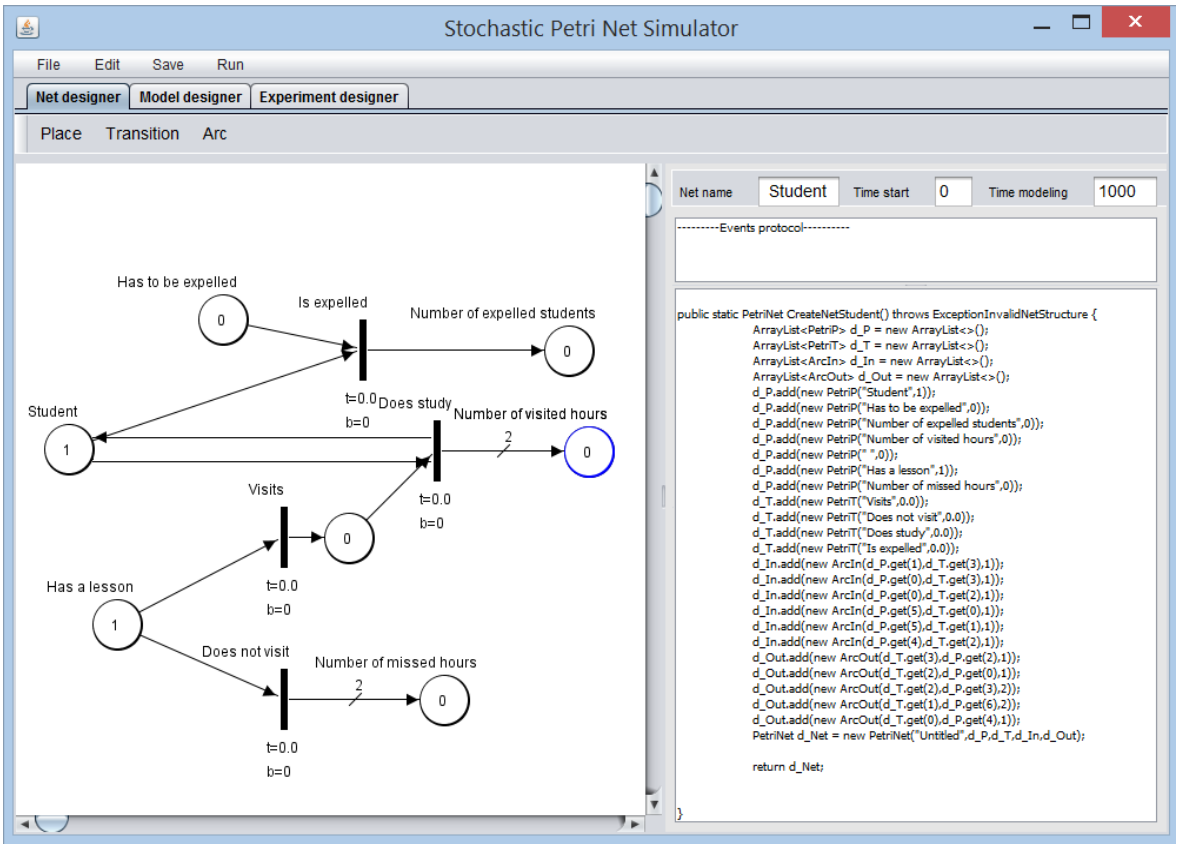


Рис. 1. Метод, згенерований на основі візуальної моделі

Результат відновлення візуальної моделі з методу демонструє рисунок 2.

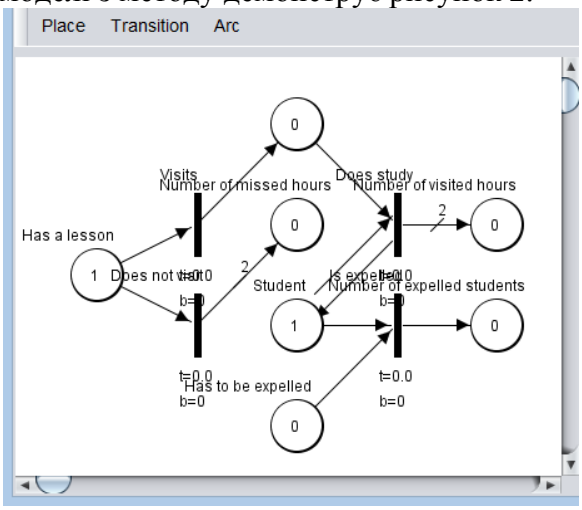


Рис. 2. Візуальна модель системи, згенерована на основі Java методу

5. Висновки

В результаті наукового дослідження розроблено інтелектуальний компонент візуального програмування стохастичних мереж Петрі, призначений для збереження моделей, побудованих за допомогою маніпулювання графічними об'єктами, у вигляді програмного коду мовою Java, а також відновлення візуальних моделей з програмного коду. Розроблений компонент підвищує легкість сприйняття моделі користувачем за рахунок автоматичного розташування елементів мережі Петрі, що відновлюється з програмного коду, таким чином, щоб мінімізувати кількість перетинів між дугами та випадків, коли будь-які елементи мережі частково перекривають один одного. Крім того, компонент забезпечує вищу швидкість побудови та модифікації імітаційних моделей дискретно-подійних систем.

Проаналізувавши низку методів і засобів візуального програмування та порівнявши збереження імітаційних моделей у вигляді програмного коду зі збереженням їхніх графічних образів, ми дійшли висновку, що, застосовуючи візуальне програмування імітаційних моделей, можна вирішити проблему досягнення компромісу між зручністю використання моделі та витратами ресурсів на її зберігання.

Список літератури

1. Zaitsev, D. A. Clans of Petri Nets: Verification of protocols and performance evaluation of networks / D. A. Zaitsev. – LAP LAMBERT Academic Publishing, 2013. – 292 p.

2. Haas, P. J. Stochastic Petri Nets: Modelling, Stability, Simulation / P. J. Haas. – Springer Science & Business Media, 2006. – 510 p.

3. Aalst, W. Modeling Business Process – A Petri Net-Oriented Approach / W. Aalst, C.Stahl. – The MITPress, 2011. – 400 p.

4. Стеценко И. В. Теоретические основы Петри-объектного моделирования систем / И. В. Стеценко // Математичні машини і системи. – 2011. – № 4. – С. 136-148.

5. Stetsenko, I. V. Petri-Object Simulation: Software Package and Complexity / I. Stetsenko, V. Dorosh, A. Dyfuchyn // Proceedings of the 8th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS'2015). – Warsaw (Poland), 2015. – P. 381-385.

6. Dehouck R. Craft AI. The maturity of visual programming [Електронний ресурс] / Rémi Dehouck. – 2015. – Режим доступу до ресурсу: http://www.craft.ai/blog/the-maturity-of-visual-programming/.

7. Стеценко І. В. Проектування графічного модуля програмного забезпечення Петрі-об'єктного моделювання систем / І. В. Стеценко, О. В. Василевська // Вісник ЧДТУ. – 2013. – № 2. – С. 13-18.

УДК 004.021
ГУЛЯНИЦЬКИЙ Л.Ф.,
ШУТЬ В.А.

ВИКОРИСТАННЯ ПЕРСОНАЛЬНИХ ХАРАКТЕРИСТИК В РЕКОМЕНДАЦІЙНИХ СИСТЕМАХ ДЛЯ НОВИХ КОРИСТУВАЧІВ СИСТЕМИ

Рекомендаційні системи допомагають користувачам легко і швидко знаходити в мережі Інтернет інформацію або продукти, що будуть для них найбільш цікавими або корисними. Методи колаборативної фільтрації, що використовуються в рекомендаційних системах, створюють рекомендації для користувачів на основі вподобань найбільш подібних до них інших користувачів. Такі методи в наш час широко використовуються в різноманітних системах. Незважаючи на загальний успіх рекомендаційних систем з використанням колаборативної фільтрації, є ряд загальних проблем, з якими стикаються методи колаборативної фільтрації. Однією з таких проблем є проблема холодного старту. У цій статті пропонується метод, який базується на поєднанні особистісних характеристик користувача і поставлених користувачем оцінок при обчисленні подібності користувачів. Цей підхід створено для рекомендаційних систем, щоб вони могли давати ефективні рекомендації новим користувачам, які взагалі ще нічого не оцінили або оцінили малу кількість продуктів. В статті буде проведено порівняльний аналіз запропонованого гібридного методу, традиційного методу заснованого на оцінках користувача і методу заснованого на даних користувача. Результати проведених досліджень показують, що новий метод добре вирішує проблему холодного старту.
Ключові слова: РЕКОМЕНДАЦІЙНА СИСТЕМА, КОЛАБОРАТИВНА ФІЛЬТРАЦІЯ, ПЕРСОНАЛЬНІ ХАРАКТЕРИСТИКИ КОРИСТУВАЧА, ПОДІБНІСТЬ КОРИСТУВАЧІВ, НОВИЙ КОРИСТУВАЧ

Recommendation systems help users quickly and easily find Internet information or products that are most interesting or helpful to them. The methods of collaborative filtering provide recommendations to users based on preferences they like most others. Such methods are nowadays widely used in various systems. Despite the overall success of recommendation systems using collaborative filtering, there are some common problems faced by collaborative filtering method. One of these problems is called “cold start”. In this paper we propose a method that is based on a combination of personal characteristics of the user and set the user estimates when calculating the similarity users. This approach created to recommender system could provide effective recommendations to new users who do nothing yet rated or rated small number of products. There is a comparative analysis of the new proposed method, the traditional method based on estimates of the user and method based on personal user data in this paper. The studies show that the new method successfully solves the problem called “cold start”.
Keywords: RECOMMENDER SYSTEM, COLLABORATIVE FILTERING PERSONAL DATA USER SIMILARITY USERS NEW USER

1. Вступ

Рекомендаційні системи допомагають користувачам легко і швидко знаходити в мережі Інтернет інформацію або продукти, що будуть для них найбільш цікавими або корисними. Методи колаборативної фільтрації, що використовуються в рекомендаційних системах, створюють рекомендації для користувачів на основі вподобань найбільш подібних до них інших користувачів. Такі методи в наш час широко використовуються в різноманітних системах. Незважаючи на загальний успіх рекомендаційних систем з використанням колаборативної фільтрації, є ряд загальних проблем, з якими стикаються методи колаборативної фільтрації. Однією з таких проблем є проблема холодного старту. У цій статті пропонується метод, який базується на поєднанні особистісних характеристик користувача і поставлених користувачем оцінок при обчисленні подібності користувачів. Цей підхід створено для рекомендаційних систем,

щоб вони могли давати ефективні рекомендації новим користувачам, які взагалі ще нічого не оцінили або оцінили малу кількість продуктів. В статті буде проведено порівняльний аналіз запропонованого гібридного методу, традиційного методу заснованого на оцінках користувача і методу заснованого на даних користувача. Результати проведених досліджень показують, що новий метод добре вирішує проблему холодного старту.

2. Колаборатина фільтрація на основі даних користувача

Колаборативна фільтрація це підхід, який широко використовується для побудови рекомендаційних систем. *Фільтрація* - фільтрація інформації, вибір потрібної (інформації, щонайбільш впливає на точність рекомендацій) інформації з великого набору даних. *Колаборативна* - інформація, що використовується для створення рекомендацій, отримана від всіх користувачів, що користуються системою. Активному користувачеві подобаються продукти А та Б, цей підхід потім порівнює вподобання активного користувача з іншими користувачами і за отриманими результатами знаходить подібних користувачів (рисунок 2). В цьому випадку це другий користувач, бо йому, як і першому користувачеві, також подобаються продукти А та Б. Другому користувачеві також подобається продукт Г, тому є вірогідно, що продукт Г також сподобається і першому користувачу.

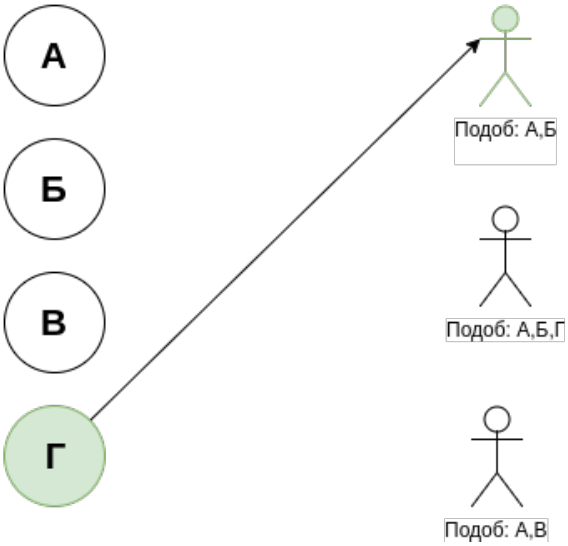


Рис. 2. Колаборативна фільтрація

3. Критерій схожості

Найбільш важливим кроком в рекомендаційних системах колаборативної фільтрації є обчислення подібності між користувачами, яка використовується для формування сусідства для активного користувача та ряду користувачів зі схожою моделлю поведінки. Процес знаходження сусідства це фактично процес навчання для алгоритму рекомендаційних систем. Існує велика кількість різноманітних підходів для обчислення подібності *simr(u,v)* між користувачами *u* та *v* [1, 2, 3]. Літера *r* означає, що схожість обчислюється на основі рейтингових даних, що відрізняється від подібності, що базується на персональних, яка буде запропонована нижче. Найбільш часто використовуваний метод обчислення подібності це коефіцієнт кореляції Пірсона [3, 4]. Якщо більш конкретно, то подібність між користувачами визначається так:

$$simr(u,v) = \frac{\sum_{i \in I_u \cap I_v} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I_u \cap I_v} (r_{u,i} - \bar{r}_u)^2 \sum_{i \in I_u \cap I_v} (r_{v,i} - \bar{r}_v)^2}},$$

де $r_{u,i}$ це рейтинг, який користувача системи ***u*** для продукту системи ***i***;

$\bar{r}_u$ це середнє значення рейтингу користувача системи *u*;

I_u це множина продуктів системи, які оцінив користувач системи *u*.

4. Прогнозування рейтингових оцінок користувача

Для того, щоб спрогнозувати значення невідомого рейтингу $\tilde{r}_{u,i}$ системи рекомендацій колаборативної фільтрації для користувача *u* та продукту *i*, необхідно зробити агрегацію рейтингів для продукту системи ***i*** від деяких інших (як правило, *k* найбільш подібних) користувачів. Формально передбачення невідомого рейтингу $\tilde{r}_{u,i}$ може бути обчислене наступним чином:

$$\tilde{r}_{u,i} = aggr_{v \in \Omega_u} r_{v,i}$$

де Ω_u - це множина *k* користувачів, що найбільш подібні до користувача *u* та які поставили оцінку продукту системи *i*.

Різні стратегії агрегування розроблені і використовуються в залежності від різних областей застосування, таких як, усереднення рейтингу, або використання подібностей в якості ваг під час агрегування. Загальна функція агрегації має наступний вигляд:

$$\tilde{r}_{u,i} = \bar{r}_u + k \sum_{v \in \Omega_u} sim(u,v) * \left(r_{v,i} - \bar{r}_v\right),$$

де *k* це нормуючий коефіцієнт;

$\bar{r}_u$ - середній рейтинг користувача *u*.

Така функція агрегації враховує той факт, що різні користувачі системи можуть користуватися різними шкалами рейтингу при оцінці продуктів в системі. Наприклад, користувач *u* вважає, що рейтинг 3 з 5 означає що цей продукт системи нормальний, тим часом як користувач *v* може вважати, що цей же рейтинг 3 має негативне забарвлення. Тому зважена сума використовує відхилення від середнього значення.

Якщо розглядати рекомендаційну систему, в якій в якості оцінки використовується факт покупки продукту, то вище згадана формула набуде наступного вигляду:

$$\tilde{r}_{u,i} = \sum_{v \in \Omega_u} sim(u,v) * r_{v,i}.$$

5. Подібність користувачів на основі персональних характеристик

Як описано вище, традиційні системи колаборативної фільтрації знаходять «сусідів» користувача на основі оцінок, які користувачі ставлять продуктам в системі. Проте, на практиці, матриця продукт-користувач *R* завжди розріджена. Тобто, кількість вже проставлених користувачами оцінок, як правило, істотно мала в порівнянні з кількістю оцінок, які потенційно ще можуть бути проставленими користувачами системи.

Вкрай важливою є можливість ефективного передбачення рейтингів маючи невелику кількість вже отриманих рейтингів. Якщо конкретний користувач оцінив деякі продукти, які були також оціненими далеко не всіма користувачами системи, то буде достатньо важко знайти достатню кількість сусідів та зробити ефективні рекомендації для конкретного користувача системи. Ця ситуація найчастіше трапляється, коли новий користувач додається в систему і при цьому не має поставлених оцінок для продуктів в системі або має дуже невелику кількість проставлених рейтингів для продуктів в системі. Для вирішення цієї проблеми ми пропонуємо новий метод обчислення міри подібності між користувачами системи, яка заснована на персональних характеристиках користувачів системи та враховує вподобання користувача, якщо вони є.

Будемо розглядати персональні характеристики користувачів як вектор. Для користувача *u* цей вектор буде мати такий вигляд:

$$p_u = (p_u^1, p_u^2, \dots, p_u^n)^T,$$

де p_u - вектор розмірності *n*.

Кожний вимір позначає одну персональну характеристику користувача в системі. Наприклад, якщо персональні характеристики користувача вимірюються так як в моделі Big Five Factor [6], яка описує користувача через основних п'ять психологічних рис, тоді p_u являє собою вектор з розмірністю 5 і кожний вимір відповідає одній з п'яти персональних психологічних рис. Також якщо користувач поставив «хорошу» оцінку хоча б одному продукту в системі, то схожих користувачів системи необхідно шукати серед користувачів, які також поставили «хорошу» оцінку хоча б одному спільному товару з активним користувачем. Отже, подібність заснована на персональних даних між двома користувачами *u* та *v* може бути обчислена наступним чином:

$$simp(u,v) = \sum_k 1 - \frac{|p_u^k - p_v^k|}{\max(p^k)}.$$

6. Оцінка ефективності

На виході системи ми отримуємо список рекомендацій, що представляє собою список з *n* продуктів, які з найбільшою вірогідністю може купити активний користувач. Для оцінки ефективності створених рекомендацій використаємо чутливість. Чутливість це статистичний показник ефективності бінарної класифікації, вимірює частку позитивних результатів які були правильно ідентифіковані. Тобто в нашому випадку

позитивний правильно ідентифікований результат це продукт системи $t_i \in T$, який придбає користувач в майбутньому та який знаходиться в списку рекомендацій. Таким чином чутливість ми можемо розрахувати за формулою:

$$TRP = \frac{|T|}{n}.$$

7. Результати дослідження

В системі рекомендацій на виході отримуємо список рекомендацій, що складається з n продуктів системи. При порівнянні алгоритмів ми змінювали значення n від 1 до 20 з кроком 1. Результати показані на рисунку 2.

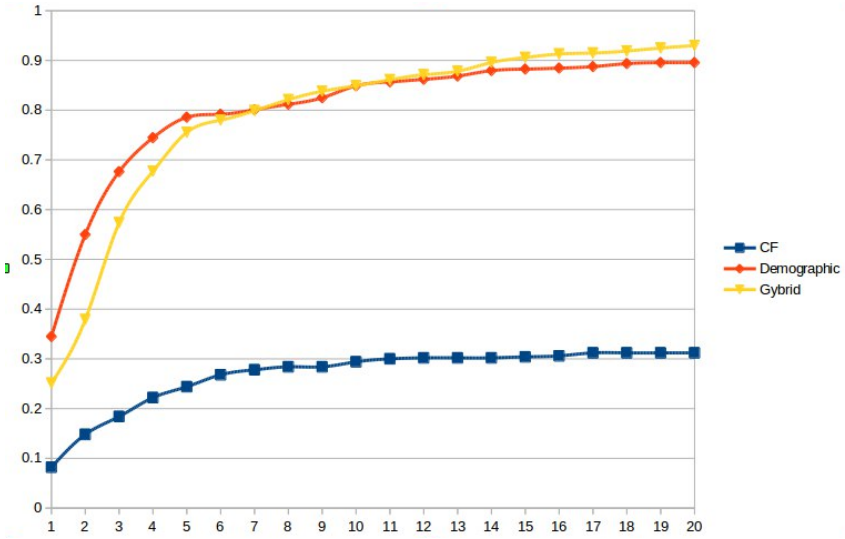


Рис. 2. Порівняння алгоритмів

8. Висновки

У цій статті ми запропонували врахувати персональні характеристики користувачів разом з їх вподобаннями в рекомендаційній системі для вирішення проблеми «холодного старту». Ми експериментально оцінили якість створених рекомендацій порахувавши чутливість для кожного з алгоритмів: алгоритму на основі персональних даних, традиційного алгоритму на основі рейтингів користувачів та запропонованого гібридного методу. Наші результати показали, що застосування методу заснованого на персональних даних та гібридного методу в рекомендаційній системі дають кращі результати ніж традиційний метод заснований на рейтингових оцінках користувачів. Зокрема алгоритм заснований на персональних даних показав кращі результати при $n \in [0; 8]$, а при $n \in [9; 20]$ кращі результати показав запропонований гібридний алгоритм.

Список літератури

1. Adomavicius, G. and Tuzhilin, A. 2005. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Trans. Knowledge and Data Eng.
2. J. L. Herlocker, J. A. Konstan, A. Borchers, J. Riedl, An algorithmic framework for performing collaborative filtering.
3. G. Salton, M. McGill, Introduction to Modern Information Retrieval
4. Ahn, H. J. 2008. A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. Information Sciences.
5. Park, S.-T., Pennock, D. M., Madani, O., Good, N., and DeCoste, D.2006. Naive filterbots for robust cold-start recommendations, in: Proceedings of KDD 06, ACM, Philadelphia, PA, USA.

6. Gosling, S. D., Rentfrow, P. J., and Swann, Jr. W.B.2003. A very brief measure of the Big-Five personality domains. Journal of Research in Personality.

УДК 519.21; 681.513.7; 612.8.001.57; 007.51/.52

ХАНЬКО Г.В.

ОГЛЯД АЛГОРИТМІВ ТЕКСТОВОГО АНАЛІЗУ ДАНИХ

Робота присвячена розвитку загального підходу до методів інтелектуального аналізу даних для класифікації текстової інформації. У якості запропонованих підходів до вирішення завдання розглядаються алгоритми TF-IDF, критерію χ^2, алгоритм k-NN, метод опорних векторів та наївний баєсівський класифікатор.

Мета даної роботи полягає у підвищенні точності класифікації текстової інформації за рахунок доцільного використання наявних методів інтелектуального аналізу даних, використовуючи найбільш ефективні методи пре обробки тексту та потужних алгоритмів машинного навчання для задач класифікації.

Розглянуті методи для вирішення задачі дозволяють вирішити проблему класифікації текстової інформації для задач фільтрації спама, контекстної реклама, категоризації новин, створення тематичних каталогів.

The work is dedicated to the development of a common approach to the methods of data mining to categorize textual information. The proposed approaches to the problem are TF-IDF algorithm, the criterion $\chi 2$, algorithm k-NN, support vector method and naive Bayesian classifier.

The aim of this work is to improve the classification accuracy of text information through appropriate use of existing data mining techniques, using the most effective methods of powerful word preprocessing and machine learning algorithms for classification problems.

The methods for solving the problem can solve the problem of classification of text information for the tasks of filtering spam, advertising, news categorization, the creation of thematic catalogs.

1. Вступ

Сьогодні важливість інтелектуального аналізу текстових документів стрімко росте. Саме тому необхідні застосування, які допоможуть знайти, відфільтрувати та структурувати текстові данні. Задача текстової класифікації має практичне застосування у багатьох сферах, наприклад фільтрація спама, контекстна реклама, категоризація новин, створення тематичних каталогів. Більшість методів автоматичної класифікації текстів засновані на припущенні, що тексти кожної тематичної рубрики містять певні ознаки (слова та словосполучення), наявність або відсутність яких говорить про належність тексту тієї чи іншої рубрики. Задача методів класифікації полягає в тому, щоб вибрати найкращі такі ознаки і сформулювати правила, які будуть приймати рішення про віднесення тексту до певної категорії та проведенні інтерактивного буріння (розбиття певних класів на підкласи).

Існуючі засоби текстового аналізу даних (наприклад система Statistica) не дають змоги отримати бажані результати при розв'язанні задач класифікації текстової інформації, тому виникає потреба в розробці нових алгоритмів.

2. Попередня обробка даних

Перед тим, як вибирати ознаки, за якими алгоритм буде визначати належність тексту до певної категорії, треба зробити пре обробку текстових документів. По-перше, текст треба токенізувати. Токенізація – це процес розбиття тексту на більш дрібні об'єкти, наприклад речення, фрази або слова. Токенізація необхідна для того, щоб представити документ у вигляді масиву токенів. Далі виконується стемінг або лематизація токенів. Мета стемінгу та лемінгу – привести словоформи та похідні форми слова до її загальної основної форми. Але є деякі відмінності між цими методами. Лематизація – процес, що

приводить словоформу до леми – її нормальної(словникової) форми, використовуються лексикон та морфологічний аналіз слів, видаляються тільки флективні закінчення і залишається тільки основна словникова форма. Стемінгом називається евристичний процес, яким полягає у видаленні закінчень в розрахунку на те, що в більшості випадків це себе виправдає. Надалі видаляються так звані «стоп-слова»: прийменники, сполучники, займенники і т.д., тобто слова, які не несуть цінної інформації про зміст документа.

Для багатьох алгоритмів машинного навчання необхідно зменшувати розмірність простору ознак, якщо від самого початку вона завелика. У більшості випадків це покращує не тільки швидкість алгоритму, але також і точність класифікації. Зважування слів - це процес надання ваги словам, в залежності від їх важливості для класифікації тексту. Важливість слів визначається їх здатністю характеризувати відмінність між категоріями в корпусі документів.

Одним з найбільш вживаним методів вибору ознак є використання TF-IDF метрики. TF-IDF (від англ. TF - termfrequency, IDF - inversedocumentfrequency) - статистична міра, яка використовується для оцінки важливості слова в контексті документа, що є частиною колекції документів або корпусу. Вага деякого слова пропорційна кількості вживання цього слова в документі, і обернено пропорційна частоті вживання слова в інших документах колекції.

$$tf - idf(t,d,D) = \frac{n_t}{\sum_k n_k} \times log \frac{|D|}{|\{d_i \in D \mid t \in d_i\}|},$$

де n_t – кількість входів слова t в документ, $\sum_k n_k$ - загальна кількість слов у документі, |D| - кількість документів в корпусі, $|\{d_i \epsilon D | t \epsilon d_i\}|$ - кількість документів з колекції D, в яких зустрічається t.

Більшу вагу в TF-IDF отримують слова з високою частотою у межах конкретного документа і з низькою частотою вживань в інших документах.

Критерій χ^2 є також популярним методом фільтрації ознак.В статистиці критерій χ^2 використовується для перевірки незалежності двох подій. Випадкові величини A та B називаються незалежними, якщо P(A·B) = P(A) · P(B). В задачі вибору оптимального набору ознак подія A – це поява об'єкта з ознакою x^i, а подія B – це поява об'єкта з класом c_j.Тоді статистика для перевірки критерія χ^2 має наступний вигляд:

$$\chi^2(D, x^i) = \sum_{x^i \in \{0,1\}} \sum_{c^i \in \{0,1\}} \frac{(\hat{V}_{x^i,c_j} - V_{x^i,c_j})^2}{V_{x^i,c_j}}$$

V – емпірична частота події $A \cap B$, а $\hat{V}$ - це очікувана частота у припущенні, що ознака x^i і клас c_jнезалежні. Критерій говорить про те, наскільки емпірична частота V та очікувана частота $\hat{V}$ відрізняються. Більше значення χ^2свідчить про те, що невірна гіпотеза про незалежність випадкових величин x^i і c_j(тобто ознака x^i може бути корисним для класифікації).

3. Класифікація даних

Після вибору ознак документи можуть бути легко представлені у формі, яку можуть використовувати алгоритми машинного навчання. Найбільш популярні алгоритми, які застосовуються для вирішення задачі класифікації тексту, це дерева рішень, наївний байєсівський класифікатор, метод опорних векторів та логістична регресія.

Метод k-NN. Метод k-NN не вимагає навчання. Для того, щоб знайти категорії, які відповідають документу, класифікатор порівнює d_j з усіма документами навчальної виборкиd_z. Потім з навчальної вибірки вибираються k документів, які є найближчими до d_j. Для категорій обраховуються функції ранжування за формулою

$$\sum_{d_z \in L_k(d_j)} p(d_j, d_z) \cdot \Phi(d_z, c_i),$$

Де $L_k(d_j)$ - найближчі k документів з L до d_j. Параметр k найчастіше вибирається у інтервалі 20…50.

Цей метод досить високоефективний, але потребує багато обчислювальних ресурсів на етапі класифікації.

Метод опорних векторів. Метод опорних векторів є лінійним класифікатором, основна ідея якого полягає у переведенні початкових векторів у простір більш високої розмірності і пошук гіперплощини, що розділяє класи з максимальним зазором. Дві паралельні гіперплощини будуються по сторонам гіперплощини, що розділяє класи. Алгоритм працює при допущенні, щоб чим більше відстань між цими двома паралельними гіперплощинами, тим менше буде середня похибка класифікатора.

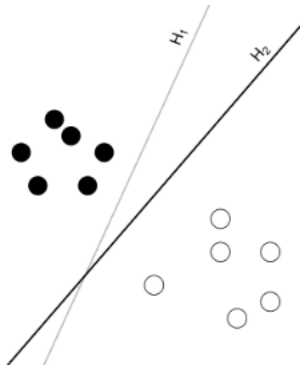


Рис.2 – Дві гіперплощини, що розділяють класи, але тільки H_2 розділяє їх з найменшим зазором

Нехай нам дана вибірка D, яка складається з n елементів:

$$D = ((x_i, y_i)|x_i \in R^p, y_i \in \{-1,1\})$$

Кожен елемент x_i є вектором розмірності p. Потрібно знайти таку гіперплощину максимальної різниці, яка б розділяла класи $y_i = 1$ та $y_i = -1$. Будь-яку гіперплощину можна записати у вигляді множини точок:

$$w*x - b = 0$$

Параметр $\frac{b}{||w||}$ визначає зміщення гіперплощини від початку координат вздовж нормалі w. Для того, щоб відстань між гіперплощинами була максимальною, треба вирішити задачу оптимізації:

$$||w|| \to min$$

$$y_i(w \cdot x_i - b) \geq 1 \text{ для } 0 \leq i \leq n$$

Для класифікаторів, побудованих на базу методу опорних векторів, не потрібно зменшувати розмірність простору ознак, бо вони досить добре масштабуються та стійкі до перенавчання.

Наївний баєсівський класифікатор. Наївний баєсівський класифікатор – це простий класифікатор, який заснований на застосування теореми Баєса з наївними припущеннями щодо незалежності подій. В задачі класифікації текстових документів для оцінки параметрів для наївних баєсівськихмоделей використовують метод максимальної правдоподібності. Перевагою цього методу є мала кількість даних для навчання, необхідних для оцінки параметрів для класифікації.

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)},$$

де $P(c|d)$- ймовірність що документ d належить класу c, саме її треба розрахувати; $P(d|c)$ - ймовірність зустріти документ d серед всіх документів класу c;
$P(c)$ – безумовна ймовірність зустріти документ класу c в корпусі документів;
$P(d)$ - безумовна ймовірність документа d в корпусі документів.
Баєсівський класифікатор використовує оцінку апостеріорного максимуму (Maximum a posterioriestimation) для визначення найбільш вірогідного класу.

$$c_{map} = \arg max \frac{P(d|c)P(c)}{P(d)}$$

Окрім вищезазначених методів для класифікації широко використовуються наступні: метричні алгоритми класифікації (класифікатор Роше); класифікатори на основі вирішальних правил; лінійні класифікатори (класифікатори на основі регресії, нейронні мережі).

Висновки

У даній роботі розглянуто застосування методів інтелектуального аналізу даних для класифікації текстової інформації. Розглянуто попередню обробку текстових документів за допомогою токенізації, лемінгу, стемінгу та видалення стоп-слів. Розібрані основні методи для вибору ознак, такі як TF-IDF метрика для оцінки важливості слова у межах корпусу документів та популярний метод фільтрації ознак критерій χ^2. Описані найбільш ефективні алгоритми класифікації текстової інформації, такі як алгоритм k-NN, метод опорних векторів та наївний баєсівський класифікатор.

Список літератури

1. T. S. Guzellaand W. M. Caminhas, “A reviewofmachinelearningapprochestospamfiltering,” Elsever, ExpertSystemwithApplications, 2009.– с. 11–20.
2. C.-H. Leeand H.-C. Yang, “Constructionofsupervisedandunsupervisedlearningsystemsformultilingualtextcategorization, ” ExpertSystemswithApplications, 2009. – с. 2400–2410.
3. A. Markovand M. Last, “A simple, structure-sensitiveapproachforwebdocumentclassification,” inAtlanticWebIntelligenceConference - AWIC, 2005. - с. 293–298.
4. Vinciarelli A., “NoisyTextCategorization, PatternRecognition”, 17th InternationalConferenceon (ICPR'04) , 2004. – с. 554-557.
5. Bao Y. andIshii N., “CombiningMultiple k-NN ClassifiersforTextCategorizationbyReducts”, LNCS 2534, 2002.. – 340-347 с.

УДК 51-74

ШУЛЬКЕВИЧ Т.В.
БАКЛАН І.В.
СЕЛІН Ю.М.

ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ ПРИ ПРОГНОЗУВАННІ НЕЛІНІЙНИХ НЕСТАЦІОНАРНИХ ПРОЦЕСІВ РІЗНОЇ ПРИРОДИ

Інтелектуальний аналіз даних (ІАД) - термін, що застосовується для опису здобуття знань у базах даних, дослідження даних, обробки зразків даних, очищення і збору даних. Це процес виявлення кореляції, тенденцій, шаблонів, зв'язків і категорій. Інтелектуальний аналіз даних розвивається на базі таких наук, як прикладна статистика, розпізнавання образів, штучний інтелект, теорія баз даних тощо.

Але проблема прогнозування нелінійних нестаціонарних часових рядів процесів парадигми якості життя є проблемою міждисциплінарною. Оскільки, не дивлячись на достатньо розвинений математичний апарат аналізу та прогнозування окремих складових, можна констатувати наступне. Математичні апарати є несхожими, тож фахівці економісти не володіють засобами обробки екологічних даних і навпаки, екологи не використовують мат апарат прогнозування економічних показників. Тож, можна констатувати відсутність математичного апарату, що можуть спільно використовувати фахівці різних напрямків.
Ключові слова: інтелектуальний аналіз даних, нелінійні процеси, нестаціонарні процеси, приховані марковські моделі, метод подібних траекторій, метод лінгвістичного моделювання

Data mining (DM) - a term used to describe the acquisition of knowledge in databases, research data, sample processing data cleansing and data collection. This process of identifying correlations, trends, patterns, relationships and categories. Data mining is developing on the basis of such sciences as applied statistics, pattern recognition, artificial intelligence, data base theory and so on.

But the problem of predicting nonlinear non-stationary time series process paradigm of quality of life is an interdisciplinary problem. Because, despite the rather advanced mathematical tools of analysis and prediction of individual components, we can say the following. Mathematical devices are dissimilar, so economists experts do not own the means of the processing of environmental data and vice versa, environmentalists do not use mathematical apparatus forecasting economic performance. So, we can say the lack of mathematical tools that can be used together specialists in different areas.

Keywords: Data mining, nonlinear process, non-stationary processes, hidden markov model, method similar trajectories, methods of linguistic simulation

Вступ

Сутність і мету технології інтелектуального аналізу даних (ІАД) можна описати таким чином: технологія, яка призначена для пошуку у великих інформаційних масивах даних неочевидних, об'єктивних, корисних на практиці закономірностей. ІАД здійснюється за допомогою використання технологій розпізнавання шаблонів, а також статистичних і математичних методів.

При розвідці даних багаторазово виконуються операції і перетворення над "сирими" даними (відбір ознак, стратифікація, кластеризація, візуалізація і регресія), що призначені для знаходження: структур, які інтуїтивно зрозумілі для людей і краще розкривають суть бізнес-процесів, що лежать в основі їх протікання; моделей, які можуть передбачити результат або значення певних ситуацій, використовуючи історичні або суб'єктивні дані.

Задачі що вирішує ІАД: Classification; Associations; Sequence; Forecasting;

Deviation Detection; Estimation; Link Analysis; Visualization, Graph Mining;

Summarization.

Зазначені вище задачі поділяються за призначенням на описові і предиктивні.

Описові, або дескриптивні (descriptive), задачі пов'язані з поліпшенням розуміння аналізованих даних. Ключовий момент у таких моделях - простота і прозорість результатів для сприйняття людиною. До такого типу задач належать кластеризація і пошук асоціативних правил.

Рішення предиктивних (predictive), або прогнозуючих, задач поділяється на два етапи. На першому етапі на підставі набору даних з відомими результатами будується модель. На другому етапі вона використовується для прогнозу результатів на підставі нових наборів даних. Вимагається, щоб побудовані моделі працювали максимально точно. До цього типу задач відносять задачі класифікації і регресії. Сюди можна віднести і задачу пошуку асоціативних правил, якщо результати її рішення можуть бути використані для прогнозу появи деяких подій.

Коротко розглянемо деякі методи, що можна застосовувати для аналізу та прогнозування нелінійних нестаціонарних процесів в задачах прогнозування якості життя. До них можна віднести наступні методи:

- Приховані марковські моделі;
- Метод подібних траєкторій;
- Лінгвістичне моделювання.

Розглянемо ці методи.

1. Метод прихованих марковських моделей (ПММ).

ПММ може визначатися наступним чином [1,2,3]:

N – кількість станів

T – кількість спостережень

$\theta_{i=1\ldots N}$ – параметр розподілення спостереження для стану

$\sigma_{i=1\ldots N, j=1\ldots N}$ – вероігідніть переходу зі стану I в стан j

$\sigma_{i=1\ldots N}$ – N-вимірний вектор, складається з $\sigma_{i,1\ldots N}$, сума елементів дорівнює 1.

$x_t = 1..T$ – прихований стан у час t

$y_t = 1..T$ – спостережувальний стан у час t

$F(y \mid \theta)$ – вероігідність розподілу спостереження, параметризована по θ

$x_t = 2..T \sim (\sigma_{x_{t-1}})$

$y_t = 1..T \sim F(\theta_{x_t})$

Для c_i підрахуємо частоту існування пар символів $(c_i c_{i+1}) j = \overline{1, N-1}$ та побудуємо таблицю ймовірностей виникнення символу $c_{i+1} P_{j+1}(c_{i+1} \mid c_i)$.

В послідовності c_i підрахуємо частоту існування трійок $(c_{i-1} c_i c_{i+1}) j = \overline{1, N-2}$ та будуємо таблицю ймовірностей $P_{j+1}(c_{i+1} \mid c_i c_{i-1})$. Аналізуємо частоту існування послідовностей $P_{j+1}(c_{i+1} \mid c_{i-k} \ldots c_i)$. (ймовірність появи символуc_{i+1} за умови, які відомі послідовності попередніх символів) [2]

Основні завдання при застосуванні ПММ до визначення параметрів процесів. Для використання ПММ при розпізнаванні мови необхідно вирішити три задачі [4].

Завдання 1: Якщо задані послідовність спостережень і модель $\lambda = (A, B, \Pi)$ то як ефективно обчислити $P(O \mid \lambda)$ - вірогідність такої послідовності при заданих параметрах моделі?

Завдання 2: Якщо задані послідовність спостережень і модель $\lambda = (A, B, \Pi)$ то як визначити відповідну послідовність внутрішніх станів ?

Завдання 3: Якщо задані послідовність спостережень, то як визначити параметри моделі $\lambda = (A, B, \Pi)$, виходячи з критерію максимізациі $P(O \mid \lambda)$?

2. Метод «подібних траєкторій».

Однією з проблем прогнозування часової послідовності є те, що існує можливість збільшення вимірів даних за рахунок зменшення періоду дискретизації чи за допомогою інтерполяції; тобто можна отримати більше даних, не додаючи при цьому нової інформації. Це є проблемою, оскільки у дискретизованого із надто великою частотою сигналу усі найближчі траєкторії будуть знаходитись у часовій послідовності поруч один до одного.

Опишемо алгоритм пошуку найближчих траєкторій при наявності алгоритму пошуку найближчих точок [6].

Нехай знайдена точка x_i, яка знаходиться ближче за k-ту найближчу точку, знайдену алгоритмом раніш.

Підрахуємо відстані до попередніх точок послідовно від точки x_i (x_{i-1}, x_{i-2} ,...) для пошуку найближчого локального мінімуму. Повторимо цю процедуру для точок, що йдуть після x_i (x_{i+1}, x_{i+2} ,...). Локальний мінімум позначимо за $x_{\min}$. Це буде найближча точка в даному сегменті траєкторії.

Виключимо інші точки цього сегменту з подальшого розгляду. Для цього підрахуємо відстані до попередніх точок послідовно від точки x_i (x_{i-1}, x_{i-2} ,...), поки не досягнемо локального максимуму, або відстань перевищить відстань до k-тої найближчої точки, знайденої раніше. Позначимо цю точку за $x_{\max}$ і виключимо з подальшого розгляду точки, що знаходяться між $x_{\min}$ та $x_{\max}$.

Повторимо попередній крок для точок, що йдуть після $x_{\min}$. Замінимо k-ту найближчу точку точкою $x_{\min}$ і продовжимо пошук найближчих точок.

Дамо графічну інтерпретацію методу подібних траєкторій". Ідея методу полягає в наступному. Маємо ряд спостережень екологічного процесу, що їх зроблено за якийсь час $\{y(1); y(2); ...; y(n)\}$.

Змінна $y(i), \quad i = \overline{1, N}$ тут представлена фізичними значеннями відповідного процесу (наприклад, сила вітру, інтенсивність стоку води, сила підземних поштовхів).

За обраним критерієм обирається ділянка траєкторії "найближча" до ділянки, яка передує прогнозованій точці. Надалі оцінюється прогноз за формулою $y(n+1) = y(i+p)$, де

$$I = \min\left\{\sum_{i=1}^{p} | y(j+i-1) - y(n-p+i) | \right\} \quad J = 1,2,...,n-p;$$

I = min| y(i + j - 1) - y(n)| i = I,I + 1,…,I + p - 1

Формалізувати метод можна наступним чином. Нехай ми маємо наступні вектори спостережень $Y_1 = (y, y_2, \dots, y_P)^T$; $Y_2 = (y_2, y_3, \dots, y_{p+1})^T$; …; $Y_K = (y_k, y_{k+1}, \dots, y_{k+p+1})^T$; …; $Y_N = (y_{n-p+1}, y_{n-p+2}, \dots, y_n)^T$;

Знаходимо найближчу точку із умови мінімальної відстані

$$Y_k = \arg\min_j d(Y_n, Y_j).$$

Є й інші способи пошуку найближчої точки, наприклад, найбільш поширена метрика – квадрат евклідової відстані

$$d(Y_k, Y_n) = (Y_k - Y_n)^T (Y_k - Y_n).$$

3. Метод лінгвістичного моделювання.

3.1. Побудова лінгвістичної моделі

Для досягнення поставленої мети має бути вирішена задача знаходження лінгвістичного образу часового ряду, який включає в собі: обчислення різницевих рядів вихідного часового ряду; вибір критерію інтервалізації різницевих рядів; інтервалізація певного різницевого ряду згідно обраного критерію; знаходження лінгвістичного ланцюжка для певного різницевого ряду; знаходження матриці переходів для кожної можливої пари символів у лінгвістичному ланцюжку певного різницевого ряду.

Вхідними даними для даної задачі є значення часового ряду.

Вихідними даними для цієї задачі є лінгвістичний образ часового ряду (динамічного процесу), що являє собою:

– множину інтервалів, отриману в результаті інтервалізації різницевого ряду певного порядку від часового ряду;

– матрицю переходів (передування), побудовану на множині інтервалів (описаній вище) та по часовому ряду.

Вказаний лінгвістичний образ будується окремо для різницевих рядів, вхідного часового ряду, різних порядків [6-8]. Таким чином отримуємо множину лінгвістичних образів, що і є проміжним результатом задачі прогнозування з використанням лінгвістичного моделювання.

Надалі буде розглядатися підхід лінгвістичного моделювання для побудови

лінгвістичного образу вхідного часового ряду.

Згідно з етапів побудови лінгвістичної моделі вихідна задача буде розбита на такі підзадачі:

– підзадача отримання різницевих рядів;
– підзадача інтервалізації;
– підзадача лінгвістизації;
– підзадача побудови матриці переходів.

3.2. Підзадача отримання різницевих рядів.

Призначенням даної підзадачі є отримання рядів, котрі характеризують динаміку зміни руху курсору «мишки»: швидкість(різницевий ряд 1-ого порядку), прискорення(різницевий ряд 2-ого порядку) і т. д. Таким чином різницеві ряди являються похідними від вихідного ряду.

Дано: Вектор цілих чисел $\overline{X}$ з потужністю $n = \left|\overline{X}\right|$.

Результати: Вектор цілих чисел $\overline{D}$ з потужністю $k = \left|\overline{D}\right|$.

Обмеження:

$$\forall d_i \in \overline{D}: \quad d_i = x_{i+1} - x_i$$

де $i \in [0; n-1)$; $x_{i+1}, x_i \in \overline{X}$; $k = n - 1$;

3.3. Підзадача інтервалізації.

Призначенням даної підзадачі є побудова алфавіту користувача шляхом розбиття відсортованого різницевого ряду на множину інтервалів, кожний елемент якої характеризує певну літеру алфавіту.

Дано:

– гіпотетична потужність алфавіту a;
– вектор цілих чисел $\overline{D}$ з потужністю $k = \left|\overline{D}\right|$.

Результати:

Вектор пар цілих значень $\overline{I}$ з потужністю $k = \left|\overline{I}\right|$.

Обмеження:

$$\forall x \in \overline{I}: \quad x^1 \le x^2 \tag{1}$$

$$\forall x_i, x_{i+a} \in \overline{I}: \quad x_i{}^2 \le x^1{}_{i+1}, \quad \text{де} \tag{2}$$

$i \in [0; n-1)$;

$n \le a$

$a << k$

$$\exists x \in \overline{I}: \quad \forall d \in \overline{D}, \quad d \in \left[x^1; x^2\right] \tag{3}$$

$$\forall d_i; d_{i+1} \in \overline{D}: \quad d_i \le d_{i+1}, \quad \text{де}$$

$i \in [0; k-1)$;

$$x_0 \in \overline{I}: \quad x_0 \in \left(-\infty; x_1^1\right) \tag{4}$$

$$x_n \in \overline{I}: \quad x_n \in \left(x_{n-1}^2; +\infty\right) \tag{5}$$

3.4. Підзадача лінгвістизації.

Призначенням даної підзадачі є отримання лінгвістичного ланцюжку шляхом знаходження відповідної літери алфавіту для кожного значення різницевого ряду. Літера алфавіту однозначно відповідає певному інтервалу з множини інтервалів, отриманих в результаті розв'язання попередньої задачі.

Дано:

вектор цілих чисел $\overline{D}$ з потужністю $k = \left|\overline{D}\right|$, що відповідає обмеженню представленому у формулі 3;

вектор пар цілих значень $\overline{I}$ з потужністю $n = \left|\overline{I}\right|$ з обмеженнями, що наведені у формулах 1 та 3, а також у 4 та 5.

Результати:

Вектор цілих чисел $\overline{A}$ з потужністю $k = \left|\overline{A}\right|$.

Обмеження:

$$\forall x_i \in \overline{A}: \quad \exists d_i \in \overline{D}, \exists y_i \in \overline{I}, d_i \in \left[y_j^1; y_h^2\right], x_i = j$$

,де $i \in [0; k)$, $j \in [0; n)$; (6)

3.5. Підзадача побудови матриці переходів.

Призначенням даної підзадачі є побудова матриці переходів між двома літерами алфавіту в реченні. Алфавіт та його літери визначені у підзадачі

інтервалізації, а речення – у задачі лінгвістизації.

Дано:

– вектор цілих чисел $\overline{A}$, що відповідає обмеженню 6 з потужністю $k = \left|\overline{A}\right|$;

– потужність множини інтервалів n, отриманої в результаті розв'язання підзадачі інтервалізації.

Результати: квадратна матриця раціональних чисел $\overline{P}$ розмірністю n.

Обмеження:

$$\forall x_{ij} \in \overline{P} : x_{ij} \in [0.0;1.0], \text{ де } i, j \in [0;n)$$

Отриману послідовність аналізують на наявність граматичних конструкцій. На виході отримуємо список граматичних конструкцій з імовірностями їх наявності в процесі, а також матрицю імовірностей переходу з символу в символ. Цей етап тісно перекликається з моделюванням (прихованих) марківських процесів, а також з методом подібних траєкторій.

Висновки. Таким чином було наведено математичний апарат, що об'єднує в собі три види подачі інформації за для її аналізу та прогнозування. Перший вид -- звичний числовий, що зустрічається чи не в 99% відомих авторам джерел, другий – графічний, що ще й досі отримуються з аналогових приладів реєстрації та третій – символьний (метод лінгвістичного моделювання), що ще не є розповсюдженим. Для методу лінгвістичного моделювання сформульована змістовна та математична постановки задачі знаходження лінгвістичних образів часових рядів. Було введено поняття лінгвістичного моделювання та поетапно описано спосіб застосування даного підходу для розв'язання поставленої задач. Також, для більш повного розгляду задачі, були введені додаткові підзадачі, описані їх математичні постановки та наведені приклади їх виконання.

Наведені методи є універсальними як з боку виду отриманої інформації так і з боку наявності в цій інформації нелінійностей та нестаціонарностей. Але мають загальний недолік всіх статистичних методів – брак історичної інформації.

Список литературы

1. Juang, B. H., Rabiner, L. R. Hidden Markov models for speech recognition, Technometrics, 1991.
2. Baklan I., Komada P. Hybrid hidden Markov models // Elektronika (LIV). - No 8/2013. – P.28-31.
3. Морозова О.А., Баклан І.В. Застосування лінгвістичного моделювання для прогнозування часових рядів // Комп'ютерно-інтегровані технології у сьогоденні: збірка наукових праць молодих вчених (студентів, магістрів і аспірантів) / [Під редакцією Г.В. Рудакової та ін]. – Херсон: вид-во ПП Вишемирськиий В. С., 2016. – С.26-29.
4. Rabiner L.R., «A tutorial on Hidden Markov Models and Selected Applications in Speech Recognition», Proceedings of the IEEE, vol, 77. no.2, February 1989, pp. 257-284.
5.] G. Kollios, D. Gunopulos, and V. J. Tsotras. Nearest neighbor queries in a mobile environment. In Spatio-Temporal Database Management 1999, pp. 119–134.
6. Баклан І. В. Аналіз поведінки економічних часових рядів з використанням структурних підходів. Сборник МКММ-2006. — Херсон: ХГТУ, 2006.
7. Баклан І. В. Структурний підхід до розпізнавання образів у системах безпеки. Національна безпека України: стан, кризові явища та шляхи їх подолання. Міжнародна науково-практична конференція (Київ, 7-8 грудня 2005 р.). Збірка наукових праць. — К.: Національна академія управління — Центр перспективних соціальних досліджень, 2005. — С.375-380.
8. Баклан І. В. Лінгвістичне моделювання: основи, методи, деякі прикладні аспекти. Систем. технології. — 2011. — № 3. — С. 10-19.

9. Fu K. S., Sequential Methods in Pattern Recognition and Machine Learning. — Academic Press, 1968.

УДК 517.977.8

ДЕХТЯРУК О.М.
ЧИКРІЙ А.О.

ПРО МЕТОД РОЗВ'ЯЗУЮЧИХ ФУНКЦІЙ В ІГРОВИХ ЗАДАЧАХ ДИНАМІКИ

Дане дослідження відноситься до одного з розділів математичної теорії керування, де розглядаються об'єкти, що рухаються, які функціонують в умовах конфлікту або невизначеності. Розглянуто квазілінійні конфліктно-керовані процеси загального вигляду з циліндричною термінальною множиною. Особливим є представлення розв'язку у вигляді суми довільної функції та блоку керування. Це дозволяє розглянути більш широкий клас динамічних задач. При дослідженні в якості базового використовується метод розв'язуючих функцій.

Ключові слова: КОНФЛІКТНО-КЕРОВАНІ ПРОЦЕСИ, ТЕРМІНАЛЬНА МНОЖИНА, КЕРУВАННЯ, ДИНАМІЧНА ГРА, ВТІКАЧ, ПЕРЕСЛІДУВАЧ.

This study refers to one of the sections of mathematical control theory, which deals with moving objects that operate in conflict or uncertainty. Considered quasi-linear conflict-controlled processes of general form of a cylindrical terminal set. The special is presentation of solution as the sum of an arbitrary function and control unit. It allows to consider a wider class of dynamic problems. The study uses the method of resolving functions as base.

Key words: CONFLICT-DRIVEN PROCESSES, TERMINAL SET, CONTROL, DYNAMIC GAME, FUGITIVE PURSUER.

1. Вступ

Дане дослідження відноситься до одного з розділів математичної теорії керування, де розглядаються об'єкти, що рухаються, які функціонують в умовах конфлікту або невизначеності. Розглянуто квазілінійні конфліктно-керовані процеси загального вигляду з циліндричною термінальною множиною. Особливим є представлення розв'язку у вигляді суми довільної функції та блоку керування. Це дозволяє розглянути більш широкий клас динамічних задач. При дослідженні в якості базового використовується метод розв'язуючих функцій.

2. Постановка проблеми

Будемо розглядати класичні задачі переслідування та втечі, які є центральними в теорії конфліктно-керованих процесів. Вони були в основі виникнення теорії, є найбільш змістовними і представляють значний інтерес для дослідників. Поштовх до їх розвитку дали реальні прикладні задачі в економіці, космічній техніці, у воєнній справі, біології, медицині, тощо.

Конфліктно-керовані процеси це розділ математичної теорії керування, який вивчає керування рухомими керованими об'єктами, що функціонують в умовах конфлікту та невизначеності. Для визначення цього кола питань також використовують термін динамічні ігри, диференціальні ігри.

При цьому еволюція об'єкту може описуватися системами різницевих, звичайних диференціальних, диференціально-різницевих, інтегральних, інтегро-диференціальних рівнянь, системами рівнянь з розподіленими параметрами, системами рівнянь з дробовими похідними, імпульсним впливом та їх різними комбінаціями (гібридними системами).

Термін диференціальні ігри використовують стосовно ігор, в яких динаміка об'єкту описується системою звичайних диференціальних рівнянь. Якщо ж процес описується більш складними рівняннями, то вживають термін динамічні ігри. І, нарешті, конфліктно-керовані процеси – це найбільш загальний термін для визначення кола питань, що стосуються ігрових задач.

Розрізняють динамічні ігри двох або декількох гравців. Змістовно динамічні ігрові задачі можна описати наступним чином. Є динамічна система, в якій частина керуючих параметрів перебуває під контролем першого гравця (переслідувача), а інша частина контролюється другим гравцем (втікачем).

При постановці задачі, що стоїть перед першим або другим гравцем, припускають, що вибір керувань цього гравця, який гарантує йому досягнення певної мети при будь-якому заздалегідь невідомому керуванні супротивника, може спиратися лише на деяку інформацію про поточні стани системи або на інформацію про миттєві керування супротивника.

Першим, хто систематично досліджував динамічні ігри, був Р.Айзекс. Його монографія [1] поклала початок теоретичним дослідженням динамічних ігрових задач. Вона показала, що існує широке коло задач прикладного плану, які мають ігровий характер і не вкладаються в рамки сформованої теорії оптимального керування. Її цінність полягає в значній кількості постановок задач, які носять проблематичний характер. До них відносяться задачі з неповною інформацією, групами гравців та з фазовими обмеженнями, важливість та складність вирішення яких він неодноразово підкреслював.

Відрізняють динамічні гри двох типів – гри ступеню та гри якості [1]. На траєкторіях динамічної системи заданий деякий функціонал, що залежить від початкового стану та від керувань гравців.

В іграх першого типу мета першого гравця – мінімізувати цей функціонал, заданий на траєкторіях системи, мета іншого – максимізувати його. В іграх другого типу цим функціоналом є час виходу траєкторії об'єкту на задану термінальну множину і задача полягає в аналізі можливості виведення переслідувачем траєкторії системи на термінальну множину (задача зближення) чи відхилення втікачем траєкторії від цієї множини (задача відхилення).

Якщо ж рухи гравців незалежні, описуються окремими системами диференціальних рівнянь, то замість зближення-відхилення іноді вживають терміни переслідування-втечі. Відхиленню та втечі еквівалентний вираз уникнення сутичок.

Обмеження на керуючі параметри гравців можуть бути геометричними, інтегральними, імпульсними або змішаними. Можливі фазові обмеження на змінні, що визначають стан конфліктно-керованого процесу.

Відомі стратегії переслідування здебільшого розроблено для військового призначення. Їх можна розподілити на дві підгрупи:

- методи з фіксованим положенням необхідного направлення вектора швидкості відносно лінії переслідувач – втікач (до таких методів належать метод погонної кривої та метод постійного випередження);
- методи зі змінним положенням необхідного направлення вектора швидкості відносно лінії переслідувач – втікач (до таких методів належать метод пропорційного зближення та метод паралельного переслідування).

В якості базового методу при вивченні динамічних ігор використовується метод розв'язуючих функцій.

Для динамічних систем, еволюція яких описується інтегральним рівнянням, з циліндричною термінальною множиною при умові Л.С.Понтрягіна введена розв'язуюча функція, через неї визначено час закінчення гри. Особливістю основної схеми методу є та обставина, що час закінчення гри залежить від деякого селектора, вибір якого знаходиться у владі переслідувача.

Розв'язуюча функція характеризує хід гри. Коли в деякий момент часу інтеграл від неї перетворюється в одиницю, це означає попадання траєкторії на термінальну множину. Надаються достатні умови розв'язуваності задачі зближення з термінальною множиною. При цьому процес переслідування розбивається на два етапи.

На першому із них $[0, t_*)$, де t_* - момент перемикання, працює власне метод розв'язуючих функцій з використанням переслідувачем в кожен момент часу t всієї передісторії керування втікача $v_t(\cdot)$ (квазістратегії). Коли в момент t_* інтеграл від розв'язуючої функції стає рівним одиниці, процес переслідування перемикається на перший прямий метод Л.С.Понтрягіна і реалізується в класі контркерувань. Грубо кажучи, від моменту перемикання до розрахованого моменту закінчення гри «тягнеться» час, причому, на цій ділянці розв'язуюча функція вважається рівною нулю, так як накопичувати її більше немає сенсу.

Аппарат розв'язуючих функцій, які є, як правило, великими позитивними коренями квадратних рівнянь, виявився надзвичайно

зручним і універсальним при вирішенні конкретних прикладів.

3. Аналіз останніх досліджень та публікацій

В теорії конфліктно-керованих процесів (диференціальні ігри) разом з зворотніми процедурами Понтрягіна-Пшеничного [2, 3], правилом екстремального прицілювання Красовського [4] та ідеологією Айзекса [1], яка стосується основного рівняння теорії диференціальних ігор, існують ефективні методи, які можна виділити в окремий напрямок.

Це перший метод Понтрягіна та метод розв'язуючих функцій [5]. Їх об’єднує загальний принцип побудов керувань першого гравця на основі теореми вимірного вибору Філіппова-Кастена [6]. І хоча у авторському варіанті [2] в доведені використовувалась техніка ε-стратегій, П.Б.Гусятников та М.С. Нікольський використовували для доведення першого прямого методу принцип вимірного вибору. Метод розв'язуючих функцій виник внаслідок розв'язку ігрових задач втечі від групи переслідувачів.

В одній із робіт Б.Н.Пшеничного [3] на цій основі були сформульовані необхідні та достатні умови розв'язуваності задачі групового переслідування об'єктів з простим рухом і рівними максимальними швидкостями в оточенні. Це дало поштовх розвитку методів вирішення задач групового переслідування [5] на основі методики, яка використовує ідеологію розв'язуючих функцій.

Більшість ігрових задач з участю груп об'єктів, якими керуються, та динамікою, яка укладається в узагальнений контрольний приклад Понтрягіна, міститься в роботі [5]. При цьому часто динаміка об'єктів припускається однаковою, інакше розв'язок дуже ускладнюється.

З першим прямим методом Понтрягіна тісно пов’язаний метод розв’язуючих функцій [5].

Метод розв’язуючих функцій заснований на використанні обернених функціоналів Мінковського. Він дає пояснення класичного правила паралельного зближення, давно відомого інженерам-проектувальникам авіаційної та ракетної техніки на евристичному рівні.

Цей метод з успіхом застосовувався при різних формах умови Понтрягіна для дослідження ігрових задач з групами учасників, з термінальним функціоналом, з фазовими обмеженнями та неповною інформацією, а також для вивчення динамічних систем, які описуються рівняннями дробового порядку, та інших систем з більш складною динамікою, ніж звичайні диференціальні рівняння.

Для дослідження конфліктно-керованих процесів застосовується метод розв’язуючих функцій. Суть методу полягає в побудові по відомих параметрах процесу деяких числових функцій, що характеризують інтегрально хід конфліктно-керованого процесу, тобто ступінь близькості траєкторії до термінальної множини, і грають ключову роль при розв’язанні конкретних задач.

Оскільки розв’язуючі функції є опорними до визначальних багатозначних відображень, то апарат опуклого аналізу дозволяє їх будувати в аналітичному вигляді для досить широкого класу конфліктно-керованих процесів.

Привабливою стороною методу розв'язуючих функцій є той факт, що він дає повне обґрунтування класичного правила паралельного зближення, а також дозволяє ефективно використовувати сучасну техніку багатозначних відображень та їх селекторів в обґрунтуваннях ігрових конструкцій та отриманні на їх основі змістовних результатів.

При вирішенні конкретних ігрових задач на основі розв'язуючих функцій, важливої ролі набувають питання строгого обґрунтування методики. Тоді на передній план виходять проблеми, пов'язані з властивостями спеціальних багатозначних відображень, їх селекторів, які грають ключову роль при доведенні тверджень.

Отримані результати базуються на використанні формули Коші, що не завжди є зручним, та не охоплюють весь клас функціонально-диференціальних систем.

Головною метою цієї роботи є модифікація методу розв’язуючих функцій шляхом його узагальнення для складних динамічних, функціонально-диференціальних систем.

4. Виклад основного матеріалу

Розглянемо конфліктно-керований процес, еволюція якого описується рівністю:

$$z(t) = g(t) + \int_0^t \Omega(t,\tau)\varphi(u(\tau),v(\tau))d\tau, t \geq 0. \quad (1)$$

Тут $z(t) \in R^n$, функція $g(t), g: R_+ \to R^n, R_+ = \{t : t \geq 0\}$, вимірна по Лебегу і обмежена при $t > 0$, матрична функція $\Omega(t,\tau), t \geq \tau \geq 0$, вимірна по t шумується по τ для кожного $t \in R_+$. Блок керування задається функцією $\varphi(u,v), \varphi: U \times V \to R^n$, яка вважається неперервною за сукупністю змінних на прямому добутку непорожніх компактів U та V, тобто $U \in K(R^m), V \in K(R^l), m, l, n$ - натуральні числа.

Керування гравцями $u(\tau), u: R_+ \to U$, і $v(\tau), v: R_+ \to V$, - вимірні функції часу.

Крім процесу (1), задано термінальну множину M^*, яка має циліндричний вигляд:

$$M^* = M_0 + M, \quad (2)$$

де M_0 - лінійний підпростір із R^n, а $M \in K(L)$, де L – ортогональне доповнення до M_0 в R^n.

Цілі першого *(u)* та другого *(v)* гравців протилежні. Перший намагається вивести траєкторію процесу (1) на термінальну множину за найкоротший час, а інший – максимально відтягнути момент потрапляння траєкторії на множину M^* або взагалі уникнути зустрічі.

Представлення розв'язку динамічної системи вигляду (1) дозволяє в єдиній схемі розглянути широке коло функціонально-диференціальних систем, які функціонують в умовах конфлікту, зокрема, систем інтегральних, інтегро-диференціальних, диференціально-різницевих, а також систем рівнянь з класичними дробовими похідними Рімана-Ліувілля, регуляризованими дробовими похідними Капуто. Аналогічне представлення в дискретній ситуації дає можливість досліджувати багатокрокові процеси та імпульсні системи.

Приймемо сторону першого гравця і будемо орієнтуватися на вибір противником в якості керування довільної вимірної функції, яка приймає значення із *V*. Вважаємо, якщо гра (1), (2) відбувається на інтервалі $[0,T]$, то керування першого гравця в момент t вибираємо на основі інформації про $g(T)$ та $v_t(\cdot)$, тобто у вигляді вимірної функції:

$$u(t) = u(g(T), v_t(\cdot)), t \in [0,T], u(t) \in U, \quad (3)$$

де $v_t(\cdot) = \{v(s) : s \in [0,t]\}$ - передісторія керування дорогого гравця до моменту *t*, або у вигляді контркерування:

$$u(t) = u(g(T), v(t)), t \in [0,T], u(t) \in U. \quad (4)$$

Обговорювати окремо питання фізичного виконання при керуванні вигляду (3) тут не є доцільним. Відповідь на нього буде автоматично випливати з конструктивного способу побудови керування першого гравця в подальшому при доведенні тверджень.

Зокрема, якщо $g(t) = e^{At} z_0, \Omega(t,\tau) = e^{A(t-\tau)}, z(0) = z_0$, а e^{At} - матрична експонента, то формула (1) є відомою формулою Коші для квазілінійного конфліктно-керованого процесу. Керування $u(t) = u(z_0, v_t(\cdot))$ реалізує квазістратегію, а контркерування $u(t) = u(z_0, v(t))$ - передписано стробаскопічною стратегією Хайека.

Проміжки часу $[0, t_*)$ та $[t_*, T]$ називаються «активним» та «пасивним» відповідно. Опишемо спосіб керування першим гравцем на кожному з них. Для цього розглянемо компактнозначне відображення:

$$U(\tau, v) = \{u \in U : \pi\Omega(T,\tau)\varphi(u,v) - \gamma(T,\tau) \in \in \alpha(T,\tau,v)[M - \xi(T, g(T), \gamma(T,\cdot))]\}. \quad (5)$$

В силу теореми про обернений образ воно $L \times B$-вимірне, а значить, відповідно до теореми вимірного вибору в багатозначному відображенні $U(\tau, v)$ існує хоча б один $L \times B$-вимірний селектор $u(\tau, v)$, який є суперпозиційно вимірною функцією. Позначимо $u(\tau) = u(\tau, v(\tau))$. Керування першого гравця на «активному» проміжку покладемо рівним $u(\tau)$.

Розглянемо «пасивний» проміжок часу $[t_*, T]$. Покладемо у виразі (5) при $\tau \in [t_*, T]$, $v \in V$, розв'язуючу функцію $\alpha(T,\tau,v) \equiv 0$. Отримаємо багатозначне відображення:

$$U_0(\tau, v) = \{u \in U : \pi\Omega(T,\tau)\varphi(u,v) - \gamma(T,\tau) = 0\}.$$

Як і в попередньому випадку, із теореми про вимірний селектор випливає, що в $L \times B$-вимірному замкнутому відображенні $U_0(\tau, v)$ існує $L \times B$-вимірний селектор. Позначимо його $u_0(\tau, v)$ і керування переслідувача на «пасивному» проміжку виберемо рівним $u_0(\tau) = u_0(\tau, v(\tau))$. У випадку $\xi(T, g(T), \gamma(T, \cdot)) \in M$ керування першого гравця виберемо на всьому проміжку $[0, T]$ у вигляді $u_0(\tau) = u_0(\tau, v(\tau))$, де $u_0(\tau, v)$ - вимірний селектор багатозначного відображення $U_0(\tau, v)$.

5. Висновки і пропозиції

Отже, виконано модифікацію методу розв'язуючих функцій, яка полягає узагальненні опису еволюції конфліктно-керованого процесу, що дозволяє поширити використання методу на більш широкий клас функціонально-диференціальних систем, які функціонують в умовах гри або конфлікту. Визначено, що при виборі керувань переслідувача на «активному» і «пасивному» проміжках по вказаних правилах траєкторія конфліктно-керованого процесу буде приведена на термінальну множину в момент T при будь-яких допустимих керуваннях другого гравця.

Подальші дослідження полягають в знаходженні необхідних та достатніх умов закінчення диференціальної гри.

Список літератури:

1. Isaacs R. Differential games. – New York: J. Wiley and Sons, 1965. – 480 p.
2. Понтрягин Л.С. Избранные научные труды, Т.2. – М.: Наука, 1988. – 576 с.
3. Пшеничный Б.Н. Остапенко В.В., Дифференциальные игры. – К.: Наукова думка, 1992. – 260 с.
4. Красовский Н.Н., Игровые задачи о встрече движений. – М.: Наука, 1970. – 420 с.
5. Chikrii A. Conflict-controlled processes. – Boston, London, Dordrecht: Kluwer Acad. Publ., 1997. – 424 p.
6. Филиппов А.Ф. Дифференциальные уравнения с разрывной правой частью. – М.: Наука, 1985. – 224 с.

УДК 004.891.3

ГОЦ О. П.
СЕЛІН Ю. М.

ІНТЕЛЕКТУАЛЬНА ДІАГНОСТИЧНА МЕДИЧНА СИСТЕМА З ВИКОРИСТАННЯМ БАЙЄСОВИХ МЕРЕЖ ТОЧНОГО ВИСНОВКУ

В даній статті розглянуте практичне використання методу кластеризації для побудови точного ймовірнісного висновку в Байєсових мережах (БМ) для встановлення діагнозу пацієнта на основі статистичних показників, що характеризують реальний медичний стан пацієнта. Наведено порівняння існуючих апроксимаційних та точних методів побудови ймовірнісного висновку в БМ. На прикладі невеликої моделі показана ефективність використання алгоритму кластеризаці для побудови точного ймовірнісного висновку.

The subject of this article is the application of the clustering method for constructing exact probabilistic inference in Bayesian networks in conclusion for diagnosis of the patient based on symptoms characterizing the actual medical state of the patient. The comparison of existing exact and approximate methods for constructing probabilistic inference in Bayesian networks is provided. Efficiency of clustering algorithm for constructing exact probabilistic conclusion is provided.

1. Вступ

Розробка діагностичного медичного забезпечення є актуальною сьогодні, адже сучасне життя стає небезпечним через появу нових вірусів та хвороб, через недбалість лікарів та неправильні діагнози. Тому розробка інтелектуальної медичної системи конче необхідна.

У роботі запропоновано підходи до створення інтелектуальної інформаційної системи діагнозу хвороб на основі БМ з побудовою точного ймовірнісного висновку.

Система відрізняється тим, що дозволяє швидко адаптувати методи діагностики до появи нових показників, переналаштувати систему після нових симптомів і хвороб.

Розроблено методику дослідження статистичних показників, що характеризують реальний медичний стан пацієнтів за допомогою точного ймовірнісного висновку в БМ.

2. БМ як інструмент інтелектуального аналізу даних

БМ ефективні в інформаційних системах обробки кількісних даних, представлених часовими рядами і часовими перерізами, а також якісними даними, представленими експертними оцінками, лінгвістичними змінними, інтервальними значеннями і т. д.

Використовуються БМ зазвичай в системах класифікації даних різної природи, системах прогнозування, системах автоматичного розпізнавання мовних сигналів, маркетингу і бізнесі.

МБ можна представити у вигляді направленого ациклічного графу, вершинами якого є набір таблиць умовних ймовірностей (ТУЙ).

Змінні, що використовуються в МБ, можуть бути як дискретними, так і неперервними, а характер їх надходження при аналізі та прийнятті рішення може бути і в режимі реального часу, і у вигляді статистичних масивів інформації та баз даних.

Завдяки представленню взаємодії між факторами процесу у вигляді причино-наслідкових зв’язків в мережі досягаються максимально високий рівень візуалізації та, як наслідок, чітке розуміння суті взаємодії факторів процесу між собою. Саме це відрізняє МБ від інших методів інтелектуального аналізу даних (ІАД) [1].

Формула Байєса дозволяє «переставити причину і наслідок»: за відомим фактом події обчислити вірогідність того, що вона була викликана даною причиною.

Нехай подія A може відбутись тільки разом з однією із попарно несумісних подій $H_1,H_2,..,H_n$, які називаються гіпотезами і утворюють групу:

$$\sum_{i=1}^{n} P(H_i) = 1$$

Тоді, якщо відбулась подія A, то це означає, що відбулась одна із попарно несумісних подій $AH_1, AH_2, .., AH_n$.

Це означає: $A = H_1 * A + H_2 * A + .. H_n * A$.

Використавши теорему додавання, одержимо:

$$P(A) = P(H_1 * A + H_2 * A + .. + H_n * A) = P(H_1 * A) + P(H_2 * A) + .. + P(H_n * A)$$

З теореми множення ймовірностей:

$$P(H_i) = P(H_i) * P_{H_i}(A),$$

де $i = 1, 2, 3, ..., n$.

$$P(A) = P(H_1) * P_{H_1}(A) + P(H_2) * P_{H_2}(A) + .. + P(H_n) * P_{H_n}(A). \quad (1.1)$$

Одержана формула (1.1) називається *формулою повної імовірності.*

Після цього нас цікавить питання про те, як зміняться ймовірності гіпотез H_i, де i=1, 2…, n, якщо подія A відбулась. Тобто, як обчислити $P_A(H_i)$. Справедливі рівності:

$$P(H_i A) = P(A) * P_A(H_i) = P(H_i) * P_{H_i}(A),$$ звідки

$$P_{H_i}(A) = \frac{P(H_i)}{P(A)} \quad (1.2)$$

Ця формула (1.2) називається *формулою Байєса* [2].

3. Ймовірнісний висновок в БМ

Існує два основних типи знаходження висновків в мережах Байєса:

- ймовірнісний висновок (probabilistic inference або belief updating)
- максимальне апостеріорне пояснення (Maximum a Posteriori - MAP explanation або belief revision).

Метою ймовірнісного висновку є знаходження $P(X|E)$ - апостеріорної ймовірності шуканих вершин X , при деякому значенні спостережуваних вершин E.

За розміром вирішуваних задач можна виділити два класи ймовірнісного висновку: точний та апроксимаційний. При вирішення реальних життєвих великих задач застосування точного ймовірнісного висновку стає неможливим через велику обчислювальну складність, і саме тоді застосовуються апроксимаційні методи, які виконують обчислення наближено.

Проте, коли задача структурована та важлива точність імовірнісного висновку, алгоритми точного ймовірнісного висновку будуть доцільними. Важливим етапом при використанні алгоритмів точного висновку є правильна побудова моделі, при великій кількості вузлів, таку модель потрібно сегментувати чи розбити на кілька під моделей, які працюватимуть незалежно одна від одної. Лише за таких умов задача вирішуватиметься ефективно.

Основоположний алгоритм побудови точного ймовірнісного висновку в мережах Байєса – алгоритм передачі повідомлення між вузлами мережі (алгоритм Перла). З часом з'явилися алгоритми, побудовані на основі ідеї алгоритму Перла, які є більш ефективними і можуть бути використаними для обчислення точного ймовірнісного висновку у значно складніших системах. До них належать:

- алгоритм cutset condition (визначеного перетину);
- алгоритм variable elimination (виключення змінних);
- алгоритм bucket elimination (поглинаючого виключення);
- алгоритм clusterization (кластеризації).

До найбільш ефективних алгоритмів апроксимаційного висновку належать:

- алгоритм stochastic sampling (стохастичної вибірки);
- алгоритм model simplification (спрощення моделі);
- алгоритм search-based (пошукові);
- варіаційні алгоритми.

4. Алгоритм кластеризації

Джуді Перл був першим, хто запропонував побудову точного ймовірнісного висновку у мережах Байєса, що базується на основі ідеї обміну повідомленнями між вершинами-батьками і вершинами-нащадками у направлених ациклічних графах для обчислення значення їх ймовірностей. Ним був розроблений алгоритм передачі повідомлень між вершинами в МБ.

Ключовою особливістю є те, що мережа повинна бути однозв'язною, тобто представлятися у вигляді ациклічного направленого графу, у якому між двома будь-якими вершинами існує лише один

шлях. Однозв'язні мережі також називають полідеревами.

З часом було запропоновано алгоритм кластеризації (LS – алгоритм, від винахідників Lauritzen та Spiegelhalter)

Алгоритм оперує об'єднаними деревами, кожна вершина якого містить деякий набір змінних і ТУЙ, це дозволяє використовувати ідею обміну повідомленнями ймовірнісного висновку на основі ідеї Перла.

Алгоритм побудови об'єднаного дерева представлений блок-схемою на рис. 1.

Рис. 1. Алгоритм побудови об'єднаного дерева в МБ

Об'єднане дерево формується з доменного графа МБ - так називається граф, вершинами якого є вузли МБ, а ребрами з'єднуються ті вершини, які в мережі були залежні один від одного, тобто імовірності наслідків однієї вершинами залежать від результатів інших(ої) вершин (и). Доменний граф не містить в собі таблиць умовних ймовірностей (ТУЙ) БМ, тому несе в собі інформацію лише про якісні, а не кількісні характеристики залежностей змінних в мережі.

На етапах моралізації і триангуляції в доменний граф додаються додаткові ребра, необхідні для подальшого перетворення в деревовидний граф.

Моралізованим (moral) називається доменний граф, в якому проведені додаткові шляхи (ребра) між кожними вершинами A і B, для яких у БМ знайдеться вершина C, залежна і від A, і від B.

Граф називається триангульованим (triangulated), якщо в ньому відсутні цикли з чотирьох і більше вершин. Цикл в даному випадку визначається як множина вершин, в якій кожна вершина з'єднана рівно з двома іншими вершинами цієї множини.

Після цього будується «заготовка» для об'єднаного дерева – дерево, що представляє собою деревовидний граф, вузлами якого є об'єднання вершин доменного графа. Далі в дерево об'єднань включаються таблиці умовних ймовірностей з розглянутої БМ - цим завершається процес побудови об'єднаного дерева.

Для реалізації алгоритму кластеризації в БМ зазвичай використовується архітектура *Hugin*.

У архітектурі Hugin маємо об'єднане дерево і відповідні таблиці для кожної кліки (така підмножина вершин, що кожні дві вершини з цієї підмножини поєднанні ребром). Сепаратор кожного ребра дерева буде також містити відповідні таблиці.

Процес пропагації (передача повідомлень) також відбувається в два етапи – сходження догори та донизу. В архітектурі Hugin на етапі сходження догори відправник не ділить свою таблицю на повідомлення, а замість цього записує його в сепаратор. Це економить підрахунки, але й потребує більшого об'єму памяті. На етапі сходження донизу сепаратор ділить нове повідомлення на те, яке він раніше зберігав і саме на це відношення множить свою таблицю отримувач повідомлення. Економія обчислень відбувається завдяки діленню таблиць сепараторів, що мають менший розмір, ніж таблиці клік.

5. Ефективність алгоритму

Алгоритм кластеризації для побудови точного ймовірнісного висновку в БМ гарантує точність обчислень, нехтуючи обчислювальною складністю. Він є одним із найефективніших в своєму класі.

В свою чергу апроксимаційні алгоритми більш прості в обрахунках, проте результат має певну похибку, в залежності від типу алгоритму. Існують такі апроксимаційні алгоритми, які взагалі не

гарантують точність, що заперечує їхньому використанню у медичній галузі.

Використовуючи Java-бібліотеку Jayes [3], було проведено ряд експериментів для порівняння обчислювальної складності алгоритму кластеризації для побудови точного ймовірнісного висновку в БМ і алгоритму стохастичної вибірки для апроксимаційного.

Побудувавши деяку модель БМ, для кожного алгоритму було проведено по 10 випробовувань, при різній кількості вершин графу БМ для знаходження часу навчання системи. На рис. 2. приведені результати експерименту.

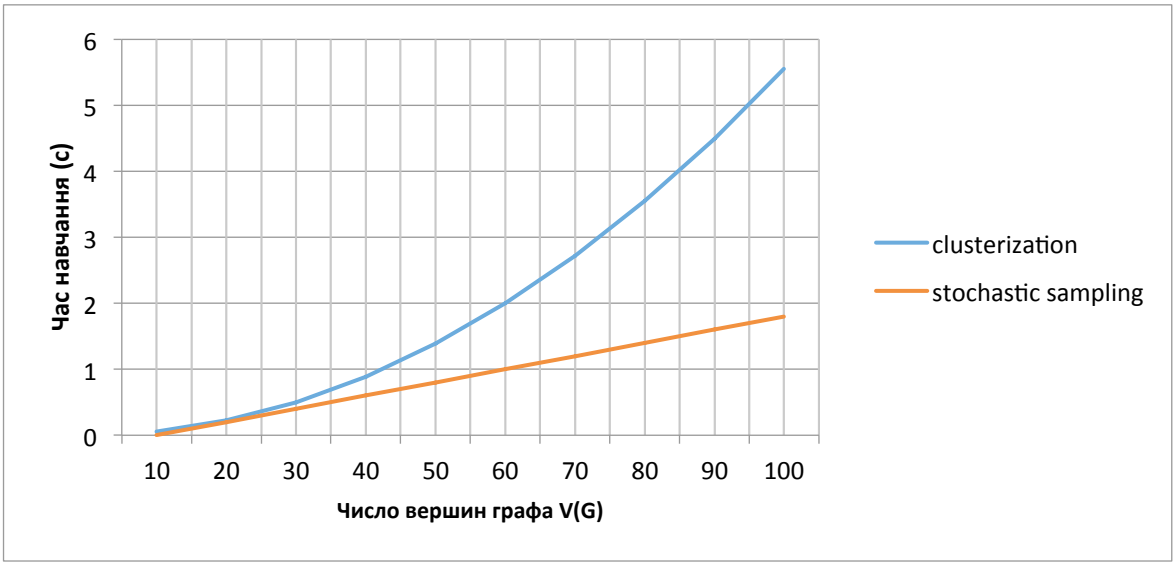


Рис. 2. Графік залежності к-сті вершин графу від часу навчання системи для алгоритмів стохастичної вибірки і кластеризації

Як видно з графіку, алгоритм кластеризації має більшу обчислювальну складність, про що свідчить експоненціальне зростання часу при навчанні, після збільшення кількості вершин графу БМ. В свою чергу, складність апроксимаційного алгоритму має лінійну залежність від кількості вершин графу.

Проте, при кількості вершин до 30, час навчання для обох алгоритмів майже однаковий.

Для моделювання структури медичної інтелектуальної системи БМ можна розбити на підмережі, графи якої матимуть біля відносно невелику кількість вершин. Відповідно до МКХ-10 (міжнародний класифікатор хвороб), можна виділити такі класи, для кожного з яких в подальшому побудувати БМ:

- Деякі інфекційні та паразитарні хвороби;
- Новоутворення;
- Хвороби крові і кровотворних органів та окремі порушення з залученням імунного механізму;
- Хвороби ендокринної системи, розладу харчування та порушення обміну речовин;
- Розлади психіки та поведінки;
- Хвороби нервової системи;
- Хвороби ока та придаткового апарату;
- Хвороби вуха та соскоподібного відростка;
- Хвороби системи кровообігу;
- Хвороби системи дихання;
- Хвороби органів травлення;
- Хвороби шкіри та підшкірної клітковини;
- Хвороби кістково-м'язової системи та сполучної тканини;
- Хвороби сечостатевої системи;
- Вагітність, пологи та післяпологовий період [4].

Таке розбиття допоможе зберегти відносно невелику обчислювальну складність для точного алгоритму

кластеризації, яка не сильно відрізнятиметься від апроксимаційних алгоритмів, при цьому зберігатиметься виграш у точності обчислень.

6. Висновки

БМ чудово підходять для аналізу процесів різної природи, та мають ряд переваг серед інших інтелектуальних методів аналізу даних та прогнозування.

Раціональне використання БМ, їх швидка і надійна робота в першу чергу залежать від моделі та алгоритму побудови висновку в мережі.

У даній роботі був описаний і проаналізований алгоритм кластеризації, який належить до класу точних алгоритмів. Цей алгоритм поєднує в собі три переваги: точність результатів, відносно малий час роботи і універсальність.

Експериментальним шляхом проаналізована різниця в обчислювальній складності алгоритму кластеризації з апроксимаційним алгоритмом. З чого зроблені висновки, що БМ для всієї медичної системи потрібно розбити на незалежні БМ, що дозволить зберегти точність та покращити ефективність обчислень.

Сьогодні медична статистика представлена достатньо повно і з необхідним рівнем достовірності, що дозволить побудувати якісну модель системи та використовувати в ній алгоритм кластеризації.

Список літератури

1. Бідюк П. І. Інтелектуальний аналіз слабоструктурованих даних за допомогою байєсових мереж: звіт по результатам виконання робіт за грантом грант НТУУ „КПІ” № 3/5-ГР, 2006-2007р. / П. І. Бідюк, О. М. Терентьєв, Л. О. Коршевнюк. – 2007. – 85 с.
2. Тичинська Л.М. Формула повної ймовірності. Формула БАйєса // Теорія ймовірностей. - 2010. – 112 с.
3. Michael Kutschke. An Introduction to Bayesian Networks with Jayes [Електронний ресурс]. – 2013. Режим доступу: http://www.codetrails.com/blog/introduction-bayesian-networks-jayes/ (останній візит: 30.03.17).
4. МКХ-10 [Електронний ресурс]. – 2016. Режим доступу: https://mkh10.com.ua/ (останній візит: 29.03.17).

УДК 004.93(015.7)

ФЕДИШИН М.В.,
СТОЛПАКОВ Д.Є.

ОДНОЧАСНА ПОБУДОВА 3D КАРТИ ТА ВИЗНАЧЕННЯ МІСЦЕЗНАХОДЖЕННЯ КАМЕРИ

Розглянена задача у галузі робототехніки з побудови 3D карти та знаходження свого місцяперебування. Запропонований універсальний та точний алгоритм для її реалізації.

We consider the computational problem of constructing a map of an unknown environment while simultaneously keeping track of an agent's location within it. Versatile and accurate implementation algorithm is proposed

.

1. Вступ

SLAM - це алгоритмічно обчислювальна задача побудови і оновлення карти невідомого оточення з одночасним відстежуванням місцезнаходження, рухаючись по ньому.

Розв'язок алгоритмів Bundle Adjustment (BA) забезпечує таку ж точну оцінку місцезнаходження камери, як і розріджені геометричні перетворення[1]. Протягом тривалого часу такий підхід вважався недозволеним для використання в режимі реального часу. Ціллю оптичного SLAM'у є оцінка траєкторії камери під час відновлення середовища. Сьогодні можна досягти точних результатів без надмірно великої обчислювальної складності.

Для обчислення BA нам потрібно:

1. мати відповідності спостережень ознак точок карти серед підмножини обраних ключових кадрів;
2. уникати непотрібних шумів при виборі ключових кадрів, оскільки з їх збільшенням обчилювальна складність зростає;
3. мати сильнозв'язну мережу поєднань ключових кадрів і точок, щоб забезпечити точні результати;
4. знати первісну оцінку місцезнаходження ключових кадрів і точок для нелінійної оптимізації;
5. досліджувати карту місцевості, коли оптимізація сфокусована на досягнення масштабованості;
6. мати можливість застосувати швидку глобальну оптимізацію для закриття циклу в режимі реального часу.

У нашій роботі ми використовуємо ідею алгоритму PTAM[2], розпізнання місця на основі роботи Galvez-Lopez and Tardos [3], масштабовану шкалу для закриття циклу Strasdat et. al [4] та інформацію спільної видимості для великомасштабних операцій[5], [6].

2. Огляд системи

Однією з головних ідей системи є використання однакових характеристик для відстеження, відображення та розпізнання місця для застосування релокалізації частоти кадрів та виявлення циклу. Саме це робить систему ефективною та не потребує інтерполяції глибини схожих на SLAM[4] ознак. Ми використовуємо лише ті ознаки, що зустрічаються частіше ніж кожні 33мс, виключаючи ознаки методів SIFT(300мс), SURF(300мс), A-KAZE(100мс).

Використання ORB ознак, що визначаються багатомасштабованими кутами FAST, пов'язаними 256 бітовим дескриптором, приводить до швидких обчислень та знаходжень збігів. Це дозволяє зіставити їх з великою кількість базисних ліній, підвищуючи точність алгоритму BA.

Система включає в себе три процеси, що виконуються паралельно:

1. відстеження (tracking);
2. локальне відображення (local mapping);
3. завершення циклу (loop closing).

Відстеження відповідає за місцезнаходження камери в кожному кадрі та вирішує, коли вставляти новий ключовий кадр. Спочатку ми знаходимо початкові збіги характеристик з попереднім кадром і оптимізуємо місцерозташування,

використовуючи алгоритм motion-only BA. Якщо відстеження втрачається (через оклюзії чи різкі рухи), для глобальної повторної локалізації використовується модуль розпізнання місця. Після того, як отриманого початкову оцінку місця розташування камери та збігів ознак, за допомогою графа спільної видимості (covisibility graph) ключових кадрів відновлюється локальна карта. Тоді за допомогою повторної проекції шукаються збіги з локальною картою точок та оптимізується місцезнаходження камери у відповідності з усіма збігами. Після цього процес відстеження вирішує, чи вставляти новий ключовий кадр.

Процес локального відображення обробляє новий ключовий кадр і для досягнення оптимального відновлення в межах позиції камери, застосовується алгоритм local BA. Для триангуляції нових точок в новому ключовому кадрі шукаються нові відповідності в пов'язаних ключових кадрах з covisibility graph'у. Через деякий час після створення графу, на основі інформації, зібраної під час відстеження, для того, щоб зберегти лише кращі точки, відбувається їх відбір. Надлишкові ключові кадри теж видаляються.

Процес закриття циклу шукає цикли з кожним новим ключовим кадром. Якщо цикл знайдений, обраховується схожість перетворення, що говорить про повільні відхилення в циклі. Тоді обидві сторони циклу вирівнюються, а повторювані точки видаляються. В кінці кінців, для досягнення глобальної узгодженості, граф місцярозташування оптимізується за подібністю.

На рисунку 1 зображено покрокове виконання 3-ох процесів системи.

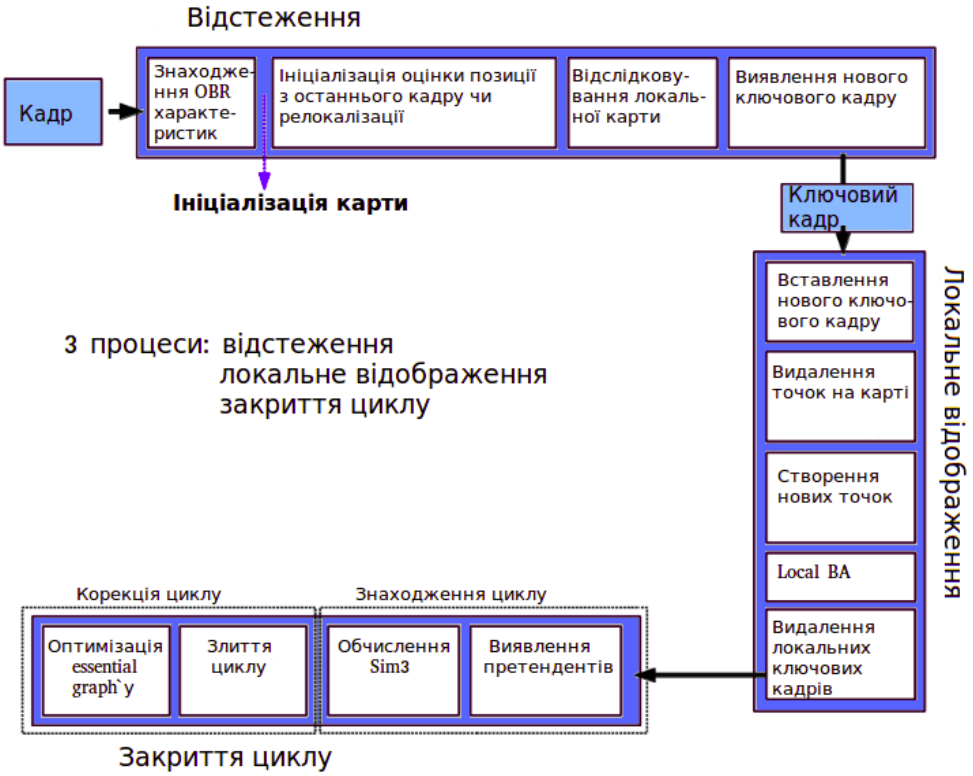


Рис. 1. Покрокове виконання 3-ох процесів системи

3. Точки карти, ключові кадри та їх вибір

Кожна точка карти $X_{w,i}$ надає:

1. своє 3D розташування в системі координат світу;
2. оглядовий напрям n_i, що задається як середній одиничний вектор всіх оглядових напрямів (промені, що з'єднують точку з оптичним центром ключового кадру);
3. характерний ORB дескриптор D_i, який є асоціатиним ORB дескриптором, гамільтонова відстань якого є мінімальною відносно всіх інших асоціативних дескрипторів ключових кадрів з розглянутою точкою;
4. максимальна d_{max} та мінімальна d_{min} відстані, в яких може бути розглянута точка, враховуючи межі масштабування ORB характеристик.

Кожен ключовий K_i кадр надає:

1. з точок навколишнього середовища перетворену до системи координат позицію камери T_{iw};
2. властивості камери, що включають фокусну відстань та основну(головну, початкову) точку;
3. всі вибрані в кадрі ORB характеристики.

Точка карти і ключові кадри відбираються певним механізмом, що виявляє надлишкові ключові кадри, неправильно вибрані чи точки карти, які не відстежуються. Це дозволяє карті розширюватись під час досліджень, що підвищує стійкість відстеження у важких умовах(повороти, швидкі рухи), а розміру бути обмеженим при постійних повторних відвідуваннях одного і того ж місця.

4. Експерименти

Ми провели експерименти над системою, використовуючи великий датасет NewCollege [39], оцінюючи загальну ефективність системи, точність локалізації, релокалізації. Всі експерименти з були проведені з використанням Intel Core i7-4700MQ (4 ядра, 2.40 ГГц) і 8 Гб оперативної пам'яті. Датасет NewCollege [39] містить послідовність відео кампусу та прилеглих парків, сумарно на 2.2 км. Відео записувалося з допомогою монокулярної камери при 20 кадрах в секунду і розширенням 512×382. Воно також містить декілька циклів і швидкі повороти, що робить послідовність досить складним завданням для монокулярного зору.

Ми знайшли статистику часу, що витрачається з кожним циклом в цьому експерименті. На рисунку 2 наведено результати для відстеження та локального відображення. Відстеження спрацьовує за умови частоти кадрів близько 25-30 Гц. При необхідності час можна зменшити, обмежуючи кількість ключових кадрів, які включено в локальну карту. В процесі локального відображення найбільш складним завданням є local BA. Під час дослідження території через вставлення нового ключового кадру алгоритм local BA може перерватись, саме це призводить до варіювання часу його виконання.

Відстеження та відображення (датасет NewCollege)

Процес	Операція	median (мс)	mean (мс)	std (мс)
Відстеження	Вибір ORB ознак	11.10	11.42	1.61
	оцінка позиції	3.38	3.45	0.99
	Track local map	14.84	16.01	9.98
	Сумарно	30.57	31.60	10.39
Локальне відображення	Вставка кадрів	10.29	11.88	5.03
	Видалення точок	0.10	3.18	6.70
	Створення точок	66.79	72.96	31.48
	Local BA	296.08	360.41	171.11
	Видалення кадрів	8.07	15.79	18.98
	Сумарно	383.59	464.27	217.89

Рис. 2. Статистика часу відстеження та відображення

На рисунку 3 наведені результати для кожного з 6 знайдених циклів закриття. Можна побачити, як збільшується кількість виявлення циклів з кількістю ключових кадрів. Це відбувається через ефективне виконання запитів до бази даних, що тільки порівнюють підмножину зображень зі словами, які показують здатність мішка слів у розпізнанні місця.

Статистика часу закриття циклу (датасет NewCollege)

			Визначення циклу (мс)		Корекція циклу (мс)		
Цикл	Ключові кадри	Створення ребер в essential graph'і	Визначення претендентів	Перетворення подібності	Злиття	Оптимізація essential graph'у	Сумарно (с)
1	287	1347	4.71	20.77	0.20	0.26	0.51
2	1082	5950	4.14	17.98	0.39	1.06	1.52
3	1279	7128	9.82	31.29	0.95	1.26	2.27
4	2648	12547	12.37	30.36	0.97	2.30	3.33
5	3150	16033	14.71	41.28	1.73	2.80	4.60
6	4496	21797	13.52	48.68	0.97	3.62	4.69

Рис. 3. Статистика часу закриття циклу

5. Висновок

У цій роботі ми розглянули монокулярну SLAM систему з детальним описом будівель та повною оцінкою на публічних даних. Система може обробляти послідовності зображень, зроблені в кімнатах чи на вулиці,

їдучи на автомобілі чи ідучи по вулиці. Точність системи, як правило, відрізняється від реальності менше ніж 1 см в кімнатних приміщеннях, та на декілька метрів на вулиці.

В даний час алгоритм PTAM [7] вважається найбільш точним методом SLAM з використанням монокулярного відео в режимі реального часу. Це не випадково, адже бекенд алгоритму PTAM оснований на алгоритмі BA, який є стандартом для автономної проблеми побудови структури від руху (Structure From Motion)[1]. Основним внеском роботи є розширення універсальності з PTAM в умовах, які є не розв'язуються цією системою. Процес видалення ключових кадрів дозволяє їх створювати кожні декілька кадрів і видаляти, коли вони зайві. Таке гнучке розширення карти є дуже важливим в погано обумовлених(наявні повні обертання чи швидкі рухи) траєкторіях, що досліджуються. При повторній роботі в тому ж середовищі, карта збільшується лише тоді, коли візуальний обсяг середовища змінюється, зберігаючи всю історію цих змін. Аналізуючи історію довготривалих досліджень, можна спостерігати цікаві результати.

ORB ознак достатньо для розпізнавання важкодоступних місць з ретельною зміною точки огляду. Крім того, вони дуже швидко вибираються і знаходять збіги (без необхідності мультипоточності чи використання GPU), які дозволяють точне відстеження і відображення в режимі реального часу.

6. Додаток. Нелінійна оптимізація

Bundle Adjustment (BA):

Точка карти 3D місцерозташування $X_{w,j} \in R^3$ та позиція ключового кадру $T_{iw} \in SE(3)$ оптимізуються мінімізуючи похибку повторної проекції на відповідні ключові точки $x_{i,j} \in R^2$. Умова похибки спостереження точки карти j в ключовому кадрі i:

$$\mathbf{e}_{i,j} = \mathbf{x}_{i,j} - \pi_i(\mathbf{T}_{iw}, \mathbf{X}_{w,j}) \qquad (5)$$

де π_i - функція проекції:

$$\pi_i(\mathbf{T}_{iw}, \mathbf{X}_{w,j}) = \begin{bmatrix} f_{i,u}\frac{x_{i,j}}{z_{i,j}} + c_{i,u} \\ f_{i,v}\frac{y_{i,j}}{z_{i,j}} + c_{i,v} \end{bmatrix} \qquad (6)$$

$$\begin{bmatrix} x_{i,j} & y_{i,j} & z_{i,j} \end{bmatrix}^T = \mathbf{R}_{iw}\mathbf{X}_{w,j} + \mathbf{t}_{iw}$$

де $R_{iw} \in SO(3)$ і $t_{iw} \in R^3$ є обертанням і переміщенням T_{iw} та $(f_{i,u}, f_{i,v})$ відповідно та $(c_{i,u}, c_{i,v})$ є фокальною відстанню і основною точкою камери i. Функція вартості, яку потрібно мінімізувати:

$$C = \sum_{i,j} \rho_h(\mathbf{e}_{i,j}^T \mathbf{\Omega}_{i,j}^{-1} \mathbf{e}_{i,j}) \qquad (7)$$

де p_h є надійною функцією вартості, а $\Omega_{i,j} = \sigma^2_{ij} I_{2x2}$ - матриця коваріації, пов'язана з масштабом, при якому була виявлена ключова точка. В випадку full BA, оптимізуються всі точки і ключові кадри, за винятком першого ключового кадру, який залишається фіксованим в якості початку координат. В local BA оптимізуються всі точки, які включені в локальну область, в той час як підмножина ключових кадрів є фіксованою. В оптимізації позиції чи motion-only BA всі точки фіксовані, а оптимізується лише позиція камери.

Список літератури

1. B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, “Bundle adjustment a modern synthesis,” in Vision algorithms: theory and practice, 2000, pp. 298–372.
2. http://www.robots.ox.ac.uk/~gk/PTAM/
3. D. Galvez-Lopez and J. D. Tardos, “Bags of binary words for fast place recognition in image sequences,” IEEE Transactions on Robotics, vol. 28, no. 5, pp. 1188–1197, 2012.
4. H. Strasdat, J. M. M. Montiel, and A. J. Davison, “Scale drift-awarelarge scale monocular SLAM
5. H. Strasdat, A. J. Davison, J. M. M. Montiel, and K. Konolige, “Double window optimisation for constant time visual SLAM,” in IEEE International Conference on Computer Vision (ICCV), Barcelona, Spain,November 2011, pp. 2352–2359.
6. C. Mei, G. Sibley, and P. Newman, “Closing loops without places,” in IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS), Taipei, Taiwan, October 2010, pp. 3738–3744.
7. G. Klein and D. Murray, “Parallel tracking and mapping for small AR workspaces,” in IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR), Nara, Japan, November 2007, pp. 225–234.

РУДЯКОВ Ю.И.,
ТОМАШЕВСКИЙ В.Н.

ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ РОЕВОГО ИНТЕЛЛЕКТА В МУЛЬТИАГЕНТНОЙ СИСТЕМЕ

Предлагается подход, который может быть использован для адаптации имитационных моделей мультиагентных систем за счет использования методов роевого интеллекта и нечеткой логики для описания коллективного поведения агентов децентрализованной системы. Практическое применение этого подхода демонстрируется на примере имитационной модели распределенной энергосистемы, а также использования роевого интеллекта для некоторые других систем.

An approach is proposed that can be used to adapt the simulation models of multi-agent systems through the use of methods of roving intelligence and fuzzy logic to describe the collective behavior of agents of a decentralized system. The practical application of this approach is demonstrated by the example of a simulation model of a distributed power system, as well as the use of roving intelligence for some other systems.

1. Введение

Предлагается подход, который окажет поддержку в адаптации некоторых имитационных моделей к показателям реального мира и усилит их стабильность за счет внедрения роевого интеллекта. Этот подход включает использование методов нечеткой логики и роевого интеллекта для описания коллективного поведения агентов децентрализованной системы. Для имитации поведения агентов активно используется событийно-ориентированное програм-мирование, а для описания реакций агентов на события применяется нечеткая логика.

2. Имитационное моделирование нечетких систем

Имитационное моделирование используется во всех сферах деятельности человека, начиная от моделей технических, технологических и организационных систем, и заканчивая проблемами развития человечества [1]. В обзоре [2] показано, что имитационное моделирование является одним из самых распространенных методов, используемых на практике.

Основная задача нечеткого моделирования заключается в нахождении конечного множества локальных отношений вход-выход, которые описывают систему или процесс в виде нечетких «ЕСЛИ-ТО» правил. Обучение нечетких систем включает два основных этапа: определение структуры и оценка параметров. На первом этапе определяются такие характеристики нечеткой системы, как число нечетких правил, количество лингвистических термов, на которые разбиты входные и выходные переменные. Этот этап может быть выполнен с использованием субъективного распределения данных, алгоритма диффузии или нечеткого кластерного анализа [3].

3. Нечеткие множества первого типа

Нечеткое множество первого типа (НМ1) – множество, в котором функция принадлежности элемента множества может принимать любые значения в интервале [0, 1], а не только значения 0 или 1, как описано в работе [4]. Это свидетельствует, что элемент входит в нечеткое множество с некоторой уверенностью.

В общем виде НМ1 можно определить следующим образом:

$$\widetilde{A} = (B, f),$$

$$f \to B{:}X;\ f \in [0,1], X = \emptyset,$$

где B – базис, X – универсальное множество, f – отображение базиса на универсальном множестве.

4. Нечеткие множества второго типа

Нечеткие множества второго типа (НМ2) являются обобщением нечетких множеств первого типа и используется для обработки большей степени неопределенности.

Они дают возможность описать всю неопределенность в функции принадлежности теории нечетких множеств. Если же неопределенность имеет достаточно низкий уровень, то НМ2 можно свести к НМ1.

Исходя из приведенного выше определения НМ1 можно вывести формулу для определения нечеткого множества *n*-порядку

$$\widetilde{A_n} = \left\{ f(x) \middle| \forall x \in B, f(x) = \widetilde{A_{n-1}} \right\}.$$

При описании объектов и явлений с помощью нечетких множеств используется понятие нечетких и лингвистических переменных.

Нечеткая переменная характеризуется тройкой <α, *X*, *A*>, где α – имя переменной, *X* – универсальное множество (область определения α), *A* – нечеткое подмножество из *X*, описывающее ограничения на значения нечеткой переменной α.

Лингвистической переменной называется набор <β, *T*, *X*, *G*, *M*>, где β – имя лингвистической переменной; *T* – множество его значений (терм-множество), которые представляют имена нечетких переменных, областью определения которых является множество *X*.

Множество *T* называется базовым терм-множеством лингвистической переменной; *G* – синтаксическая процедура, которая позволяет оперировать элементами терм-множества *T*, в частности, генерировать новые термы (значения).

Множество $T \cup G(T)$, где $G(T)$ – множество сгенерированных термов, которое называется расширенным терм-множеством лингвистической переменной; *M* – семантическая процедура, которая позволяет преобразовать новое значение лингвистической переменной, образованной процедурой *G*, в нечеткую переменную, то есть сформировать ответное нечеткое множество.

Во избежание большого количества символов:

- символ β используют как для названия самой переменной, так и для всех ее значений;
- для обозначения нечеткого множества и его названия пользуются одним символом, например, терм «молодой», является значением лингвистической переменной β = «возраст», и, одновременно, нечетким множеством *M* («молодой»).

Нечеткий вывод формируется при прохождении следующих этапов:

- фаззификация (англ. fuzzy – нечеткий) преобразует четкие величины, измеренные на выходе объекта управления, в нечеткие величины, которые описаны лингвистическими переменными в базе знаний;
- формирование базы знаний – описание лингвистических переменных и нечетких правил;
- формирование блока решений – используются нечеткие условные (if – then) правила, заложенные в базу знаний, для преобразования нечетких входных данных в необходимые управляющие воздействия, которые также носят нечеткий характер;
- дефаззификация – преобразования нечетких данных с выхода блока решений в четкую величину, которая используется для управления объектом.

5. Управление распределенными энергосистемами

В качестве системы, на примере которой можно разработать и применить рассматриваемый подход, выбрана система управления распределенными энергосистемами. На фоне растущего и меняющегося спроса на электрическую энергию обостряется проблема ее оптимального распределения между потребителями. Наряду с традиционными ископаемыми-энергоносителями все чаще используются возобновляемые источники. Технологии их применения становятся все более выгодными и

удобными. Однако большинство таких источников не позволяют производить необходимый объем энергии постоянно. Эффективность их зависит от сезона, времени суток, текущих погодных и природных условий. Чтобы в режиме реального времени сбалансировать спрос и предложение энергии, произведенной с использованием возобновляемых и не возобновляемых ресурсов, необходимы системы интеллектуального управления.

Методы роевого интеллекта позволяют оптимизировать распределение энергии: связать объекты энергосетей, использующих различные центры производства энергии (солнечные панели, ветровые установки, теплоэлектроцентрали и др.), с одной стороны, и центры потребления (здания, предприятия, электромобили и др.) – с другой. Новые интеллектуальные решения способны рассчитывать оптимальные способы и каналы передачи энергии между ее поставщиками и потребителями, прогнозировать спрос и предложение с учетом накопленных статистических данных.

В системе управления распределенными энергосистемами данный подход позволит улучшить следующие показатели:

- повысить безопасность и стойкость инфраструктуры электросетей (если агент системы выходит из строя, остальные продолжают действовать и подстраиваться под ситуацию);
- уменьшить затраты на энергоносители;
- эффективнее распределять и использовать энергию, снизить выбросы парниковых газов.

6. Имитационное моделирование распределенной энергосистемы

При имитационном моделировании роевого интеллекта в распределенной энергосистеме агенты делятся на несколько типов:

- производитель электроэнергии;
- потребитель электроэнергии;
- подстанция передачи электроэнергии.

Производитель (электростанция) производит электроэнергию и имеет такие свойства, как текущая и максимальная мощность, функцию выработки электроэнергии. Потребитель потребляет электроэнергию и имеет такие свойства, как текущий и максимальный показатель потребления, показатель перегрузки, функцию потребления электроэнергии. Подстанция помогает балансировать и настраивать передачу электроэнергии.

Описанные агенты системы соединяются в сеть с помощью линий электропередач, у которых имеются такие свойства как пропускная способность и коэффициент потерь при передаче. Пример такой системы приведен на рис. 1. Квадратами изображены две солнечные электростанции. Они соединены с подстанциями (желтые кружки), которые в свою очередь соединены с потребителями (красные кружки). Сбоку у электростанций и у потребителей изображаются их показатели мощности (вырабатываемой и потребляемой соответственно).

В системном времени каждый день обновляются функции выработки и потребления соответствующих агентов системы, с небольшими случайными отклонениями относительно задаваемых стандартов (пример показан на рис. 2).

Цель агентов-производителей состоит в стабильной поставке электроэнергии: ни больше, ни меньше требуемого потребителями количества. Агент может прогнозировать спрос и предложение с учетом накапливаемых в себе и в других агентах статистических данных, а также реагировать на события и последствия влияния его окружения, применяя нечеткую логику в поведении.

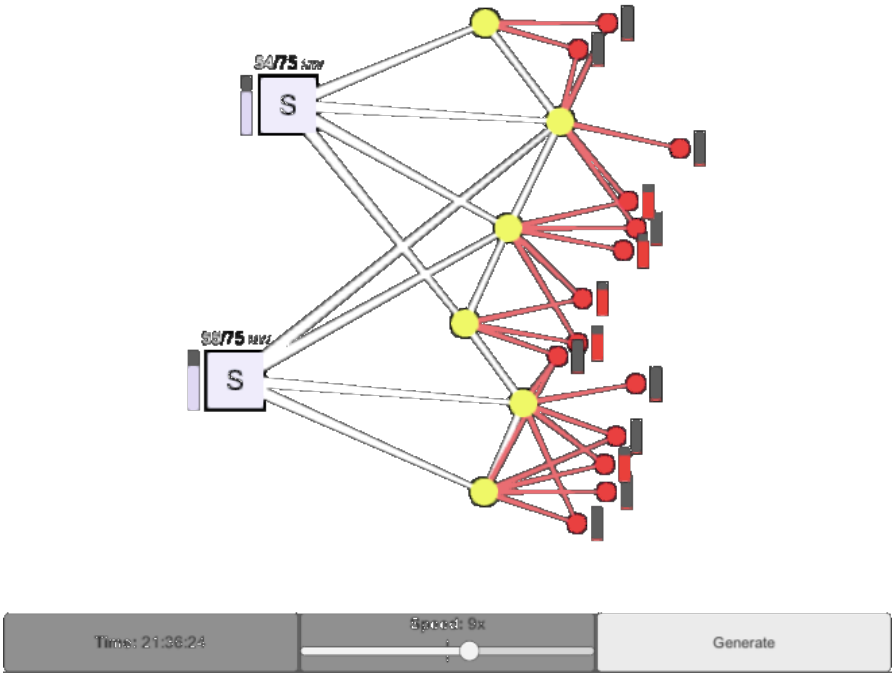


Рис. 1. Имитационная модель распределенной энергосистемы.

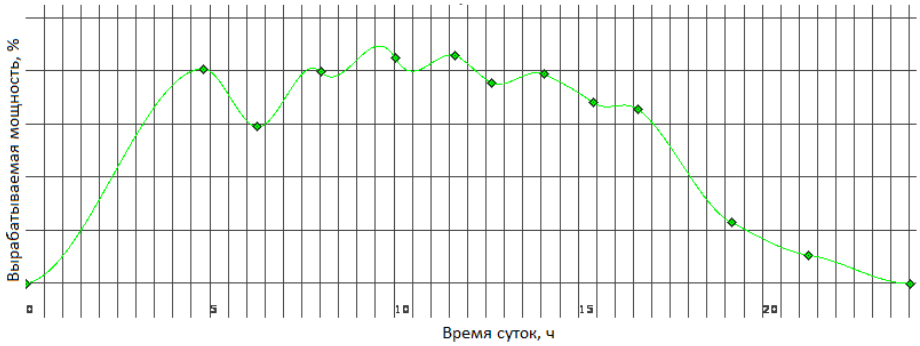


Рис. 2. График функции выработки электроэнергии за день у солнечной электростанции.

7. Заключение

На примере имитационной модели распределенной энергосистемы был продемонстрирован подход, который предоставляет возможность моделирования систем роевого интеллекта с нечеткой логикой в поведении агентов.

Такие системы отличаются высокой устойчивостью к сбоям, адаптивностью к показателям реального мира и гибкостью в регулировке. Пользователь системы может настраивать специфику поведения агентов, задавая нужные функции принадлежности и количественные характеристики в реакциях на события, которые программируются разработчиком системы.

Кроме распределенных энергосистем, примерами систем, в которых возникает потребность их моделирования, являются роевые системы управления беспилотными автомобилями. Подключенный к системе роевого интеллекта автомобиль может автономно двигаться в потоке машин, «предвидя» преграды и повороты, а также обмениваться данными с другими транспортными средствами. Информация о локальной дорожной обстановке и желаемых маневрах других участников движения, поступающей от сенсоров и камер – своих и от соседних автомобилей, позволяет выбирать оптимальные маневры, минимизировать риски столкновения. По этому принципу можно наладить децентрализованную транспортную систему с самоорганизующихся беспилотных автомобилей.

Другим примером использования данного подхода может быть моделирование систем сервисных роботов-спасателей. Природные бедствия (землетрясения, наводнения) и большие техногенные аварии несут с собой большие разрушения и забирают большое количество человеческих жизней. Реагировать на подобные ситуации нужно максимально быстро. Чем быстрее будут найдены пострадавшие, тем больше жизней можно спасти. Проникать в самые сложные завалы, места аварий и возгораний, выдерживая при этом большие нагрузки (высокие температуры, обводненность, отсутствие видимости и т.д.), могут роботы-спасатели. Оснащенные роевым интеллектом, камерами и сенсорами, сервисные работы могут действовать совместно, охватывая всю территорию катастрофы. Каждый из них действует по алгоритмам с учетом поведения других роботов децентрализованной системы и поступающих от них данных. Спасателю – оператору такой системы достаточно подавать отдельные команды, чтобы определять область обнаружения и регулировать поведение агентов системы.

Список литературы

1. Томашевський В.М. Моделювання систем. – К.: Видавнича група BHV, 2007. – 352 с.
2. Beasley J. E. and Whitchurch G. O. R. education – a survey of young O. R. workers. // Journal of the Operational Research Society. - 1984. - № 35. - P. 281 — 288.
3. Guillaume S.: Designing Fuzzy Inference Systems from Data: An Interpretability-Oriented Review // IEEE Transactions on Fuzzy Systems, 2001. V. 9, N. 3. P. 426-443.
4. M. Dorigo, V. Maniezzo, A. Colorni, “TheAntSystem: Optimization by a colony of cooperating agents” // IEEE Transactionson Systems, Man, and Cybernetics-Part B, 26, 1, стр. 29-41, 1996 г.

УДК 004.93(015.7)

КОЛІЩАК Б.В.,
БАКЛАН І.В.

ВІЗУАЛЬНИЙ ГЕНОМ: ЗВ'ЯЗОК МІЖ МОВОЮ ТА ЗОБРАЖЕННЯМ

Незважаючи на прогрес у перцептивних задач, таких як класифікація зображень, комп'ютери як і раніше погано працюють з когнітивними завданнями, такими як опис зображення. Ключовою ціллю роботи є не тільки пізнавальна здатність нейронних мереж, але і їхні міркування про наш візуальний світ. В той час, як моделі використовують багатий зміст образів зображення, механізми працюють з наборами даних, призначених для перцептивних завдань. Щоб домогтися успіху в розпізнавальних задачах, моделі треба розуміти взаємодію і відношення між об'єктами на зображенні. У цій статті представлений набір даних візуального генома, який дозволяє моделювати такі відносини. Для вивчення моделі збирається щільна анотація об'єктів, атрибутів і зв'язків зображення. Модель канонізує об'єкти, атрибути, відносини і субстантивні словосполучення на описаній ділянці і відповідає на питання пар-синонімів.

Despite progress in the field of perceptual tasks such as image classification, computers still can’t solve hard cognitive tasks, such as the description of some image. A key aim of the work is not only a cognitive capacity of neural networks but also their reasoning about our visual world. While the models use the rich content of image representation, the mechanisms work with datasets designed for perceptual tasks. It is necessary to understand the interaction and relationships between the objects in the image to succeed in the identification task model. This article presents a dataset of visual genome, which gives the opportunity to simulate such relationships. In order to explore the model, it is required to collect a specific information about the abstract objects, attributes and relations of the image. The model canonizes the objects, attributes, relationships, and noun phrases which are described on the site and answers the questions pairs of synonyms pairs.

1. Вступ

Огляд даних, необхідний для переходу від обізнаного сприйняття до когнітивного розуміння зображення, здійснюється щільною анотацією зображення з численними описами ділянки, об'єктів, атрибутів і зв'язків.

На рисунку показаний приклад, в якому компоненти:

1. Ділянка опису («Дівчина годує великого слона», «Чоловік робить знімок позаду дівчини»);
2. об’єкт («слон»);
3. атрибут («великий»);
4. зв’язок («годування»).

Метою комп'ютерного зору є повне розуміння візуальних сцен: модель, яка здатна виявляти об'єкти, описувати їх атрибути і створювати відносини між ними.

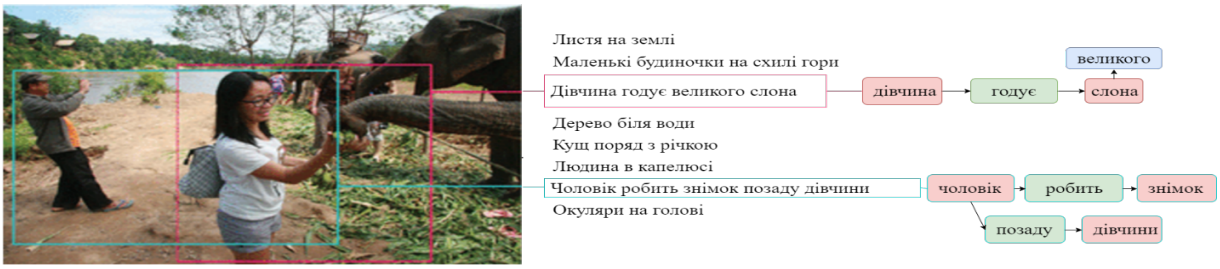


Рис. 1. Структуроване представлення частин зображення

2. Актуальність

Розуміння візуальних сцен дозволить додаткам, таким як пошук зображення, зробити значний прогрес у штучному інтелекті.

3. Проблема

Як показано на рисунку 1, існуючі моделі можуть виявити дискретні об'єкти на фотографії, але не в силах пояснити їх взаємодію. Подібні описи несуть пізнавальний характер, інтегруючи сприйняту інформацію для виведення відносин між об'єктами у візуальній сцені [1]. Когнітивне розуміння нашого візуального світу вимагає доповнити можливості комп'ютерів для виявлення об'єктів, зі здатністю описувати і розуміти їх взаємодію в межах сцени.

4. Рішення

Щоб ретельно зрозуміти образ, необхідно додати два ключових елементи до існуючих баз даних:

- повний набір описів для кожного зображення на основі декількох зображень;
- формалізоване подання компонентів зображення.

Візуальний геном – набір даних, здатний забезпечити структуроване формалізоване представлення зображення, в формі, яка широко використовується в датасеті NLP [2]. База даних складається з чотирьох основних компонентів: області опису, об'єктів, атрибутів і відносин (на рисунку 1 показані приклади кожного компонента для сцени). Крім того, всі об'єкти, атрибути і взаємозв'язки кожного зображення в наборі даних візуального геному є канонізовані до відповідного WordNet [3] ID. Це зіставлення пов'язує всі зображення у візуальному геномі і забезпечує ефективний спосіб послідовного запиту.

Щоб дослідити образ, збирається опис. Це сирі тексти і слова. Далі вилучаються об'єкти, атрибути та зв'язки з описів, які являють собою формальний образ зображення.

Кожне зображення складається з об'єктів, які окреслені обмежуючими прямокутниками.

Задля уникнення повторень між описами використовуються BLEU [4] (n - грами) бали між парами пропозицій. Визначається показник подібності S між описом d_i і попереднім описом d_j :

$$S_n(d_i, d_j) = b(d_i, d_j) \exp\left(\frac{1}{N}\sum_{n=1}^{N} \log p_n(d_i, d_j)\right),$$

де застосовується штраф за стисліть, використовуючи:

$$b(d_i, d_j) = \begin{cases} 1, & якщо\ len(d_i) > len(d_j); \\ e^{1-\frac{len(d_i)}{len(d_j)}} & в\ іншому\ випадку. \end{cases}$$

p_n обчислює відсоткову кількість n - грам в d_i, яка відповідає n -грамам в d_j. Коли описується новий об'єкт, примусово встановлюється порогове значення BLEU (рівне 0.7), щоб переконатися, що він відрізняється від інших описів.

Об'єкти можуть мати нуль або більше атрибутів, пов'язаних з ними, які

дозволяють описати, порівняти, і більш легко класифікувати об'єкти. Екстракція речення проводиться шляхом об'єднання об'єктів, атрибутів і відносин з кожної ділянки зображення.

5. Підбір зображень

Геном візуального представлення даних складається з 100.000 завантажених зображень з сервісу Flickr, а так само інших сховищ, що дозволяє провести повну анотацію вмісту фотографії, щоб уникнути втрат істотних описів аспектів областей на сцені.

6. Статистична модель

Одним із значущих компонентів візуального геному є його опис ділянки. Кожне зображення містить в середньому 30 областей з обмежуючими прямокутниками і їх описом.

Рис. 2. Приклад зображення з набору даних з описаними ділянками

Виділено лише шість описів, щоб уникнути безладу. Вони, як правило, різноманітні, охоплюють найбільш важливі частини і можуть зосереджуватися на одному, як «Рюкзак», так і на декількох об'єктах, як «Хобот слона хапає два банани».

Опис області має бути достатньо конкретним для опису окремих об'єктів («Водойма»), але так само має бути достатньо загальними, щоб описувати високорівневі концепції («Маленькі будиночки на схилі гори»). Відзначається, що регіони, які покривають великі ділянки зображення мають тенденцію загального опису, в той час як ділянки, які покривають малу частину – більш конкретного опису.

7. Семантичне розмаїття

Фактично, смисловий зміст опису використовує безконтрольний підхід до аналізу семантики. Зокрема, це попередньо навчена модель word2vec [5] від Google, яка перетворює кожне слово у вектор. Вдаючись до використання ієрархічної агломераційної кластеризації на векторних представленнях кожного опису – знаходимо семантичні та синтаксичні угруповання. На риснку 2 продемонстровано чотири кластери, кожен з яких відповідає за свою тему описів.

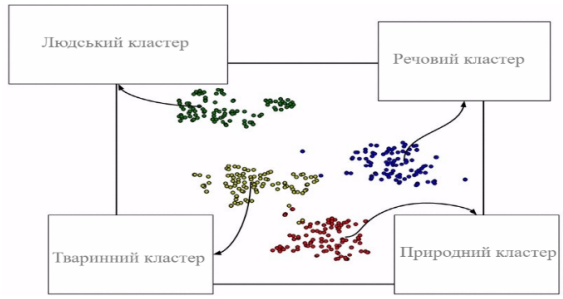


Рис. 3. Ілюстрація чотирьох груп кластерів

Щоб оцінити різноманітність описів семантики візуальним геномом, обчислюється число кластерів на ділянці. У середньому, число кластерів на зображенні чотири, що в двоє більше за показник інших тестованих баз даних.

8. Висновок

Візуальний геном забезпечує багаторівневе розуміння зображень. Це дозволяє з різних точок зору вивчити сцену з пікселів на рівні об'єктів, дискретно описуючи ділянки

зображення, що дозволяє глибше описувати когнітивні завдання. З візуальним геномом, моделі демонструють ширше розуміння нашого візуального світу, доповнюючи можливості комп'ютерів виявляти об'єкти з можливостями їх опису, поясненнями їх взаємодій і відношень. Візуальний геном – це велике формалізоване подання знань для візуального розуміння з повним набором описів, що становить основу візуальної концепції мови.

Список літератури

1. Bruner, J. (1990). Culture and human development: A new look. Pp. 344–355.
2. Gao, H., Mao, J., Zhou, J., Huang, Z., Wang, L., & Xu, W. (2015). Are you talking to a machine? Dataset and methods for multilingual image question. Pp. 2296–2304.
3. Miller, G. A. (1995). Wordnet: a lexical database for english. Pp. 39–41.
4. Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). BLEU: A method for automatic evaluation of machine translation. Pp. 311–318.
5. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. Pp. 2-10.

УДК 519.8

ВОРОНА М. В.

ЗАСТОСУВАННЯ АГЕНТІВ ЛОКАЛЬНОГО ПОШУКУ В ЕВОЛЮЦІЙНИХ АЛГОРИТМАХ ДЛЯ РОЗВ'ЯЗАННЯ VRPTW

Розглядається застосування алгоритмів локального пошуку як агентів, що покращують дослідження простору розв'язків задачі VRPTW еволюційним алгоритмом. Описується адаптація локального пошуку, призначена для використання у гібридному алгоритмі.

The subject of this article is an application of local search algorithms as an agents of evolutional algorithm that improve investigation of the solutions set of VRPTW. Article includes description of the adaptation of local search in order to increase quality of VRPTW.

1. Вступ

Виникнення досліджень у галузі транспортної маршрутизації почалося в 1959 р., коли Данцигом і Рамсером була сформульована задача оптимізації вантажних перевезень [1]. Суттєві дослідження щодо формалізації задачі маршрутизації транспорту було зроблено на проміжку між 1971 р. [2] та 2002 р. [3], втім останнім часом нові технології та вимоги практики обумовлюють виникнення нових варіацій цієї задачі. Однією з найбільш важливих варіацій задачі маршрутизації є задача маршрутизації транспорту із часовими вікнами (Vehicle Routing Problem With Time Windows, VRPTW), яка включає в себе додаткові обмеження на час прибуття транспорту до точки призначення.

Розв'язки цієї задачі використовуються в компаніях, де критичним є час доставки – кур'єрських службах, доставках їжі, сервісах таксі.

В статті подається адаптований до VRPTW алгоритм локального пошуку. Розглядається ідея включення в локальний пошук ущільнення розв'язку з метою мінімізації кількості маршрутів, що утворюють розв'язок. Проведено порівняльний аналіз результатів роботи генетичного алгоритму без та з локальним пошуком.

2. Постановка VRPTW

Змістовна постановка.

Існує набір споживачів, яким потрібно доставити вантаж. Обсяг вантажу може відрізнятися для кожного споживача. Товар для конкретного споживача можна доставити лише у певний проміжок часу. Для доставки існує парк машин, які мають однакову вантажопідйомність і знаходяться в депо. Потрібно побудувати такі маршрути для кожної машини, щоб сумарно усі споживачі були обслуговані в свої проміжки часу, і при цьому було використано якнайменше машин, а сумарна пройдена відстань була мінімізована.

Формальна постановка.

VRPTW задається множиною гомогенних транспортних засобів V, множиною споживачів C та направленим графом $G = (V, C)$. Граф складається з $|C| + 2$ вершин, де споживачам відповідають вершини від 1 до n, а стартова точка-депо позначена вершиною 0.

VRPTW – це багатокритеріальна задача, ціль якої – мінімізувати кількість необхідних транспортних засобів, а також сумарну відстань, яку ці засоби подолають. Множина дуг A подає зв'язки між стартовою точкою і споживачами, а також між самими споживачами. Кожна дуга починається і закінчується в вершині 0. Кожна дуга асоціюється з часом t_{ij} та вартістю c_{ij}, які можуть бути відображені єдиним значенням (час і буде вартістю).

Кожний транспортний засіб має вантажопідйомність q, а кожний споживач i – потребу в вантажі d_i. Споживач i також характеризується часовим вікном $[a_i, b_i]$. Транспорт має прибути до споживача до того, як настане час b_i. Він може прибути раніше аніж почнеться час a_i, але транспорт не почне розвантаження раніше, аніж настане a_i. Депо також має часове вікно $[a_0,$

b_0]. Усі машини не можуть виїхати з депо раніше a_0 та не можуть повернутися пізніше b_0. На q, a_i, b_i, d_i, c_{ij} накладено обмеження невід'ємності та цілочисельності, тоді як t_{ij} – натуральні числа. Модель VRPTW включає в себе множину змінних x_{ijk} ($i \neq j$, $i \neq n+1$), кожна з яких приймає значення 1, якщо в отриманому розв'язку транспортний засіб k їде від вершини i до j, а інакше $x_{ijk} = 0$.

s_{ik} – час, коли транспорт k починає розвантаження у споживача i.

Розв'язком задачі має бути набір маршрутів для кожного транспортного засобу, такий, що кожен споживач буде відвіданий рівно один раз, при цьому кожен маршрут починається і закінчується в депо.

Цільова функція:

$$min \sum_{k \in V} \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ijk} \quad (1.1)$$

Обмеження:

$$\sum_{k \in V} \sum_{j \in N} x_{ijk} = 1, \quad \forall i \in C \quad (1.2)$$

$$\sum_{i \in C} d_i \sum_{j \in N} x_{ijk} \leq q, \quad \forall k \in V \quad (1.3)$$

$$\sum_{j \in N} x_{0jk} = 1, \quad \forall k \in V \quad (1.4)$$

$$a_i \leq s_{ik} \leq b_i, \qquad \forall i \in N, \forall k \in V \quad (1.5)$$

$$x_{ijk} \in \{0, 1\}, \qquad \forall i, j \in N; \ \forall k \in V \quad (1.6)$$

3. Еволюційний алгоритм з агентами локального пошуку

Еволюційні методи (ЕМ) призначені для пошуку кращих розв'язків і засновані на статистичному підході до дослідження ситуацій й ітераційному наближенні до шуканого стану систем.

На відміну від точних методів математичного програмування ЕМ дозволяють знаходити розв'язки, близькі до оптимальних, за прийнятний час, а на відміну від відомих евристичних методів оптимізації характеризуються істотно меншою залежністю від особливостей додатка (тобто більш універсальні) і в більшості випадків забезпечують кращий ступінь наближення до оптимального розв'язку.

Найважливішим класом ЕМ є генетичні алгоритми. Генетичні алгоритми засновані на пошуку кращих розв'язків за допомогою спадкування й посилення корисних властивостей множини об'єктів певного додатка в процесі імітації їхньої еволюції. [4]

4. Модифікація локального пошуку для VRPTW

Алгоритми детермінованого локального пошуку - це сім'я ітераційних методів, заснована на частковому перебиранні варіантів на кожній ітерації серед точок околу поточної точки, тобто серед сусідніх до неї.

В алгоритмах цього типу замість повного перебору застосовується спрямований локальний перебір у підмножинах варіантів, які називаються околами. Цим пояснюється їх назва - алгоритми локального пошуку (ЛП, Local search). У сфері комбінаторної оптимізації алгоритми ЛП мають давню історію через свою наочність і високу ефективність. Наприклад, перший алгоритм локального пошуку для задачі комівояжера був запропонований ще в 1956 р., а локальний пошук для задачі розміщення обладнання був розроблений у 1962 р. Загальна схема ЛП у задачах мінімізації така: починаючи з деякого припустимого розв'язку задачі, новий розв'язок із кращим значенням цільової функції шукають у його околі. Якщо такий розв'язок знайдено, то він приймається і пошук поліпшення розв'язку далі здійснюється вже в його околі і т. д. Алгоритм закінчується, коли досягнуто локального оптимуму, тобто коли в околі поточного розв'язку немає ніякого іншого варіанта з меншим значенням цільової функції [4].

Оскільки розв'язок VRPTW має оптимізуватися в двох напрямках – мінімізації кількості використаних машин та мінімізації сумарної пройденої відстані, то використаний алгоритм локального пошуку має враховувати обидва ці напрямки. Тому запропонований алгоритм складається з двох частин.

Перша частина намагається максимально ущільнити наявні маршрути, переставляючи споживачів з найкоротших шляхів в інші наявні. Якщо маршрут стає пустим, він видаляється. Навіть якщо за результатами роботи алгоритму жоден маршрут видалити не вдалося, усе одно отримана сукупність маршрутів стає більш щільною в тих маршрутах, де це можливо, що звільняє простір для подальших маніпуляцій.

Друга частина базується на звичайному локальному пошуку, який для знаходження нових варіантів із простору розв'язків використовує класичну заміну 2-opt. Застосовуючи заміну до VRPTW, вона значитиме наступне: узяти споживача *a* з одного маршруту і знайти такого споживача *b* з іншого маршруту, що якщо поміняти *a* і *b* місцями, то усі маршрути залишаться валідними, а сумарна відстань зменшиться.

Псевдокод алгоритму локального пошуку для VRPTW:

```
Procedure local_search(solution)
solution := sort_routes_by_customers(solution)
while True
break_flag := False;
for i in len(solution) – 1
if break_flag == True
break;
end if;
for j in i + 1, len(solution)
if break_flag == True
break;
end if;
for k in 1, len(solution[i]) – 1
if break_flag
break;
end if;
for l in 1, len(solution[j]) – 1
mutable_route := solution[j];
mutable_route.insert(l, solution[i][k])
if check_route_validity(mutable_route)
price := count_price(mutable_route)
if price < best_price
best_price := price
best_donor := k
best_recepient := l
end if;
delete mutable_route[l]
end if;
if best_donor
donor := solution[i][best_donor]
solution[j].insert(best_recepient, donor)
delete solution[i][best_donor]

if len(solution[i]) < 3
delete solution[i];
end if;
break_flag := True;
break;
end if;
if break_flag == False;
break;
end if;
end while;

# local distance improvement
while True
break_flag := false;
for i in len(solution) – 1
if break_flag == True
break;
for j in i + 1, len(solution)
if break_flag
break;
end if;
for k in 1, len(solution[i]) – 1
if break_flag
break;
end if;
for l in 1, len(solution[j]) – 1
if distance_decreased(i, j, k, l)
switch_elements(i, j, k, l)
if is_solution_valid(solution)
break_flag := True;
break;
else
switch_elements(i, j, k, l);
end if;

if break_flag == False
break;
end if;
end while;
return solution;
```

sort_routes_by_customers - розташовує маршрути в порядку від найменшої кількості споживачів до найбільшої.

check_route_validity - перевіряє, чи залишився розв'язок валідним.

count_price – повертає вартість вставки споживача в маршрут.

switch_elements – міняє місцями споживачів *k*, *l* в маршрутах *i*, *j* відповідно.

Псевдокод генетичного алгоритму з локальним пошуком:

```
Procedure GeneticLocalSearch(population)
for member in population
member := local_search(member);
end for;
best_solution = populatin[0];
while counter < iterations_number
counter := counter + 1;
for member in population
par1 := tournament_candidate(population);
par2 := tournament_candidate(population);
child := crossover(par1, par2);
new_population.insert(child);
end for;
population := population + new_population;

for member in population
if random() <
MUTATION_PROBABILITY
member := perform_mutation(member);
member := local_search(member);
end if;
end for;

sort_by_distance(population);
cut_population_size(population);

if first_better(population[0], best_solution)
best_solution := population[0];
end if;

end while;
return best_solution;
```

tournament_candidate - взяти двох випадкових членів популяції і повернути кращого

crossover – схрестити двох членів популяції і повернути нащадка

perform_mutation – випадковим чином змінити розв'язок

sort_by_distance – відсортувати членів популяції за пристосованістю (від найкращого до найгіршого)

cut_population_size – залишити лише *N* найкращих членів популяції

first_better – True якщо перший розв'язок краще за другий

5. Методологія випробувань

Для перевірки впливу агенту локального пошуку на розв'язок VRPTW була проведена серія випробувань з генетичним алгоритмом без локального пошуку та з ним, за результатами якої узяті найкращі результати. Алгоритми запускалися на однакових початкових популяціях. Вхідні дані узяті з стандартної бібліотеки VRPTW – Solomon [5]. Кожна задача містить 100 споживачів з часовими вікнами, задачі розділені на 2 типи – C та RC. В задачах типу C користувачі кластеризовані, в RC використана суміш кластеризованих та випадкових точок.

Параметри генетичного алгоритму:
Розмір популяції – 15
Кількість ітерацій – 20
Імовірність мутації – 0.3

Оскільки метаевристики не гарантують точного результату а лише покроково наближають розв'язок до оптимуму, то результати кожного разу можуть бути різні. Тому обчислювальний експеримент проводився 5 разів, кожного разу з новою початковою популяцією.

Програмна частина розроблена на мові Python. Параметри процесора: 2.7 GHz Intel Core i5, оперативна пам'ять – 8 GB

6. Результати

Результати роботи обох алгоритмів – з локальним пошуком та без – приведені в таблиці 1.

	З локальним пошуком		Без локального пошуку	
Задача	**Min t**	**Max t**	**Min t**	**Max t**
C101	77.48	99.66	71.66	86.6
C102	113.06	172.6	80.72	128.63
C103	186.75	241.94	170.59	232.77
C104	355.24	399.35	240.48	320.2
C105	57.66	73.35	57.36	89.48
C106	115.33	127.03	91	125.03
C107	97.26	122.67	70.74	89.69
C108	153.67	224.37	130.95	152.4
C109	220.85	249.71	179.69	222.68
RC101	200.67	228.16	197.07	229.64
RC102	197.44	305.19	202.12	259.09
RC103	293.65	397.69	245.52	255.09
RC104	275.87	475.19	344.97	420.35
RC105	249.01	371.07	224.23	313.3
RC106	226.39	341.07	218.88	361.24
RC107	226.46	404.99	267.19	301.08
RC108	295.1	509.08	287.43	381.52

Табл. 1. Результати роботи алгоритмів

Отримані результати ілюструють, що на задачах типу C (з зарані кластеризованими споживачами) локальний пошук інколи заганяв розв'язки в локальний оптимум, не даючи змоги генетичному алгоритму дійти до кращого варіанту. При цьому кількість маршрутів була однакова, що можна пояснити кластеризацією – задачі типу C спеціально підготовані для легкого знаходження оптимальної кількості маршрутів.

На випадкових даних в задачах типу RC ж можна побачити зовсім іншу картину – на більшості результатів видно що локальний пошук не просто зменшив сумарну відстань, але й зменшив кількість необхідних машин, що є серйозною перевагою.

Порівнюючи час роботи алгоритмів можна помітити, що сумарна частка локального пошуку в часі роботи алгоритму не є суттєвою.

Висновок:

Було розроблено алгоритм локального пошуку для VRPTW та продемонстровано ефективність його роботи в якості агенту, застосованого для кожного члена популяції генетичного алгоритму. Створений алгоритм локального пошуку доцільно застосовувати на некластеризованих даних, які і зустрічаються в реальних задачах. Основний напрямок роботи даного алгоритму локального пошуку – ущільнення маршрутів та як наслідок зменшення кількості необхідних машин.

Список літератури

1. Dantzig G.B., Ramser J.H. The truck dispatching problem // Management Science – 1959. – №1. – с. 80-91.

2. Eilon S., Watson-Gandy C. D. T., Cristofides N. // Distribution Management. – 1971.

3. Toth P., Vigo D. The Vehicle Routing Problem // Society for Industrial and Applied Mathematics. – 2002. – 386 с.

4. Гуляницький Л.Ф., Мулеса О.Ю. // Прикладні методи комбінаторної оптимізації. - Видавничо-поліграфічний центр "Київський університет", 2016. – 142 с.

5. Solomon benchmark problems [Електронний ресурс] // Режим доступу: http://people.idsia.ch/~luca/macs-vrptw/problems/welcome.htm

УДК 004.043

ГАВРИЛЕНКО О.В.,
ГЕТЬМАНЕНКО О.В

ПОШУК НЕОБІХДНОГО КОНТЕНТУ З НЕСТРУКТУРОВАНИХ ДАНИХ НА ПРИКЛАДІ ЗНАХОДЖЕННЯ ЗОБРАЖЕНЬ НА САЙТАХ

У даній статті розглядається проблема пошуку інформації з неструктурованого масиву даних. Для прикладу, це контент сайту чи електронне повідомлення. Досить часто постає завдання вибірки корисних матеріалів з великого масиву таких даних. Це може бути електронна адреса, номер телефону, зображення, стаття. Для таких випадків використовуються регулярні вирази і створюються патерни для пошуку відповідної інформації. Проте інколи такий пошук не може бути однозначним: зображення можуть мати різні розширення, а інколи вони взагалі можуть міститися на інших ресурсах. Це призводить до труднощів створення однозначних регулярних виразів, і, як наслідок, безпосередньо впливає на швидкість роботи алгоритму. Розглянуто обхідні рішення і приклади боротьби з такими ситуаціями, проаналізовано результати тестування засобів парсингу і визначено рекомендації по їх вибору.
Ключові слова: НЕСТРУКТУРОВАНІ ДАНІ, ПАРСИНГ, РЕГУЛЯРНІ ВИРАЗИ, HTML-РОЗМІТКА.

This article discusses the problem of finding information from unstructured data array. For example, it can be site content or message. Often, the goal is to select useful materials from a large array of data. An example of such data is the email address, phone number, pictures, articles. For such cases regular expressions are used and search patterns for relevant information are created. But sometimes such a search may not be exact: images can have different extensions, and sometimes they even may be stored in other media resources. This leads to difficulties in creating exact regular expressions, and therefore directly affects the speed of the algorithm. Considered workarounds and examples how to deal with such situations, analyzed test results of parsing tools, and determined recommendations for their selection.
Keywords: UNSTRUCTURED DATA, PARSING, REGULAR EXPRESSIONS, HTML-MARKUP.

1. Вступ

У даній статті розглядається проблема пошуку інформації з неструктурованого масиву даних. Для прикладу, це контент сайту чи електронне повідомлення. Досить часто постає завдання вибірки корисних матеріалів з великого масиву таких даних. Це може бути електронна адреса, номер телефону, зображення, стаття. Для таких випадків використовуються регулярні вирази і створюються патерни для пошуку відповідної інформації.

Проте інколи такий пошук не може бути однозначним: зображення можуть мати різні розширення, а інколи вони взагалі можуть міститися на інших ресурсах. Це призводить до труднощів створення однозначних регулярних виразів, і, як наслідок, безпосередньо впливає на швидкість роботи алгоритму.

Розглянуто обхідні рішення і приклади боротьби з такими ситуаціями, проаналізовано результати тестування засобів парсингу і визначено рекомендації по їх вибору.

2. Регулярний вираз для пошуку зображення

Регулярний вираз – це рядок, що описує або збігається з множиною рядків, відповідно до набору спеціальних синтаксичних правил. Вони використовуються в багатьох текстових редакторах та допоміжних інструментах для пошуку та зміни тексту на основі заданих шаблонів.

Елементарний вигляд регулярного виразу для пошуку зображення:

/<img[^>]+>/i

У даному випадку відбувається пошук усіх зображень, що містяться у стандартному html тезі – img.

Загальний алгоритм вибірки зображень зі сторінки буде мати наступний вигляд:

1. зчитування даних зі сторінки (отримання всього html-контенту);
2. пошук зображень у відповідності до заданого патерну;
3. збереження посилань на зображення у масив.

Якщо сторінка містить лише зображення зі стандартними розширеннями, то схема даного алгоритму повністю задовольнить умови. Однак здебільшого сайти розміщують статичні дані на сторонніх ресурсах (одним з відомим є Amazon) – це правильно з архітектурної точки зору, оскільки такий підхід дозволяє зменшити навантаження на сервер, на якому відбуваються розрахунки і ведуться логічні дії. Проте ситуація ускладнює пошук і алгоритм, наведений вище, перестає працювати. Шаблон не буде задовольняти усіх вимог. Потрібно змінити його – розширити можливість пошуку лінків, на яких може знаходиться зображення. Варто врахувати, що посилань на сайті міститься велика кількість і проста вибірка усіх лінків нам не допоможе. Знайдені посилання міститимуть інші статті, CSS стилі, JS, посилання на відеоролики. Однак ми можемо видозмінити наший алгоритм наступним чином: додати перевірку знайденого ресурсу, а перед цим ще робити відсіювання стандартних розширень файлів – html, css, js.

Отож, алгоритм набуде наступного вигляду:

1. зчитування сторінки;
2. пошук лінків, зображень;
3. збереження знайдених даних, що співпали з патерном у масив;
4. перебіг по масиву (якщо невідоме розширення – визначаємо тип, якщо це зображення – додаємо, інше – відкидаємо).

Це загальний вигляд алгоритму. У наступному розділі поговоримо про оптимізацію.

3. Варіанти оптимізації пошуку зображень

Описані алгоритми будуть працювати, проте припустимо, що перед нами стоїть завдання оптимізації, оскільки не кожен користувач захоче очікувати на знайдені зображення довгий час.

Один із методів покращення – це попередній аналіз сторінки. Стандартний вигляд і структура веб-сторінки представлена на рисунку 1.

```
<html>
  <head>
    <meta ...>
  <titile>Example web page</title>
  </head>
  <body>
    <h1>Header</h1>
    <p>First article</p>
    <p>Second article</p>
    <p>Third article</p>
   <img src="..." alt="image 1">
    <img src="..." alt="image 2">
    <img src="..." alt="image 3">
  </body>
</html>
```

Рис. 1. Загальна структура веб-сторінки

Усі зображення, які є актуальними, знаходяться у тілі тега body, тому доцільно відкинути іншу розмітку і аналізувати контент лише з цього місця. Усе просто, однак є ситуації, коли розробники задають зображення через прописування у стилях (через css, який знаходиться в head). Тому доцільно ще проаналізувати файл стилів. Варто пам’ятати, що деякий контент виводиться динамічно, а тому при зчитуванні сторінки потрібно враховувати цей момент. Це призведе до потреби аналізу JavaScript.

Отож, після модифікації алгоритм виглядатиме наступним чином:

1. зчитування конкретної частини сторінки;
2. пошуку лінків, зображень;
3. збереження знайдених даних, що співпали з патерном у масив;
4. перебіг по масиву (якщо невідоме розширення – визначаємо тип, якщо це зображення – додаємо, інше – відкидаємо).

Інший варіант прискорення – це оптимізація і розгляд уже самих патернів. Якщо враховувати, що патерни складені правильно і уже є оптимізованими – залишається покращувати самі алгоритми.

Один із варіантів – використання не звичайних регулярних виразів, а так званих парсер-комбінаторів, які дозволять покроково знаходити і аналізувати необхідні елементи.

Залишається ще один варіант прискорення – розгляд роботи регулярних виразів із середини і оптимізація алгоритму для пошуку зображення. Задля цього варто розглянути розуміння кінцевих автоматів.

4. Проведення експериментів

Розглянемо можливі методи вилучення інформації за допомогою мови програмування PHP 7 версії. Ця мова має високу продуктивність і підтримку сучасного об’єктно-орієнтованого стилю програмування. На даний момент не існує подібного дослідження для засобів мови PHP, не дивлячись на довге існування деяких з них.

Для парсингу веб-документів HTML, XHTML, XML стандартнми засобами PHP частіш за все застосовуються програмні інтерфейси (API) DOM (Document Object Model), SAX (Simple API for XML), SimpleXML, XMLReader. Ці типи інтерфейсів стандартно включені в дистрибутиви PHP, базуються на бібліотеці libxml2 проекту GNOME.

За способом завантаження веб-документа API бувають із завантаженням у пам’ять повністю або з потоковим завантаженням (послідовно по частинах), у останньому випадку розрізняють передаючий (push) аналізатор (на основі подій) і приймаючий (pull) аналізатор (курсорний).

У даному дослідженні проаналізовано також неспеціалізовані засоби парсинга, що використовують функції роботи з рядками, і спеціальну формальну мову регулярних виразів (RegExp). Регулярні вирази в PHP реалізовані в бібліотеці PCRE (Perl-сумісні регулярні вирази). Розробка на їх основі вимагає реалізації під конкретне завдання.

У ході тестування дотримані певні умови:

- усі коди парсинга виконують однакову корисну роботу, таким чином знаходяться в рівних умовах один з одним. Для досліджень обрано ситуацію збору посилань зображень з HTML документа;
- перевіряється коректність вилучення посилань з тестової сторінки, в тому числі в невалідному форматі, вкладених в елементах і в коментарях;
- враховується наявність накладних витрат на роботу з кодуваннями. Код прикладу, який використовує засіб парсинга, повинен бути доповнений кодом, який вирішує проблему різних кодувань, якщо, спираючись на свої можливості, засіб не може цього робити.

Проведемо дослідження з використанням готових існуючих бібліотек для пошуку у текстовому масиві даних і створеному регулярному виразі.

Розглянемо коротко найбільш популярні бібліотеки і потім порівняємо зі швидкодією описаних алгоритмів (регулярного виразу).

Розширення DOM дозволяє працювати з XML-документами через DOM API з PHP. Це реалізація W3C DOM Core Level 3, незалежний від мови і платформи інтерфейс, який дозволяє програмам і скриптам динамічно отримувати доступ і змінювати зміст, структуру і оформлення документів.

XMLReader – синтаксичний аналізатор XML. Клас-зчитувач виступає в якості курсору, слідує по потоку документа.

SimpleXML представляє досить простий і легкий у використанні набір інструментів для перетворення XML в об’єкт, з яким можна працювати через його властивості і за допомогою ітераторів.

phpQuery – це серверний API DOM, відмінною якістю якого є використання селекторів CSS і ланцюжка викликів методів, заснований на використанні JavaScript бібліотеки JQuery і підтримки інтерфейсу командного рядка.

Zend_dom представляє інструменти для роботи з документами і структурами DOM. Пропонується інструмент Zend_Dom_Query з підтримкою єдиного інтерфейсу для запитів до документів DOM з використанням XPath і CSS селекторів.

QueryPath це бібліотека для роботи з XML і HTML, створена для роботи не тільки з локальними файлами, але і з веб-службами і ресурсами БД. У ній реалізована більша частина інтерфейсу JQuery, але, на відміну від неї, розрахована на використання на стороні сервера.

fDOMDocument розширює стандартний DOM використанням винятків для обробки помилок замість виводу warnings чи notices. Також у бібліотеці реалізовані різноманітні методи і шорт-карти для більш комфортного і зручного використання DOM.

Для дослідження швидкодії пошуку зображень візьмемо звичайні статичні сторінки з web-у, оберемо Інтернет-магазини, оскільки вони мають достатньо велику кількість зображень. Для початку дослідимо і знайдемо зображення за адресою сайту http://oceanplaza.com.ua/about/photo/.

Результати будуть наступними:

Табл. 1. Результати тестів

Метод (API)	**Кількість знайдених зображень**	**Час роботи**	**Примітка**
DOM	40	0.75682401657104	-
phpQuery	40	0.82092118263245	-
Zend_dom	40	0.80724906921387	-
QueryPath	40	0.69632387161255	-
XMLReader	-	-	Потребують додаткової валідації HTML сторінки
Simple	-	-	
Регулярний вираз	40	0.59101920239413	-

Для аналізу ефективності роботи використовували один і той самий сервер. Варто відмітити, що кожен з API має свої особливості. Наприклад, ми не проводили приведення HTML коду до валідного XML. Для виправлення веб-документів з невалідною розміткою застосовуються спеціальні інструменти (приклад, HTML Tidy).

Проведемо аналогічні дослідження, але використаємо кілька ітерацій і дослідимо час виконання. Результати дослідження представлені на рисунку 2.

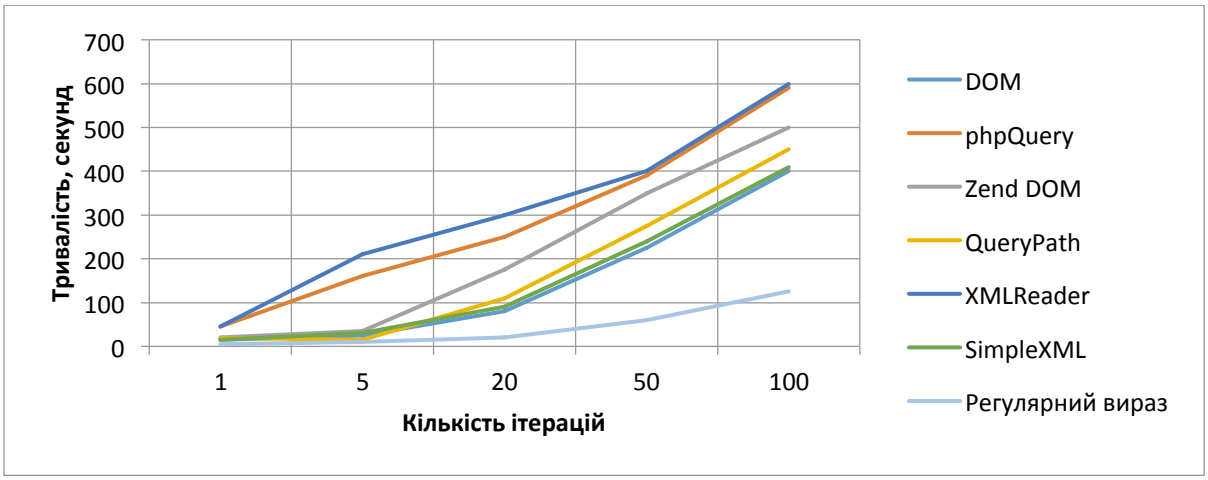


Рис. 2. Залежність виконання обробки документу розміром 150 мб парсером від кількості ітерацій

Розглянемо ресурси, які затрачаються при виконанні парсингу. При порівнянні спожитої пам'яті засобами парсинга виявлено, що:

- при разовому вимірюванні максимально використаної пам'яті при обробці HTML документів виграють засіб Expat з перетворенням в структуру і засоби парсинга за допомогою регулярних виразів і строкових функцій, а також XPath версії засобів fDOMDocument, DOM, вільний DOM, проте Expat зі створенням структури не ефективний для обробки XML документів;
- у динаміці зростання ітерацій спостерігаються витоки пам'яті у Advanced HTML DOM, phpQuery, fDOMDocument (CSS і XPath), вільний DOM (CSS), а також у Ganon і Simple HTML DOM;
- потокові засоби Expat і SimpleXMLReader виграють в обробці XML документів у інших засобів, однак у SimpleXMLReader було відзначено зростання споживання ресурсів з ростом кількості ітерацій.

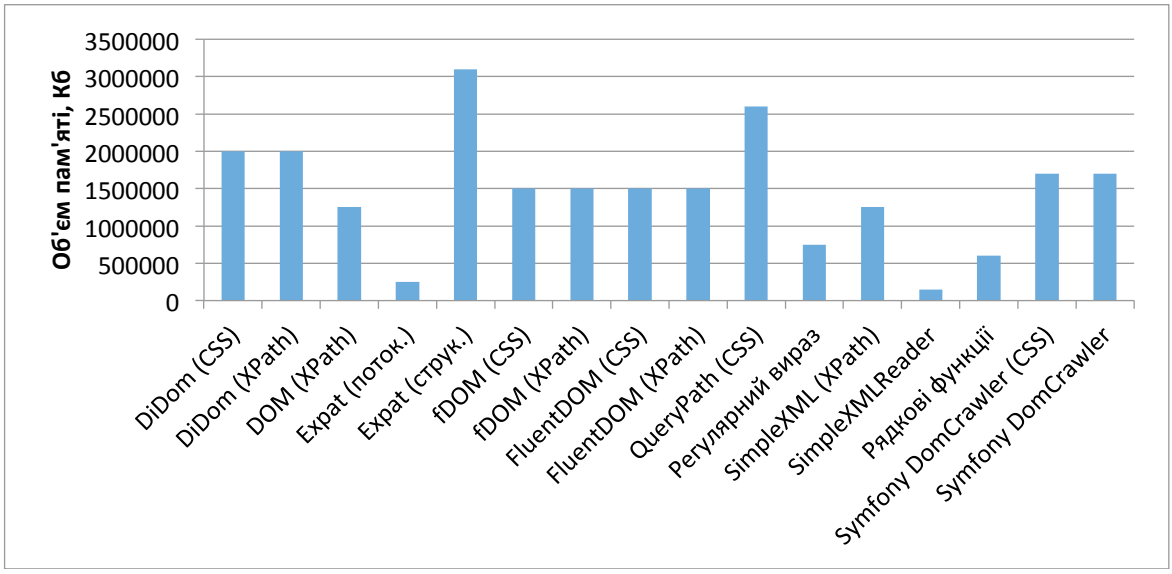


Рис. 3. Використання пам'яті при обробці XML документу розміром 150 Мб

5. Висновок

Засоби парсинга, у яких кожен аналізований веб-документ зберігається в пам'яті у вигляді ієрархії його елементів (вузлів), при повторному зверненні до різних частин документа мають менший час обробки веб-документа, оскільки не потрібно заново завантажувати і будувати дерево. Такі засоби надають найбільш зручний спосіб доступу до елементів веб-документа за допомогою запитів (CSS і XPath) і підходять для будь-яких завдань багаторазової вибірки інформації з веб-документа.

Засоби парсинга з потоковою обробкою використовуються в задачах, де повторне звернення не потрібно. У такому випадку відбувається послідовний перебір вузлів до потрібного і почергового завантаження в пам'яті тільки поточного вузла. Дані засоби можуть аналізувати дуже великі документи і підходять для таких завдань як індексація або перетворення в інші формати (наприклад, заміна дескрипторів XML на дескриптори HTML).

Засіб парсинга регулярними виразами застосований для задач, де необхідно вилучити конкретну частину документа із заздалегідь відомої структури. Швидкий пошук відповідників з шаблонами, написаними за допомогою регулярних виразів, гарантує ефективний результат за умови надійно складених виразів. Синтаксис регулярних виразів набагато складніший, ніж у CSS селекторів і навіть XPath запитів, а також чутливий до невеликих змін у розмітці. Такий засіб парсинга доцільніше використовувати там, де засоби, що використовують стандартні розширення бібліотеки LibXML, не

можуть допомогти, наприклад, у вилученні коментарів у коді або фрагментах коду перед розбором будь-яких інших засобів.

Отож, якщо потрібно написати аналізатор тексту для конкретного завдання, більш доцільно використовувати регулярний вираз, складений з урахуванням усіх оптимізацій, оскільки це буде одним із кращих варіантів рішень. А для прискорення виконання цьому регулярному виразу необхідно подавати на аналіз попередньо відфільтровану інформацію.

Список літератури

1. Гетьманенко О. В. Створення парсер-комбінаторів з подальшою оптимізацією // «Теорія і практика сучасної науки» (м. Дніпро, 24-25 лютого 2017 р.). — Херсон : Видавничий дім "Гельветика", 2017.
2. Лыгина Н.И., Пудич А.С. Исследование правильности и эффективности средств парсинга информации на веб-ресурсах [Електронний ресурс] // Инновационная наука. 2017. №3-1. URL: http://cyberleninka.ru/article/n/issledovanie-pravilnosti-i-effektivnosti-sredstv-parsinga-informatsii-na-veb-resursah.
3. Schrenk M. Webbots, Spiders, and Screen Scrapers 2nd edition: A Guide to Developing Internet Agents with PHP/CURL. No Starch Press Inc. 2012. P. 362.
4. Чеботарев А. XML: свобода, ограниченная только фантазией [Електронний ресурс] // ЦИТ Форум. 2004. URL: http://citforum.ru/internet/xml/xml_fant/.
5. Бенчмарк HTML парсеров [Електронний ресурс] // Сайт «Хабрахабр» – Разработка. 2012. 26 декабря. URL: https://habrahabr.ru/post/163979/.
6. Морган К. [Morgan C.] XML для PHP-разработчиков: Часть 2. Расширенные методы парсинга XML: пер. с англ. [Електронний ресурс] // Сообщество developerWorks. 2010. 15 апреля. URL: http://www.ibm.com/developerworks/ru/library/x-xmlphp2/index.html.
7. Расти Хэролд Э. [Rusty Harold E.] Синтаксический анализ XML в PHP: пер. с англ. [Електронний ресурс] // Сообщество developerWorks. 2007. 11 октября. URL: http://www.ibm.com/developerworks/ru/library/x-pullparsingphp/index.html.
8. Gervasio A. Introducing SimpleXML in PHP 5 [Електронний ресурс] // Dev Shed. 2006. 12 июня. URL: http://www.devshed.com/c/a/PHP/Introducing-SimpleXML-in-PHP-5/.
9. Grant J. PHP and XML: Using the expat functions [Електронний ресурс] // 2000. URL: http://www.phpbuilder.com/columns/justin20000428.php3.
10. ZF2012-02: Denial of Service vector via XEE injection [Електронний ресурс] // Zend Framework – Security. 2012.20 августа. URL: https://framework.zend.com/security/advisory/ZF2012-02.

УДК 004.023

СВІТЯЩУК О.В.,
ОЛІЙНИК Ю.О.

СПОСІБ КЛАСТЕРИЗАЦІЇ ВІДВІДУВАЧІВ РЕСТОРАННИХ ЗАКЛАДІВ

В даній статті розглянуто спосіб класифікації відвідувачів ресторанних закладів. Сформульовано задачу та її цілі з точки зору маркетингових стратегій ресторанних закладів. Виділено основні фактори для маркетингового аналізу, а також фактори, за якими здійснюється кластеризація відвідувачів. Описано алгоритм ієрархічної кластеризації.

Ключові слова: ЗАДАЧА КЛАСТЕРИЗАЦІЇ, ІЄРАРХІЧНА КЛАСТЕРИЗАЦІЯ, МАРКЕТИНГОВА СТРАТЕГІЯ, РЕСТОРАННИЙ ЗАКЛАД.

In this article is concerned with the classification of visitors of restaurant establishments. Formulated the problem and its objectives in terms of marketing strategies restaurant facilities. Marked as important factors for market analysis and the factors of clustering visitors. Described the algorithm of hierarchical clustering.

Key words: PROBLEM OF CLUSTERING, HIERARCHICAL CLUSTERING, MARKETING STRATEGY, CATERING ESTABLISHMENTS.

1. Задача отримання інформації про групи відвідувачів в ресторанному закладі

Підприємства сервісу постійно знаходяться в стані потреби відгуку своїх клієнтів. Це, насамперед, зв’язано з безперервним розвитком в умовах конкуренції та прагненням задовольнити потреби цільової аудиторії.

Заклади ресторанного господарства — одні з найскладніших підприємств сервісу. Успішні ресторанні мережі одні із ключових суб’єктів, які зацікавлені в розвитку, а як наслідок — в побудові правильної стратегії, адже це основним чином відображається на їх рейтингу, репутації і т.ін.

Для цього до процесів управління підприємством додають маркетинг, тому що — покликання маркетингу — пояснити, яким чином відбуваються процеси обміну між організацією та клієнтами, і дати рекомендації, як слід будувати відносини “обміну” [1].

Для правильного проектування маркетингової стратегії маркетологи вирішують певні задачі. Існує декілька основних завдань ресторанного маркетингу [2]:

- інформування відвідувачів про ресторан;
- залучення цільових груп клієнтів;
- розширення кола відвідувачів ресторану;
- утримання клієнтів;
- збільшення доходу з клієнта.

Безсумнівно, кожна із задач являється життєво важливою. Але на етапі, коли заклад чи мережа ресторанного господарства вже побудована та функціонує, а показники прибутковості вказують на певні проблеми, перш за все слід локалізувати проблему і це варто робити із порівняння поточної цільової аудиторії та тієї, яка планувалася на етапі розробки концепції. Адже, аудиторія закладу грає чи не найважливішу роль для підприємств сфери надання послуг. Для вирішення однієї із складових цієї задачі, а саме — синтез та аналіз поточної цільової аудиторії, пропонується автоматизувати класифікацію відвідувачів за допомогою кластерного аналізу.

2. Цільова аудиторія та її зв’язок із маркетинговими стратегіями

На стадії розробки концепції визначаються параметри закладу, здатні залучити цільову групу, а також ті, що будуть відсікати небажану публіку. Ці поняття взаємопов'язані — чітке позиціонування закладу залучає певних відвідувачів, автоматично відсікаючи інших [3].

На думку експерта у сфері ресторанного господарства, Олега Назарова, існує проблема, коли здійснюється орієнтування на публіку, яку хоче бачити ресторатор, натомість, приходить інша публіка, яку зацікавив формат закладу. Окрім цього, неграмотна реклама також може вплинути на відхилення публіки від визначеної цільової аудиторії, наприклад, привабити в

ресторан небажану публіку, що може спричинити за собою відтік постійних клієнтів.

Очевидно, що для побудови та втілення правильної маркетингової стратегії варто уникати подібних ситуацій. Щоб мінімізувати такі ризики, необхідно володіти інструментом, що дозволить якісно здійснювати моніторинг груп відвідувачів.

3. Групи відвідувачів ресторану

Для того, щоб здійснити розподіл на групи відвідувачів ресторану потрібно визначитись із портретом клієнта. При цьому, в сфері маркетингу використовують такі набори змінних:

- Географічний — географічні змінні відносяться до місцеположення і включають в себе частину світу, країну, тип населеного пункту, клімат, якщо це здійснюється аналіз на основі даних мережі. Або більш локальні, якщо це окремий заклад.
- Демографічний — змінні в основному описують персональні дані, такі як дохід вік, стать, освіта, посада, етнічна належність, релігія, національність, мова, раса, розмір сім'ї.
- Психографічний — змінні описують складнішу інформацію — спосіб життя, цінності, особистість, відношення.

На основі таких даних синтезують групи відвідувачів ресторану.

4. Збір інформації

Канонічні методи збору пропонують здійснювати збір інформації про відвідувачів [4]:

- Попросіть співробітників, щоб вони дали опис відвідувачів і розповіді — що їм сподобалося, а що ні;
- Використовуйте персонал в якості "слухачем";
- Подивіться свої записи, щоб виявити тенденції;
- Проводьте опитування гостей на виході або інші дослідження.

Однією із ключових задач маркетингу, в цілому, є створення інформаційної бази про клієнтів і роботи з цією базою [1].

Сучасний напрямок розвитку підприємств у сфері надання послуг націлений на всесторонню “смартфонізацію” бізнес-процесів, зокрема, взаємодії із клієнтами. Не є виключенням і підприємства у ресторанному господарстві. Такий стан справ надає можливість ефективніше збирати інформацію про відвідувачів закладу чи мережі, чим власне і користуються найбільш популярні ресторани.

Задачу синтезу цільових груп клієнтів та їх аналізу вирішуємо за допомогою кластерного аналізу.

5. Кластерний аналіз

Кластерний аналіз з'явився порівняно недавно – у 1939 р. Його запропонував вчений К. Тріон. Дослівно термін "кластер" в перекладі з англійської "cluster" означає гроно, згусток, пучок, група.

Особливо бурхливий розвиток кластерного аналізу відбувся у 60-х роках минулого століття. Передумовами якого були поява швидкісних комп'ютерів та визнання класифікацій фундаментальним методом наукових досліджень.

Кластерний аналіз – це метод багатомірного статистичного дослідження, до якого належать збір даних, що містять інформацію про вибіркові об'єкти, та упорядкування їх в порівняно однорідні, схожі між собою групи.

Отже, сутність кластерного аналізу полягає у здійсненні класифікації об'єктів дослідження за допомогою численних обчислювальних процедур. В результаті цього утворюються "кластери" або групи дуже схожих об'єктів. На відміну від інших методів, цей вид аналізу дає можливість класифікувати об'єкти не за однією ознакою, а за декількома одночасно. Для цього вводяться відповідні показники, що характеризують певну міру близькості за всіма класифікаційними параметрами.

Мета кластерного аналізу полягає в пошуку наявних структур, що виражається в утворенні груп схожих між собою об'єктів – кластерів. Водночас його дія полягає й у привнесенні структури в досліджувані об'єкти. Це означає, що методи кластеризації необхідні для виявлення структури в даних, яку нелегко знайти при візуальному обстеженні або за допомогою експертів.

Основними завданнями кластерного аналізу є:

- розробка типології або класифікації досліджуваних об'єктів;
- дослідження та визначення прийнятних концептуальних схем групування об'єктів;
- висунення гіпотез на підставі результатів дослідження даних;

перевірка гіпотез чи справді типи (групи), які були виділені певним чином, мають місце в наявних даних.

Кластерний аналіз потребує здійснення таких послідовних кроків:

- Проведення вибірки об'єктів для кластеризації;
- Визначення множини ознак, за якими будуть оцінюватися відібрані об'єкти;
- Оцінка міри подібності об'єктів;
- Застосування кластерного аналізу для створення груп подібних об'єктів;
- Перевірка достовірності результатів кластерного рішення.

Кожен з цих кроків відіграє значну роль у практичному здійсненні аналізу.

Кластерний аналіз — розбиття заданої вибірки об'єктів (ситуацій) на підмножини, що називаються кластерами, так, щоб кожен кластер складався з схожих об'єктів, а об'єкти різних кластерів істотно відрізнялися. Завдання кластеризації відноситься до статистичної обробки, а також до широкого класу завдань навчання без вчителя.

Для розбиття груп доцільно використовувати метод ієрархічної кластеризації. Цей метод дозволяє нехтувати експерту кількістю кластерів. Точніше, кількість кластерів початково визначена кількістю об’єктів, над якими здійснюється аналіз. Після побудови дендрограми кількість кластерів визначається експертом.

Базовий алгоритм ієрархічної кластеризації виглядає наступним чином [5]:

1. форматувати безліч кластерів одноелементними кластерами
2. $C := \{x_1\}, \{x_2\}, \ldots, \{x_m\}$
3. розрахувати початкову матрицю відстаней R
4. знайти пари кластерів P з C с найменшими відстанями R один до одного
5. об'єднати у великій кількості кластерів C знайдені пари близьких кластерів P
6. розрахувати нову матрицю відстаней R
7. якщо кількість кластерів | C | > 1
8. то перехід на п. 3
9. кінець

Для регулювання глибини дендрограми (дерева історію об'єднань кластерів) і поліпшення результатів кластеризації можна ввести поріг відстаней δ . Алгоритм модифікується в такий спосіб.

1. форматувати безліч кластерів одноелементними кластерами
2. $C := \{x_1\}, \{x_2\}, \ldots, \{x_m\}$
3. і значення порога d : = δ
4. розрахувати початкову матрицю відстаней R
5. знайти пари кластерів P з C с відстанями R <d один до одного
6. якщо пари не знайдені | P | < 1
7. то збільшити поріг d : = min (R) + δ і перехід на п. 3
8. об'єднати у великій кількості кластерів C знайдені пари близьких кластерів P
9. розрахувати нову матрицю відстаней R
10. якщо кількість кластерів | C | > 1
11. то перехід на п. 3
12. кінець

В результаті роботи алгоритму будуть отримані кластери відвідувачів за їх “купівельною спроможністю”. Після цього, з допомогою маркетологів здійснюється аналіз груп, тобто, кожного кластеру. Для кожного кластеру по вживаності продуктів клієнтами можна прогнозувати, що новий клієнт зі схожими демографічними, психографічними та географічними параметрами портрету відноситься саме до цієї групи, і його вподобання будуть такими як і у всієї групи.

На основі цієї інформації для кластерів та нових користувачів можна генерувати персоналізовані пропозиції.

Список літератури

1. Навіщо потрібен маркетинг вашому бізнесу [Електронний ресурс] : Настол руководителю – 2017. – Режим доступу: http://www.nastol.ru/Go/ViewArticle?id=3990
2. І. О. Бухаров, А. Ю. Гридина Ресторанний менеджмент // СПб.: Бонниер Бизнес Пресс. – 2009. – С. 122.
3. І. О. Бухаров, А. Ю. Гридина Ресторанний менеджмент // СПб.: Бонниер Бизнес Пресс. – 2009. – С. 123.
4. Патті Д. Шок, Джон Т. Боуен, Джон М. Стефанеллі – Маркетинг в ресторанному бізнесі // Ресторанные ведомости. – 2005. – С. 53.

5. Метод ієрархічної кластеризації [Електронний ресурс] : Метод иерархической кластеризации. Евгений Борисов – 2014. – Режим доступу: http://mechanoid.kiev.ua/ml-lnwl.html

UDC 004.023

VADYM OVCHARENKO
VLADYSLAV SARNATSKYI
ARTEM SISETSKYI

CODE-REVIEWING NEURAL NETWORK

У цій статті ми описуємо реалізацію нейронної мережі для якості коди скорингу, який ми представили в AI Spring Hackathon 2017. Мета мережі полягає в оцінці ймовірності зразка коду, щоб бути якісними. Ми розробили нашу модель, використовуючи архітектуру, запропоновану для завдання аналізу тексту настрою [1]. Дані навчальні були обрані з відкритих джерел, таких як Github і StackOverflow. Покажу, що мережа може виконувати дещо краще, ніж в базовому прогнозі і передбачає подальше дослідження в цій області.

In this paper, we describe the implementation of Neural Network for code quality scoring, which we presented at the AI Spring Hackathon 2017. The purpose of the network is to evaluate the probability of the code sample to be qualitative. We designed our model using the architecture proposed for the text sentiment analysis task [1]. Training data were selected from the open sources like Github and StackOverflow. We show that the network can perform slightly better than the baseline prediction and suggest consequent research in the area.

1. INTRODUCTION

Automatic code review is a relatively new problem. It comes from the software development industry. Both software enterprises and their clients are interested in evaluating the long-term stability and maintainability of the software. Companies are using a large stack of technologies and manual code reviews to maintain code quality. Their clients mostly do not understand the software development and therefore often involve third-party specialists to evaluate the quality of the product they are buying. An automatic code-reviewing system would be beneficial for both parties.

Prior approaches to the problem were mostly algorithmic, based on static syntax analysis [2, 3]. The code quality scoring was not previously viewed as a Machine Learning task. In this work, we suggest a novel approach to the problem and develop a baseline model for the future research. In the following section, we describe the methods of data scraping and feature extraction that we used. In the third chapter, we address data problems that we encountered during training. Further, we evaluate the model and describe several experiments. In the end, we conclude our work and suggest directions for future research.

2. DATA

The problem requires a significant amount of data. We had to find labeled samples of "poor" and "good" code with additional metadata. We decided to use the following web resources.

1. StackOverflow
2. GitHub
3. Govnokod.ru

These resources contain files or code samples uploaded by users. Also, these resources provide additional metadata for code samples, such as the rating (the number of users who found this code useful or valid, or the difference between those who voted "for" or "against"), the language of code, the date of creation, author's rating. Not all of the resources have a convenient API. For that reason, we used a script to parse data directly from web pages.

2.1. StackOverflow

Stackoverflow is a question-answer resource, where users often attach code samples to their posts. Currently, StackOverflow counts 14 million questions and 22 million answers [4]. Each answer is rated by other users. Our task was to get posts with code samples, extract the code and metadata. We treated rating of the post in which the code sample occurred as the code rating. We normalized both code and author's ratings and labeled code samples which had the product of these two metrics higher than the mean value as "good" code. Samples with negative rating were labeled as "poor"; others filtered out.

2.2. Github

Github is the largest repository of code on the Internet. Our script was randomly downloading code samples from Github Gist where each sample has a number of "stars" set by users who found it useful. We normalized the number of stars and labeled samples above mean as "good" code. We did not use Github as a source of "poor" code samples since it does not provide a possibility to downvote.

2.3. Govnokod.ru

A resource where users post samples of code which they find funny and useless. Others can "like" or "dislike" a sample. We labeled samples with the number of likes above the mean as "poor" code.

Finally, we broke large code samples into sequences of length 300 and padded the rest. One obvious problem with the data was its high dimensionality. Some GitHub Gists included characters with binary codes over 65k. We decided to remove files with Chinese and Russian comments, and for the others, we filtered out all the characters with bytecode over 128. After the filtering, we had around 17k code samples. We split it evenly and normalized the input data.

3. TRAINING

We chose a model suggested by Hong et. al [1] which contains LSTM [5] layer fed by trained word embeddings. We implemented the model in TensorFlow framework and trained it on Nvidia Tesla (K-40) GPU with the batch size set to 128. We soon realized that the model learns too fast, and the validation accuracy is too high. Character distribution plot (see Figure 1) shows that there were some serious discrepancies between character distribution in both datasets. For example, carriage return symbol (13) did not appear at the dataset with "good" code at all. Such dominating features made it easy for the model to learn. After smoothing the distribution breakouts, we trained the model again. The ultimate precision of the model was around 51% on cross-validation. Comparing to the baseline of random selection, this is not a high accuracy.

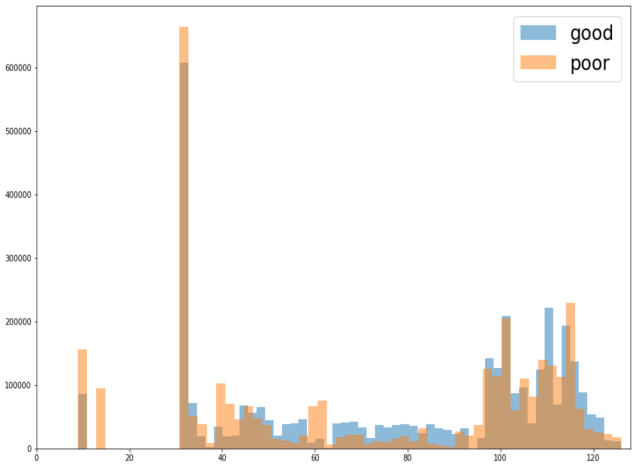


Figure 1. Character distribution in dataset.

4. EXPERIMENTS

We launched an online web-service for the hackathon participants who wanted to predict the quality score on pieces of their code. The web server written in Python ran on the machine with Nvidia GTX-970 GPU. We also collected GitHub accounts from the participants and demonstrated standings in our final presentation (Figure 2).

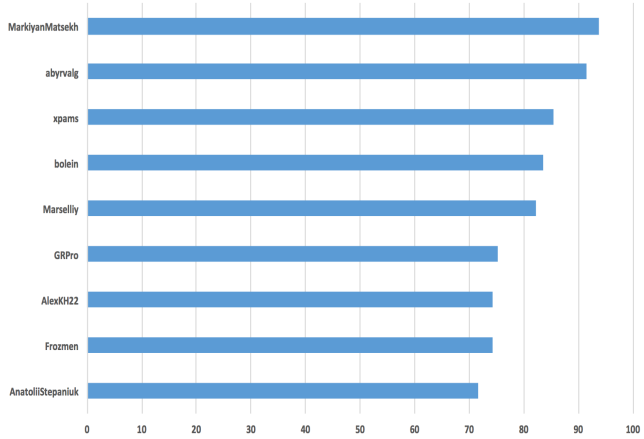


Figure 2. Standings of the hackathon participants who shared Github profiles. Score calculated as the average of the scores on each file.

5. CONCLUSION

Due to the strict time constraints, we were not able to experiment with the architecture of the model. Nevertheless, mentors proposed using an attention model [6] for learning complex features of the code. The other suggestion is to use stateful RNNs for batches of the same piece of code. Finally, to achieve a better result, it is recommended to elaborate on the dimensionality reduction of the input by extracting features from the code samples instead of simply feeding the lexeme sequences. We were not able to narrow training dataset to at least one language due to the deficiency of data, but it is strongly advised.

The code quality evaluation task is a difficult problem because of the subjectivity. A syntactically and lexically correct piece of code is not always a good one, even if it is written adhering to the community code conventions. Currently, only a professional can evaluate the real quality of code. However, this is going to change with the advance of Artificial Intelligence.

REFERENCES

1. Hong, J. & Fang, M. (2015), 'Sentiment Analysis with Deeply Learned Distributed Representations of Variable Length Texts'.
2. Merlo, E.; McAdam, I. & de Mori, R. (2003), 'Feed-forward and recurrent neural networks for source code informal information analysis.', Journal of Software Maintenance 15 (4), 205-244.
3. Goues, C. L. & Weimer, W. (2012), 'Measuring Code Quality to Improve Specification Mining.', IEEE Trans. Software Eng. 38 (1), 175-190.
4. http://data.stackexchange.com/. Web Page.
5. Hochreiter, S. & Schmidhuber, J. (1997), 'Long Short-Term Memory', 9 (8), 1735--1780.
6. Rocktäschel, T.; Grefenstette, E.; Hermann, K. M.; Kociský, T. & Blunsom, P. (2015), 'Reasoning about Entailment with Neural Attention.', CoRR abs/1509.06664.

УДК 004.75

ОНІЩУК А.А.
ЛІЩУК К.І

ПОБУДОВА ВИСОКОНАДІЙНОЇ РОЗПОДІЛЕНОЇ ФАЙЛОВОЇ СИСТЕМИ ДЛЯ СІМЕЙСТВА ПЛАНУВАЛЬНИКІВ ЗАДАЧ MAUI

У роботі розглянуто спосіб впровадження у планувальник задач для GRID систем MAUI концепції розподілених файлових систем для вирішення проблем низької масштабованості сховища даних, неможливості надійного зберігання даних та безперебійного доступу до них. Запропоновано архітектуру РФС яка є підходящою під GRID системи під керівництвом MAUI, а також інші GRID системи, що мають однорідну структуру планувальника задач.

Ключові слова: РОЗПОДІЛЕНА ФАЙЛОВА СИСТЕМА, ПЛАНУВАЛЬНИК ЗАДАЧ MAUI, МАСШТАБОВАНІСТЬ ФАЙЛОВИХ СИСТЕМ, ВИСОКА НАДІЙНІСТЬ ФАЙЛОВИХ СИСТЕМ

This article describes a way to implement a distributed file system for MAUI job scheduler, which solves the problems of low scalability and non-reliability of data storage, as well as a problem of problem of data inaccessibility. The architecture which is suitable for MAUI GRID systems is suggested as well as for other GRID systems, which have a uniform structure of job scheduler.

Keywords: DISTRIBUTED FILE SYSTEM, MAUI PROBLEMS SCHEDULER, FILE SYSTEM SCALABILITY, HIGH RELIABILITY FILE SYSTEM

1. Вступ

В зв’язку із швидким збільшенням популярності розподілених систем через їх більшу надійність, масштабованість та потужність різко зросла потреба в простому і зручному ПЗ, що спрощувало б роботу користувачів таких систем.

На даний момент користувачу GRID системи для того, щоб запустити задачу, яка працює із файлами у планувальнику Maui потрібно вручну копіювати файли на кожен із вузлів у кластері, що збільшує ймовірність помилок. А також має ряд інших недоліків у порівнянні із використанням концепції розподіленої файлової системи для організації роботи із файлами. Перевагами використанням такого концепту є масштабованість, надійність зберігання даних, надійність доступу до даних, а також дешевизна обладнання для зберігання файлів.

2. Архітектура розподіленої файлової системи для Maui

Розподілені файлові системи (РФС) – це файлові системи у яких файлові частини (у більшості ФРС блоки) зберігаються на різних носіях з`єднаних високошвидкісною мережею [1]. Система, що описується, для Maui планувальника є підвидом РФС.

У системі є два основні типи вузлів: вузол імен і вузол даних. А два допоміжні типи вузлів: журнальний вузол та резервний вузол імен (будуть розглянуті у наступному розділі). На рис.1 показана архітектура розміщення та взаємодії даних вузлів.

Вузол імен є центральним вузлом РФС. Існує лише один активний вузол імен у РФС. На ньому зберігаються метадані файлової системи, а також інформація про те, де в кластері зберігаються дані файлу. До метаданих належать: імена файлів, їх типи, права доступу, дані розміщення блоків у межах мережі. Вузол імен не зберігає ніяких файлових даних. Це зроблено для того, щоб знизити навантаження на нього. Адже при більшості файлових операцій перший, а іноді і єдиний виклик здійснюється до вузла імен.

Виключенням є операція запису у файл, яка потребує координації між усіма типами вузлів файлової системи.

Задачі MAUI щоразу звертаються до вузла імен коли вони хочуть знайти файл, або додати / копіювати / перемістити його. При читанні файлів вузол імен відповідає на запити поверненням списку відповідних вузлів імен де зберігаються блоки цих даних.

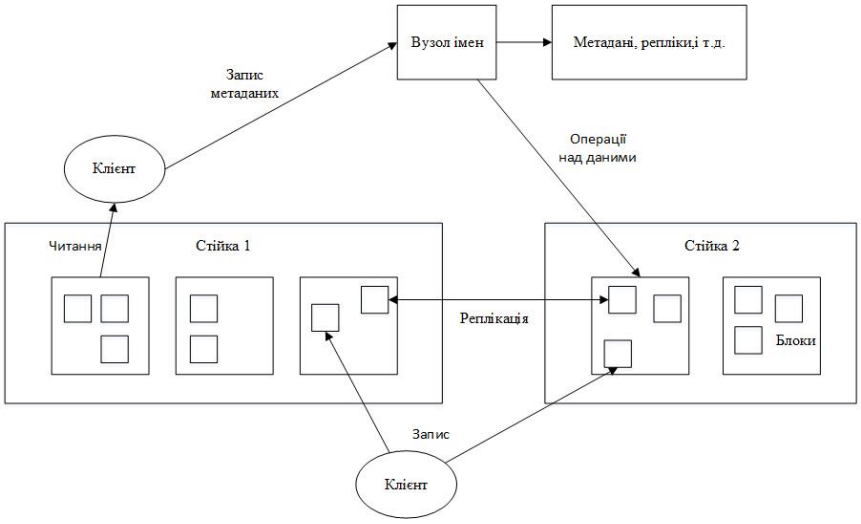


Рис. 1. Архітектура РФС

Вузол даних – це клієнтський вузол, основним призначення якого є збереження блоків даних. Для використання переваг РФС у ній має бути більше ніж один вузол даних. Кожен такий вузол знає про блоки, що на ньому знаходяться, доступ до яких користувачеві дає вузол імен за допомогою спеціального посилання, після відповідного запиту. Крім цього, вузли даних вміють робити реплікацію файлових блоків, для підвищення надійності системи. Також для таких вузлів зазвичай не потрібно використання RAID дисків, адже існує реплікація між вузлами.

Для пришвидшення доступу до файлових блоків, вузли імен кешують блоки, які найчастіше використовуються. Тому збільшення кількості RAM на вузлах даних хоча й може призвести до дещо швидшої роботи, проте не є критичним.

Ще одною частиною файлової системи є РФС клієнт. РФС клієнт це програмна бібліотека, яка дозволяє працювати із файловою системою за допомогою простих команд unix інтерфейсу. Дозволені такі команди як: ls, rm, mkdir, touch, а також деякі допоміжні команди притаманні розподіленим файловим системам: copyFromLocal, copyToLocal. Ці команди дозволяють копіювати файли чи папки із локальної файлової системи у РФС і навпаки. Ще один РФС клієнт – MAUI РФС клієнт інтегрується з API інтерфейсом MAUI, та використовується ним для проведення файлових операцій через вбудований MAUI інтерфейс [2].

Розглянемо на прикладі основних файлових операції взаємодію даних вузлів. Найбільш складною операцією з точки зору взаємодії компонент РФС є операція запису у файл (WRITE). Вона складається із чотирьох кроків описаних нижче:

1. Виклик до вузла імен для отримання списку вузлів даних між якими потрібно розподілити блоки файлу.
2. Запис файлових блоків до вузлів даних.
3. Реплікація вузлами даних отриманих блоків.
4. Надання вузлу імен інформації про розміщення записаних блоків

Операція редагування атрибутів файлу(CHANGEATTR) таких як прав, імені, розміщення обмежується одним зверненням до вузла імен. Теж саме

стосується й операцій видалення файлу (RMFILE), створення папки (MKDIR) або пустого файлу (TOUCH). Операція копіювання файлу реалізовується через серію доручень від вузла імен до вузлів даних, для копіювання блоків файлу.

Роль вузла імен та вузла даних як правило виконує типічний комп`ютер економ класу [3]. Це зумовлюється наявністю реплікацією даних і високою надійністю. Тобто, при відмові одного із комп'ютерів, що входить у РФС, не відбувається втрата даних, чи відмова у обслуговуванні. Це сильно здешевлює початкову ціну файлової системи, так як вона може складатись всього із кількох комп'ютерів економ класу і при цьому надає значну масштабованість, через те, що для збільшення потужностей достатньо додати нові вузли у РФС. На відміну від традиційний хранилищ даних, де для цього необхідна заміна обладнання.

Розглянемо концепцію блоку РФС. Звичні нам дискові файлові системи мають розмір блоку, що позначає мінімальну кількістю даних, які він може читати і писати.

Файлові системи для одного диска будуються на цьому принципі роботи з розміром даних в блоку, який кратний розміру блоку на диску. Блоки такий типічний файлових систем, як правило займають кілька кілобайт, в той час як дискові блоки, як правило, 512 байт [4].

РФС теж має концепцію блоку, але це набагато більше, блок має мати значний розмір наприклад 128 МБ.

Блоки РФС такі великі в порівнянні з дисковими блоками, і причина полягає в мінімізації витрат на пошук. Якщо блок досить великий, то час, необхідний для передачі даних з диска може бути значно більшим, ніж час, щоб звернутися до початку блоку. Таким чином, передача великого файлу з декількох блоків, працює зі швидкістю передачі диска.

Швидкий розрахунок показує, що якщо час пошуку складає близько 10 мс, а швидкість передачі 100 МБ / с, то щоб зробити час пошуку 1% від часу передачі, нам потрібно, щоб розмір блоку був близько 100 МБ.

3. Організація високої надійності для розподіленої файлової системи

Одною із головних переваг розподіленої файлової системи є можливість організації високої надійності зберігання файлів та доступу до них. Для того, щоб досягти цього для початку розглянемо проблеми, що можуть завадити доступу до даних або пошкодити самі дані:

- апаратний чи програмний збій вузла імен
- апаратний чи програмний збій одного або кількох вузлів даних
- недоступність даних через неполадки в мережі

Апаратний чи програмний збій вузла даних вирішується за допомогою реплікації файлових блоків між цими вузлами. Реплікацією у даному сенсі є надлишкове копіювання блоків даних між вузлами даних. Число вузлів на яких зберігаються блоки даних називається фактором реплікації. За допомогою налаштування високого фактора реплікації, а також налаштування реплікації у різні сегменти мережі можна добитись безперервного доступу до файлів у випадках відмови кількох вузлів одразу, або навіть цілих сегментів мережі.

Для того щоб могти забезпечити безперебійну роботу MAUI GRID системи при збої вузла імен потрібне запровадження двох допоміжних типів вузлів: журнального вузла, а також резервного вузла імен.

Дві окремі машини налаштовуються як вузли імен. У будь-який момент часу, рівно один з вузлів імен знаходиться в активному стані, а інший знаходиться в стані очікування (резервний вузол імен). Активний вузол імен відповідає за всі клієнтські операції в кластері, в той час як резервний є в режим очікування та не використовується, при цьому він зберігає достатньо інформації про стан файлової системи, щоб забезпечити швидкий перехід на нього, якщо це необхідно.

Для того, щоб резервний вузол імен був синхронізованим з активним вузлом, обидва вузла зв'язуються з групою окремих вузлів, так званих, журнальних вузлів. Коли будь-яка зміна імен виконуються активним вузлом, він реєструє запис про модифікацію для більшості журнальних вузлів. Резервний вузол імен здатний читати правки з журнальних вузлів і постійно спостерігає за змінами у їхньому журнал. Як резервний вузол бачить зміни, він застосовує їх до свого власного простору імен. У разі відмови активного вузла імен, резервний, після читання всіх правок з журналу імен, оголошує себе активним. Це гарантує, що стан простору імен повністю синхронізований, перш ніж відбувається перехід на інший вузол.

Для того, щоб забезпечити швидкий перехід простору імен на інший ресурс необхідно, щоб резервний вузол імен мав останню актуальну інформацію про місцезнаходження блоків в кластері. З цією цілю, вузли даних передають інформацію про місцезнаходження блоків, а також поточний свій статус обом вузлам імен.

Дуже важливим для правильної роботи такої системи, є той факт що тільки один з вузлів імен має бути активними в кожен момент часу. В іншому випадку стан простору імен швидко розходяться між ними, що приводить до втрати даних або інших невірних результатів. Для того, щоб забезпечити цю властивість і запобігти так званому «сценарій розколу мозку», журнальні вузли за весь час головування вузла імен дозволяють запис подій лише йому. Під час переходу на інший ресурс простору імен, новий активний вузол імен просто бере на себе обов'язок запису в журнальні вузли, що буде ефективно запобігати іншому вузлу імен продовжувати бути в активному стані.

Недоступність ж вузлів через неполадки мережі може бути легко усунена за допомогою організації надлишкової топології мережі.

В основі концепції надлишкової топології мережі лежить потреба забезпечення альтернативних шляхів для даних, щоб переміщатися в разі, якщо кабель пошкоджений або роз'їм не підключений [5]. Проте, Ethernet як стандарт, не допускає кільця або петлі в мережі, так як це викликає широкомовні цикли і в кінцевому підсумку призводить до того що мережа призупиняє роботу. Мережа Ethernet не може мати два шляхи з точки А в точку Б без механізму для підтримки цього типу топології. Для забезпечення надмірності, мережева інфраструктура (комутатори) повинні підтримувати протоколи резервування, покликані звести нанівець проблему введення петлі в мережу Ethernet. Це досягається шляхом зберігаючи кількох альтернативних шляхів на комутаторі: шляху до даних за замовчуванням і альтернативного шляху, які він перемикає коли виникає потреба.

4. Висновок

У даній роботі було описано архітектуру розподіленої файлової системи для MAUI, яка дозволяє досягнути більшої масштабованості, а також досягнути високої надійності зберігання даних, та їх доступності.

Список літератури

1. David H. M., Building Linux Clusters. – New Hampshire: O'Reilly Media, 2007. -360p
2. Jayson D., Maui Administrator's Guide. – London: Adaptive Computing, 2012. -287p
3. Distributed File Systems: Concepts and Examples – Texas: ACM Computing Surveys, 2005 - 321p.
4. Dominic Giampaolo, Practical File System Design - 2nd Edition – Cologne: Morgan Kaufmann, - 2005. -256p.
5. A. Tanenbaum, Computer Networks - 5th edition.pdf - Washington: Pearson – 2011. – 962p.

УДК 004.023

КАТЮЩЕНКО Д.О.,
ОЛІЙНИК Ю.О.

ДОСЛІДЖЕННЯ МЕТОДУ МАКСИМУМА ЕНТРОПІЇ ДЛЯ КЛАСИФІКАЦІЇ ТЕКСТІВ НА ПРИКЛАДІ APACHE OPENNLP DOCCAT

В даній статті розглянуто вирішення задачі класифікації за допомогою методу максимуму ентропії. Вона складається з двох частин: теоретичної та практичної. В теоретичній описані основні принципи та загальний алгоритм методу максимуму ентропії. В практичній представлений експериментальний аналіз ефективності даного методу, реалізованого в Apache OpenNLP Doccat.

In the present article was considered classification issue solving by using maximum entropy method. The article consists of two parts: theoretical and practical. In the theoretical part were described basic principles and general algorithm of maximum entropy method. In the practical part was investigated efficiency of the current method in the terms of implemented one in toolkit Apache OpenNLP Doccat.

1. Вступ

В даній статті розглянуто вирішення задачі класифікації за допомогою методу максимуму ентропії. Во на складається з двох частин: теоретичної та практичної. В теоретичній описані основні принципи та загальний алгоритм методу максимуму ентропії. В практичній представлений експериментальний аналіз ефективності даного методу, реалізованого в Apache OpenNLP Doccat.

2. Загальний опис класифікації методом максимума ентропії

Інформаційна ентропія це міра невизначеності ймовірнісного розподілу. Іншими словами ентропія показує наскільки тяжко робити передбачення про події, які описані імовірнісним розподілом. Чим більша ентропія, тим важче робити передбачення. Одна з головних особливостей ентропії полягає в наступному: чим більше ентропія, тим розподіл рівномірніший.

Здавалося б ентропію навпаки потрібно мінімізувати, але, при вирішенні задач класифікації, серед всіх розподілів, які відповідають імперичним даним необхідно вибирати, той який володіє найбільшою рівномірністю (тобто і з максимальною ентропією). В іншому випадку, відбувається вибір розподілу з інформацією, підтвердження якої немає в навчальній вибірці.

Метод максимуму ентропії (maximum entropy algorithm, MaxEnt) будує класифікаційну модель спираючись на ту саму інформацію, що й спрощений алгоритм Байєса: ці обидва алгоритми оперують таблицями ваг, які отримуються під час навчання з частотності класифікаційних ознак в межах класу, - по одному параметру на кожну пару класифікаційна ознака + клас. Тобто всього таких ваг n*k, де n – кількість класів, а k – кількість класифікаційних ознак.

Опишемо етап класифікації більш детально: нехай маємо n класів $(c_1, c_2, ..., c_n)$ та k класифікаційних ознак $F^1, F^2, ..., F^k$. Класифікаційна ознака – це бінарна функцій над документом, наприклад:

F = func(t) { t.contains (“економіка”)}

На основі класифікаційних ознак генеруються класифікаційні індикатори:

$$f_1, f_2, \dots, f_{k*n} \qquad (1)$$

Класифікаційний індикатор – це булєва функція над парою документ+клас. Вона істинна лише тоді, коли істинна відповідна класифікаційна ознака та співпадає класЮ що передається.

Сама класифікація виконується за формулою:

$$p\,\langle c|d,\gamma\rangle = \frac{\exp\sum_i^{n\times k}\gamma_i f_i(c,d)}{\sum_{\tilde{c}\in C}\exp\sum_i^{n\times k}\gamma_i f_i(\tilde{c},d)}, \qquad (2)$$

де f_i – і-й класифікаційний індикатор (0 або 1),
γ_i – вага і-го класифікаційного індикатора f_i,
c – клас-гіпотеза,
C – множина всіх можливих класів,
d – документ, що класифікується.

Таким чином результатом відношення (2) є значення імовірності для кожного класу.

Відмінність же MaxEnt від алгоритму Байєса полягає в підході до пошуку ваг. Якщо спрощений алгоритм Байєса не враховує можливі залежності, то MaxEnt використовує поліноміальну логістичну регресію, щоб визначити категорію передаваємого тексту.

Поліноміальна логістична регресія – це загальний випадок моделі логістичної регресії, в якій залежна змінна має більше 2-х категорій.

В поліноміальній логістичній регресії для кожної категорії залежних змінних будується рівняння бінарної логістичної регресії. При чому одна з категорій залежних змінних стає базовою, та всі інші категорії порівнюються з нею. Рівняння поліноміальної логістичної регресії прогнозує імовірність приналежності залежної змінної за значенням незалежних змінних. (знаходження незалежних імовірностей було представлено вище).

Тобто щоб знайти оптимальні ваги для формули класифікації (2) необхідно знайти набір параметрів, за яких досягається максимум наступної функції:

$$\log \mathrm{p}(C|D,\gamma) = \sum_{(c,d)\epsilon(C,D)} log\, p\,\langle c|d,\gamma\rangle = log\,\frac{\exp\sum_i^{n\times k}\gamma_i f_i(c,d)}{\sum_{\tilde{c}\in C}\exp\sum_i^{n\times k}\gamma_i f_i(\tilde{c},d)} \qquad (3)$$

Зазвичай максимум такої задачі знаходять за допомогою градієнтного спуску.

3. Результаты експеременталього дослідження MaxEnt у OpenNLP DocCat

Бібліотека Apache OpenNLP - машинний інструментарій для обробки текстів на природній мові. Document Categorizer (Doccat) – один з інструментів цієї бібліотеки, який виконує класифікацію текстів за заданими наперед категоріями Він заснований на описаному вище методі - MaxEnt.

Для класифікації деякого набору текстів за допомогою цього інструменту необхідно побудувати модель через CLI можна за допомогою наступної команди:

«opennlp DoccatTrainer -model en-doccat2.bin -lang en -data text_en_5.train -encoding UTF-8»

де DoccatTrainer - назва інструменту, який буде використано;
en-doccat2.bin - назва файлу, в який буде записана модель;
en – мова текстів - англійська;
ext_en_5.train - файл з навчальною вибіркою;
UTF-8 - кодування файлу.

Для коректної побудови моделі, кожен рядок навчальної вибірки має починатися з категорії, до якої віднесено текст. Після категорії має слідувати текст та всього його необхідно розмістити в один рядок. Розмежувати категорію та текст необхідно пробілом.

В даному експериментальному дослідженні інструменту Doccat була відібрана навчальна вибірка в 9828 рядків та створена модель, командою, яка описана вище. Побудова моделі тривала 1332 секунди (22 хвилини).

Далі ця модель використовувалась в програмі для класифікації 88 текстів між 2-ма різними категоріями. Та було отримано 71 правильно прокласифікований текст, тобто точність класифікації 81%.

4. Висновок

Використання MaxEnt для вирішення задачі класифікації текстів є досить ефективним, адже за прийнятний час дає можливість побудувати модель, яка в переважній більшості випадків дає правильний результат.

Список літератури

1. Василенко Г. И., Тараторин А. М. Восстановление изображений - М: Радио и связь, 1986. - 304 с.
2. Белашев Б. З., Сулейманов М. К. Метод максимума энтропии. Статистическое описание систем // Письма в ЭЧАЯ -2002. – С. 44-50
3. Segaran T. Programming Collective Intelligence// O'Reilly Media – 2007 – P.323
4. Сайт бібліотеки Apache OpenNLP [Електронний ресурс] // Режим доступу: https://opennlp.apache.org

УДК 004.93

ПОЛІЩУК О.В.

ПРОГРАМНО-ІНТЕРАКТИВНИЙ КОМПЛЕКС ДЛЯ ОНЛАЙН КОМУНІКАЦІЇ УЧАСНИКІВ НАДАННЯ ТА ОТРИМАННЯ ПОСЛУГ

Розглянуто актуальність комплексу, його новизну та цільову аудиторію. У роботі здійснений аналіз та обґрунтування вибору рекомендаційної системи, переваги та недоліки, а також вирішення існуючих проблем на реальному прикладі самого комплексу. Наведено основні поняття, що використовуються для розв'язку поставленої задачі та шляхи вирішення проблем.

Ключові слова: ВОЯЖЕР, ТРЕВЕЛЕР, ВИБІР ПРОФЕСІЇ, ОНЛАЙН, КОМУНІКАЦІЯ, ПОСТАЧАЛЬНИКИ ТА СПОЖИВАЧІ, ПОСЛУГИ, РОЗВАГИ, РЕКОМЕНДАЦІЙНІ СИСТЕМИ, ВЕКТОР, МАТРИЦІ, СИНГУЛЯРНИЙ РОЗКЛАД, ГРАДІЄНТ, ОЦІНКА, АЛГОРИТМ, МАШИННЕ НАВЧАННЯ

We consider the relevance of the complex, its novelty and target audience. The article has an analysis and justification of chosen recommender system, its advantages and disadvantages, as well as solving the existing problems of it on a real software product. The basic concepts used to solve this problem and ways of solving problems.

Keywords: VOYAZHER, TREVELER, CAREER CHOICES, ONLINE, COMMUNICATION, SUPPLIERS AND CUSTOMERS, SERVICES, ENTERTAINMENT, RECOMMENDATION SYSTEMS, VECTOR, MATRIX, SINGULAR SCHEDULE GRADIENT ESTIMATION ALGORITHM, MACHINE LEARNING

1. Вступ

Відпочинок, розваги, змінення буденності, пошук нових вражень, відчуттів, жага до пізнання нового, оволодіння нових вмінь, знайомство з іншими людьми, інтерес до їх добробуту, індивідуальності - це далеко не повний список того, що характеризує успішну людину 21-го століття.

2. Аналіз актуальності, цільової аудиторії як підґрунтя успішності проекту.

Внаслідок стрімкого розвитку культури, різних послуг, люди знаходяться у пошуку вдосконалення та варіаціїї свого відпочинку. Так серед тих, хто любить подорожувати все частіше зустрічаються любителі дослідження ментальності, пізнання того, що формує людину. Таких людей називають voyager. Якщо звернутися до словника [1], то traveler - це людина, яка подорожує, особливо на далекі країни. Voyager же з дослівного перекладу, людина-мандрівник, яка подорожує з метою дослідження нових земель, світів. Вояжер любить пізнавати новий світ через буденність місцевих, їм цікаві повсякденні справи, речі, які змушують місцевих приймати ті чи інші рішення, думати у тому чи іншому порядку - ось ті деталі, які цікаві представникам даного напрямку.

Людина, яка мандрує має на меті побачити все в живу, відчути це реально. Вона у пошуках нових відчуттів, пізнання, розвитку.

Найбільше враження з моїх поїздок за кордон було складене від спілкування з людьми, пізнання їх ментальності, повсякденного життя, добробут і оточення, їх методи розваг, ставлення до роботи, релігії, захоплення, мандрівок, людей, расових та культурних вподобань. Люди відрізняються не лише по віку чи статті, а й по багатьом аспектам, які формувалися на основі багатьох факторів, які оточували та/або надалі оточують і ось це намагається пізнати вояжер.

А тепер тільки уявіть можливість ознайомитися з буденністю місцевих людей й поринути у їхні буденні справи, "відчути на власній шкурі" їхнє життя,

професію, відпочинок, спілкування з друзями, колегами, затишна вечеря у сімейному колі, купівля продуктів на тиждень, тощо. Ось те, до чого прагне даний тип нашої цільової категорії - прожити хоча б один день як інша людина. Спробувати на собі, що таке бути місцевих, поринути на цілий день у життя іншої людини, робити ті справи, які не були звичними до того, про які можливо навіть не догадувався учасник даного заходу. Саме такі відчуття дають можливість на повну відчути колорит місцевого населення, обставини й умови, що слугували основою формування людини такою яка вона є.

Розглянемо іншу думку. Що вплинуло на ваш вибір професії, коли ви були школярем? Чи не першим буде у вас у думках відповідь: “позиція батьків, старших членів сім’ї”. З дослідження черкаської молоді, було виявлено, що 25% молоді ґрунтувалися на думці батьків та проінформованості [2].

Незважаючи на сучасний рівень доступу до інтернету, люди довіряють власним знанням, досвіду або рідних та близьким. Так більшість батьків рекомендують ті професії, з якими вони знайомі чи на власному досвіді, чи на досвіді рідні та близькі, які володіють даних фахом та можуть проконсультувати у даному питанні. Як наслідок школяр не має й уявлення, що то за професія, яку їй батьки рекомендують, що для цього потрібно, а про інші види робіт, які знаходяться за межами кругозору рідних варто й говорити. Як наслідок, це призводить до того, що значний відсоток молодих осіб не закінчують навчальний заклад і у кращому випадку переподає документи до іншого закладу на іншу спеціальність. А це вже втрата часу, ресурсів та інший витрат.

Уявіть, що абітурієнт має можливість на один день в живу поспілкуватися з людиною, яка володіє фахом, спробувати власноруч зробити продукт праці, дізнатися, що для цього потрібно й визначитися чи потрібно(цікаво) це йому. Навіть один день може змінити прихильність школяра. Якщо розвивати думку, то можна організувати тижневий курс, тощо - масштабування у цьому випадку буде доречним.

Активна людина завжди в пошуках нового. Це може бути новий вид відпочинку, заняття, досвід, оволодіння новим хобі, спілкування, освоєння нового фаху. Знання ніколи не бувають зайвими, кожен досвід - це досвід і підґрунтя для швидкозмінного майбутнього.

Розширення кругозору можна здійснювати й відпочинком, хтось любить читати книжки, статті, відвідувати гуртки, квест-кімнати. А як щодо активності побути один день іншою людиною? Як розвага, спробувати й провести один день зовсім іншої активності, лише один день без жодних зобов’язань. Така можливість дозволить людині отримати нові враження, зарядитися позитивом, а постачальнику послуг ще й заробити грошей.

Постачальником послуг у нашому випадку є люди, які зацікавлені у спілкування, культурному обміні й володіють бажанням поділитися своїми навиками, показати своє життя іншій людині. Якщо людина горить своєю діяльністю, навиками й із задоволенням поділиться ним, то чому б не допомогти їй у цім. А якщо й ще за це буде винагорода, то людина буде брати участь у такій активності не один раз.

Звичайно вище розглянута інформація може не охоплювати певні виключення й це не дивно, адже скільки людей, скільки й особливостей.

Метою дослідження є підвищення ефективності комунікації для постачальників послуг та тими, хто отримує ці послуги, шляхом розробки онлайн платформи та впровадження інтелектуального аналізу даних про учасників комунікації, вподобань, отриманих послуг та переглянутих сторінок.

Підвищення культурного рівня, обізнаності та оволодіння новими навиками шляхом міжособистістного обміну та розповсюдженням інформації, передачі досвіду, навчання навикам та

проведення інтернаціонального обмін між усіма учасниками активності. Як наслідок, отримання не лише корисних знань, а й емоційне задоволення потреб у спілкуванні та ознайомленні з представниками інших культур.

Метою є поліпшення спілкування старшокласників та людей, чий фах може бути майбутнім фахом сьогоднішніх школярів. Обмін досвідом підвищує обізнаність молодої особи у різних сферах та поглядах на життя, що значною мірою впливає на вибір майбутньої професії. Даний вид активностей допомагає виявити схильність та вподобання, інтерес та бажання оволодіти навиками.

Даний застосунок поліпшує відпочинок, шляхом урізноманітнення доступних активностей для користувачів послуг, передачі навиків та отримання винагороди надавачам послуг, можливість додаткового способу отримання коштів.

на зоровий нерв, і ретранслюється в головний мозок, який відтворює зображення і сприймає його. ССЗ базуються на заміні ушкоджених елементів і продовженні використання існуючих областей головного мозку у процесі сприйняття. Під час сенсорного заміщення зору неушкоджена сенсорна система ретранслює інформацію для сприйняття зоровою частиною мозку так, що людина може отримати відчуття, ніби вона може «бачити». За допомогою сенсорного заміщення інформація, отримана від однієї сенсорної системи, може досягати структур мозку, які фізіологічно пов’язані із іншими. Сенсорне заміщення дотик-зір передає інформацію від рецепторів дотику шкіри до зорової ділянки кори для інтерпретації та сприйняття. Даний факт підтверджений за допомогою функціональної магнітно-резонансної томографії, під час якої було в

3. Рекомендаційна система - залог успішності платформи

Рекомендаційні системи з'явилися в інтернеті досить давно, близько 20 років тому. Однак справжній підйом в цій області трапився приблизно 5-10 років тому, коли відбулося змагання Netflix Prize у 2006. Компанія виклала у відкритому доступі зібрані дані: близько 100 мільйонів оцінок за п'ятибальною шкалою з зазначенням ID користувачів, що їх поставили. Учасники змагання повинні були якомога краще передбачати, яку оцінку поставить певного фільму той чи інший користувач. Якість передбачення вимірювалася за допомогою метрики СКВ (середньо-квадратичне відхилення). У Netflix вже був алгоритм, який передбачав оцінок з якістю 0,9514 за метрикою RMSE. Завдання було поліпшити прогноз хоча б на 10% - до 0.8563. Переможцю був обіцяний приз в $ 1 000 000. Змагання тривало приблизно три роки. За перший рік якість поліпшили на 7%, далі все трохи сповільнилося. Але в кінці дві команди з різницею в 20 хвилин надіслали свої рішення, кожне з яких проходило поріг в 10%, якість у них була однакова з точністю до четвертого знаку. У задачі, над якою безліч команд працювало три роки, все вирішили якихось двадцять хвилин. Італійська команда, яка запізнилася (як і багато інших, які брали участь в конкурсі) залишилися ні з чим, однак сам конкурс дуже сильно прискорив розвиток в цій галузі.

У випадку із вище описаним проектом, маємо множину користувачів $u \in U$, множину послуг $i \in I$ та множину подій $(r_{ui}, u, i, ...) \in \mathcal{D}$ (події, які користувачі виконують із об’єктами). Кожна подія задається користувачем u, об’єктом i та своїм результатом r_{ui} . Поставлені задачі:

- передбачити вподобання

$\hat{r_{ui}} = \text{Predict}(u,i,...) \approx r_{ui}$

- персональні рекомендації

$u \mapsto (i_1,...,i_K) = \text{Recommend}_K(u,...)$

- подібні послуги

$u \mapsto (i_1,...,i_M) = \text{Similar}_M(i)$

Розглянемо колаборативну рекомендаційну системи так як алгоритми цього типу систем мають більш універсальний підхід, який часто дає кращий результат. Враховуючи, що для колаборативних рекомендаційних систем використовується історія оцінок як самого користувача, так й інших користувач, розглянемо найпростіший метод. Для цього потрібно вибрати умовну міру

подібності користувачів по їх історії оцінок $sim(u, v)$. Об'єднаємо користувачів у групи (кластери) так, щоб подібні користувачі були в одному кластері $u \mapsto F(u)$. У такому випадку оцінку користувача послуги можна розрахувати як середню оцінку кластера цій послузі:

$$\hat{r}_{ui} = \frac{1}{|F(u)|} \sum_{v \in F(u)} r_{vi} \sum^{\Sigma}$$

Алгоритм має декілька недоліків, а саме:

- нічого рекомендувати новим користувачам
- не враховується специфіка кожного користувачу
- якщо у кластері ніхто не оцінював цю послугу, ми не можемо отримати оцінку.

Покращити цей алгоритм можна шляхом заміни строгої кластеризації на наступне:

$$\hat{r}_{ui} = \overline{r_u} + \frac{\sum_{v \in U_i} sim(u, v)\,(r_{vi} - \overline{r_v})}{\sum_{v \in U_i} sim(u, v)}$$

Ідея базується на користувачах. Аналогічно ми можемо зробити симетрично, базуючись на послугах:

$$\hat{r}_{ui} = \overline{r_i} + \frac{\sum_{j \in I_u} sim(i, j)\,(r_{uj} - \overline{r_j})}{\sum_{j \in I_u} sim(i, j)}$$

Недоліками є:

- не якісне передбачення для нових користувачів/послугах
- тривіальність рекомендацій
- ресурсоємкість обрахунків
- холодний старт

Розглянемо алгоритм Singular Value Decomposition (SVD) - Сингуля́рний ро́зклад ма́триці (сингулярне представлення матриці чи SVD) — один з важливих методів розкладу (чи діагоналізації) матриці, що застосовується в лінійній алгебрі для обчислення псевдоінверсії, наближення матриці, обчислення рангу матриці та інше. За яким будь-яка матриця A розмірності $n \times m$ існує розкладення у добуток трьох матриць U, Σ и V^T [3].

$$A_{n \times m} = U_{n \times n} \times \Sigma_{n \times m} \times V^T_{m \times m}$$

Матриці U та V ортогональні, а Σ — діагональна.

$$UU^T = I_n, VV^T = I_m$$

$$\Sigma = diag\left(\lambda_1, \dots, \lambda_{\min(n,m)}\right), \lambda_1 \geq .. \geq \lambda_{\min\,(n,m)}$$

Існує також урізаний розклад, коли з лямбд залишаються лише перші d чисел, а інші рівні нулю. Це рівносильно тому, що у матрицях U та V ми залишимо перших d стовпчиків, а матрицю Σ обріжемо до квадратної $d \times d$.

$$A'_{n \times m} = U'_{n \times d} \times \Sigma'_{d \times d} \times V'^{T}_{d \times m}$$

Виявляється, що отримана матриця $\mathbf{A}'$ близька до матриці $\mathbf{A}$ і є найкращим низькоранговим приближенням з точки зору середньо-квадратичного відхилення.

Щоб отримати оцінку користувача U для послуги I, ми беремо деякий вектор p_u (набір параметрів) для даного користувача та вектор для даної послуги q_i. Їх скалярний добуток є передбаченням $\hat{r_{ui}} = \langle p_u, q_i \rangle$.

Алгоритм враховує індивідуальність користувачів, базуючись по історії користувачів виявити приховані ознаки об'єктів та інтересів користувачів. Наприклад, на першій координаті вектора у кожного користувача буде стояти число, що вказує поход користувач більше на хлопчика чи дівчинку, а на другій - число, що відображає приблизний вік користувача. У послузі ж перша координата буде показувати, чи цікавий він більше хлопчикам чи дівчатам, а друга - якій віковій групі користувачів він цікавий.

Алгоритм має ряд складностей. По-перше, матриця оцінок $\boldsymbol{R}$ нам повністю не відома й тому ми не можемо просто взяти її SVD-розклад. По-друге, SVD-розклад не єдиний, що у свою чергу призводить до складності визначення параметрів [4].

Для вирішення першої проблеми використано машинне навчання. Отже, ми не можемо знайти SVD-розкладання матриці, тому що ми не знаємо саму матрицю. Але ми хочемо скористатися цією ідеєю і придумати модель передбачення, яка буде працювати подібним з SVD чином. Наша модель буде залежати від багатьох параметрів - векторів користувачів і послуг. Для заданих параметрів, щоб передбачити оцінку, ми візьмемо вектор користувача, вектор послуги і отримаємо їх скалярний добуток:

$$\hat{r_{ui}}(\theta) = p_u^T q_i$$

$$\boldsymbol{\theta} = \{\boldsymbol{p_u}, \boldsymbol{q_i} | \boldsymbol{u} \in \boldsymbol{U}, \boldsymbol{i} \in \boldsymbol{I}\}$$

Але так як векторів ми не знаємо, їх ще потрібно отримати. Ідея полягає в тому, що у нас є оцінок, за допомогою яких ми можемо знайти такі оптимальні параметри, при яких наша модель передбачала б ці оцінки якнайкраще:

$$\mathrm{E}_{(u,i)}(\hat{r}_{ui}(\Theta) - r_{ui})^2 \rightarrow \min_{\Theta}$$

Отже, ми хочемо знайти такі параметри θ, щоб квадрат помилки був якомога менше. Але тут є парадокс: ми хочемо менше помилятися в майбутньому, але ми не знаємо, які оцінки у нас будуть питати. Відповідно і оптимізувати це ми не можемо. Але нам відомі вже проставлені користувачами оцінки. Спробуємо підібрати параметри так, щоб на тих оцінках, які у нас вже є, помилка була якомога менше. Крім того, додамо ще один доданок - регуляризатора.

$$\sum_{(\boldsymbol{u},\boldsymbol{i})\in\boldsymbol{D}} (\hat{r}_{ui}(\Theta) - r_{ui})^2 + \boldsymbol{\lambda} \sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \boldsymbol{\theta}^2 \rightarrow \min_{\Theta}$$

Щоб знайти оптимальні параметри, оптимізуємо ось такий функціонал:

$$\boldsymbol{J}(\boldsymbol{\Theta}) = \sum_{(\boldsymbol{u},\boldsymbol{i})\in\boldsymbol{D}} (\boldsymbol{p_U^T q_i} - \boldsymbol{r_{ui}})^2 + \sum_{\boldsymbol{u}} ||\boldsymbol{p_u}||^2 + \sum_{\boldsymbol{i}} ||\boldsymbol{q_i}||^2$$

Щоб знайти мінімум функції, що залежить від багатьох зміних, використаємо градієнт - вектор з приватних похідних по кожному параметру.

$$\nabla \boldsymbol{J}(\boldsymbol{\Theta}) = \left(\frac{\boldsymbol{\partial J}}{\boldsymbol{\partial \theta_1}}, \frac{\boldsymbol{\partial J}}{\boldsymbol{\partial \theta_2}}, \dots, \frac{\boldsymbol{\partial J}}{\boldsymbol{\partial \theta_n}}\right)^{\boldsymbol{T}}$$

Метод градієнтного спуску - це ітеративний алгоритм: він багаторазово бере параметри певної точки, дивиться на градієнт і крокує проти його напрямки:

$$\Theta_{t+1} = \Theta_t - \eta \nabla J(\Theta)$$

Проблеми цього методу полягають в тому, що він, по-перше, в нашому випадку працює дуже повільно і, по-друге, знаходить локальні, а не глобальні мінімуми. Друга проблема для нас не так страшна, тому що в нашому випадку значення функціоналу в локальних мінімумах виявляється близьким до глобального оптимуму.

5. Висновки

Виявляється, що на сприйняття рекомендацій впливає не тільки якість ранжирування, а й деякі інші характеристики. Серед них, наприклад, різноманітність (не варто видавати користувачу послуги тільки на одну тему або з однієї серії), несподіванка (якщо рекомендувати дуже популярні послуги, то такі рекомендації будуть занадто банальними і майже марними), новизна (багатьом подобаються відомі послуги, але рекомендаціями зазвичай користуються, щоб відкрити для себе щось нове) і багато інших. Рекомендаційна система, яка враховує особливості користувачів та дає якісну рекомендацію є залогом успішності нового продукту на ринку.

Описана рекомендаційна система враховує індивідуальні властивості користувачів та послуг, що значною мірою важливе для профорієнтації – однієї з ключових задач, що вирішує додаток та є мало розвинутою сферою для сучасної молоді. Масштабованість, швидкий час опрацювання даних та навчання та гнучкість системи приводить до ефективного використання для сучасного програмного продукту..

Список літератури

1. What is the difference between traveller and voyager? [Електронний ресурс] – Режим доступу до ресурсу: http://wikidiff.com/voyager/traveller.
2. Ракова Л. Що впливає на вибір професії [Електронний ресурс] / Лидия Ракова // ХайВей. – 2011. – Режим доступу до ресурсу: http://h.ua/story/333964/.
3. Сингулярний розклад матриці [Електронний ресурс] // Вікіпедія. – 2017. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Сингулярний_розклад_матриці.
4. Ройзнер М. Как работают рекомендательные системы [Електронний ресурс] / Михайло Ройзнер // Яндекс. – 2014. – Режим доступу до ресурсу: https://habrahabr.ru/company/yandex/.
5. Машинное обучение и анализ данных [Електронний ресурс] – Режим доступу до ресурсу: https://www.coursera.org/specializations/machine-learning-data-

analysis?network=g&utm_source=gg&creativeid=142928104694&matchtype=p&adgroupid=34321642979&gclid=Cj0KEQjw0IvIBRDF0Yzq4qGE4IwBEiQATMQlMegyqDONlG9tArCuOZXMOToFIBIAPxcz4UeRnapc2LoaAvzw8P8HAQ&keyword=%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%BE%D0%B5+%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5+%D0%B2%D0%BE%D1%80%D0%BE%D0%BD%D1%86%D0%BE%D0%B2&hide_mobile_promo=&campaignid=669580045&devicemodel=&adpostion=1t1&utm_medium=sem&device=c.

6. Рудницький О. Живе спілкування не замінити віртуальним [Електронний ресурс] / Олександр Рудницький // Відеотека: канал \"Ранок\", Інтер. – 2012. – Режим доступу до ресурсу: http://inter.ua/uk/video/episode/faces_utro/2012/10/19/spilkuvannia.
7. Wise L. Don't let social media replace face-to-face conversation [Електронний ресурс] / Luan Wise // Social Media. – 2014. – Режим доступу до ресурсу: https://www.linkedin.com/pulse/20140804113036-20118767-don-t-let-social-media-replace-face-to-face-conversation.
8. Паундстоун У. КАК СДВИНУТЬ ГОРУ ФУДЗИ? / Уильям Паундстоун. – Москва: Альпина Паблишер, Харвест, 2004. – 272 с.

УДК 004

ЖАРІКОВ Е.В.,
СЕРДЮК Є.О.

КОНСОЛІДАЦІЯ ВІРТУАЛЬНИХ МАШИН З ВИКОРИСТАННЯМ ПРОМЕНЕВОГО ПОШУКУ

Одною з передумов реалізації хмарних обчислень є віртуалізація. Для віртуалізованих середовищ актуальною є задача розподілу і перерозподілу ресурсів фізичних серверів та обладнання. Поява нових завдань, зміни навантаження на віртуальні машини, різні вимоги до їх функціонування призводять до необхідності ефективного розміщення віртуальних машин та їх міграції в процесі консолідації. Для вирішення задачі консолідації віртуальних машин у статті пропонується двостадійний підхід на базі використання локального променевого пошуку. Розроблені евристики першої та другої стадій оптимізації, розроблений алгоритм променевого пошуку, функція оцінювання та умови закінчення роботи алгоритму. Для перевірки роботи алгоритму використані дані про надходження завдань в кластер Google.

Ключові СЛОВА: ХМАРНІ ОБЧИСЛЕННЯ, ВІРТУАЛІЗАЦІЯ, УПРАВЛІННЯ РЕСУРСАМИ, ЦЕНТР ОБРОБКИ ДАНИХ.

One of the prerequisites for the cloud computing is a virtualization. For virtualized environments it is actual task to allocate and reallocate physical servers resources. The emergence of new tasks, changes in virtual machine workloads, different requirements for virtual machine operations leads to the need for efficient allocation of virtual machines and their migration in the process of consolidation. In this paper the two-stage approach based on a local beam search algorithm is proposed to solve virtual machines consolidation problem. The heuristics of the first and the second stages of optimization algorithm, the algorithm of beam search, the assessment function, and the stop functions of the algorithm are developed. To test the algorithm Google cluster-usage traces are used.

Keywords: CLOUD COMPUTING, VIRTUALIZATION, RESOURCE MANAGEMENT, DATA CENTER.

1. Введення

Надання широкого спектру інформаційних послуг в сучасних умовах базується на таких сервісних моделях хмарних обчислень, як інфраструктура як сервіс (IaaS), платформа як сервіс (PaaS) та застосунки як сервіс (SaaS). З метою більш ефективного використання апаратного забезпечення в центрах обробки даних (ЦОД) застосовуються технології віртуалізації апаратного забезпечення, програмних засобів, мереж, сховищ даних, робочих місць та інші. Сервісна модель IaaS дозволяє більш ефективно використовувати апаратне забезпечення фізичного сервера (ФС) за рахунок віртуалізації його локальних ресурсів. При цьому, клієнтові надається частина ресурсів ФС у вигляді віртуальної машини (ВМ). Для реалізації сучасних інформаційних послуг клієнт розгортає одну або декілька ВМ в певній конфігурації, яка визначена провайдером хмарних послуг. При цьому виникає комплекс задач, пов'язаних з управлінням віртуальними машинами, фізичними серверами, мережевою взаємодією, сховищами, застосуваннями та іншимисистемами.

Для роботи кожного екземпляру застосування створюється окрема ВМ або контейнер. В цій статті розглядається робота застосувань на базі ВМ. Виходячи з певних бізнес-потреб клієнт може динамічно змінювати конфігурацію ВМ або налаштовує механізми балансування навантаження, відмовостійкості та резервного копіювання. В процесі роботи хмарного сервісу клієнти створюють множину ВМ, яка постійно

змінюється. Кількість ВМ, розміщених на окремому ФС може постійно змінюватися, тому деякі ФС виявляються незавантаженими до максимально можливого порогу і витрачають зайву енергію. З метою більш ефективного використання апаратних ресурсів ФС засоби віртуалізації надають можливість міграції ВМ з одного ФС на інший. При цьому, вплив на роботу ВМ є мінімальним, і для клієнта міграція ВМ виявляються непомітною. Але навантаження на фізичні сервери, які обмінюються цією ВМ, зростає.

Таким чином, однією з головних задач при управлінні ресурсами хмарного ЦОД є розміщення віртуальних машин таким чином, щоб задіяти меншу кількість фізичних серверів та зменшити кількість міграцій віртуальних машин. Процес перерозподілу віртуальних машин серед фізичних серверів називають також консолідацією ВМ. Вирішення проблеми консолідації віртуальних машин представлено в багатьох публікаціях [1]. Але дослідження виконувалися для наявних на той час технологій віртуалізації та хмарних технологій. В сучасних ЦОД впроваджуються все нові і нові апаратні та системні засоби які співіснують з технологіями попередніх поколінь. Таким чином, актуальною є розробка нових алгоритмів і методів для управління обчислювальними ресурсами ЦОД в цілому, та розміщенням віртуальних машин зокрема.

Для вирішення задачі консолідації віртуальних машин у статті пропонується двостадійний підхід на базі використання локального променевого пошуку (ПП). Розроблені евристики першої та другої стадій оптимізації, розроблений алгоритм променевого пошуку, функція оцінювання та умови закінчення роботи алгоритму. Для перевірки роботи алгоритму використані дані про надходження завдань в кластер Google.

2. Аналіз публікацій

Останнім часом запропоновано багато підходів та алгоритмів для вирішення проблеми консолідації віртуальних машин [1, 2, 3]. Проблема консолідації віртуальних машин розглядається як оптимізаційна проблема з різними цільовими функціями. Особливість цієї проблеми полягає в наявності великої кількості станів середовища та обмежень. Крім того, складність виявляється при формулюванні цільової функції, до якої входять декілька показників, які треба оптимізувати. Для промислових ЦОД з хмарними інфраструктурами використовуються прості алгоритми управління, такі як first-fit, best-fit та їх модифікації (Eucalyptus [4], Microsoft [5], Google [6]). Це обумовлено вимогами робастності застосувань клієнтів та участю адміністраторів в автоматизованому процесі управління ресурсами ЦОД при постійному моніторингу віртуальних машин.

В дослідженнях використовуються такі цільові функції, як мінімізація споживання електроенергії, мінімізація порушень угод про якість сервісу (SLA), мінімізація мережевого трафіку, максимізація продуктивності та використання ресурсів. Однією з основних умов для сучасних кластерів ФС є можливість роботи алгоритму в режимі онлайн [3]. Разом з цим допускається застосування методів оптимізації, які спрацьовують при появі певних умов роботи кластеру. Для такого випадку нові завдання розміщуються на ФС, які не задіяні в процесі консолідації.

Крім традиційних евристик та детермінованих алгоритмів для вирішення проблеми консолідації ВМ використовуються і алгоритми локального пошуку, такі як еволюційні алгоритми, алгоритми оптимізації мурашиних колоній, табу пошук, пошук з емуляцією відпалу. Більш ефективним на нашу думку є використання локального пошуку на другій стадії оптимізації, після підготовки відповідного набору станів детермінованими алгоритмами з метою покращити рішення, знайдене на першій стадії.

Таким чином, в цій статті пропонується застосувати локальний променевий пошук на підготовленому наборі даних з метою зменшити кількість міграцій ВМ та збільшити кількість вивільнених ФС, отриманих на попередній стадії оптимізації.

3. Модель системи

На теперішній час більшість послуг клієнти отримують на базі хмарних центрів обробки даних. Хмарні ЦОД являють собою складні системи, що складаються з серверних підсистем, підсистем зберігання даних, мережевих підсистем та підсистем інженерного забезпечення.

В статті розглянуто задачу управління окремою серверною підсистемою ЦОД у вигляді кластера в умовах віртуалізації. Для побудови кластеру в сучасних ЦОД використовують два підходи компоновки: з

використанням гетерогенної конфігурації або гомогенної конфігурації. Обидва підходи мають свої недоліки та переваги, однак уникнути розміщення гетерогенних конфігурацій в масштабі ЦОД, коли конфігурації кожного кластеру відрізняються, в більшості випадків не вдається з причин еволюції елементної бази серверів та нових вимог користувачів до IT-інфраструктури.

Розробка і дослідження роботи алгоритму променевого пошуку виконані для кластеру, який складається з фізичних серверів різної конфігурації. Кожен ФС надає для декількох ВМ декілька ресурсів: процесор (CPU), об'єм пам'яті (RAM), доступ до підсистеми зберігання даних (IOPS), доступ до мережевої підсистеми (NET) та інші. В залежності від наявної ємності ресурсів ФС, вимог до ресурсів з боку ВМ, часу виконання завдань всередині ВМ та інтенсивності надходження завдань кількість ВМ, що виконуються на ФС, постійно змінюється. Зміна кількості ВМ відбувається в таких станах: створення нової ВМ, видалення ВМ, міграція ВМ.

В цій статті розглядаються ресурси CPU та RAM, що надаються для ВМ. Однак алгоритм може бути доповнений іншими ресурсами, які потребує ВМ. Вибір саме двох ресурсів обумовлено використанням вхідних даних з набору Google cluster-usage traces (GCT) [7]. Для дослідження роботи алгоритму ПП з набору GCT використано таблиці "Machine events" та "Task events table". Випадковим чином з першої таблиці обрано 6000 ФС, з другої таблиці обрано 70000 завдань. З таблиці "Machine events" для кожного ФС використано такі атрибути: machine ID, capacity: CPU, capacity: memory. З таблиці "Task events table" для кожного ФС використано такі атрибути: task index within the job, machine ID, resource request for CPU cores, resource request for RAM. Дані в таблицях нормовані відносно ФС з найбільшим значенням ємності відповідного ресурсу.

4. Постановка задачі

Кластер управління складається з множини M фізичних серверів та N віртуальних машин, $N, M \in \bullet$. В процесі дослідження роботи алгоритму ПП кількість ФС та ВМ не змінюється. Кожне завдання з таблиці "Task events table" виконується в окремій ВМ. В загальному випадку, між запусками алгоритму ПП кількість ФС та ВМ може змінюватись.

Задана ємність j-ї ВМ для ресурсу k, $c_j^k \in (0,1]$, $k \in \{CPU, RAM\}$ визначається вимогами завдання і нормовано відносно ФС з найбільшою ємністю ресурсу k. Ємність фізичного сервера i для ресурсу k, $C_i^k \in (0,1]$ визначається типом ФС і нормовано відносно ФС з найбільшою ємністю ресурсу k.

Множина M складається з множини A фізичних серверів, які визначені для вимикання, та множини B фізичних серверів, які надають ресурси для ВМ, що будуть мігрувати з ФС, які належать до множини A, $A \cup B = M$, $A \cap B = \varnothing$.

Міграцію віртуальної машини j на фізичний сервер i позначимо як $U_{ij} \in \{0,1\}$. Міграція відбувається якщо $U_{ij} = 1$. Кожна ВМ має свій ID, який в процесі роботи алгоритму пов'язаний з номером j. Кожний ФС теж має свій ID, який в процесі роботи алгоритму пов'язаний з номером i.

5. Метод консолідації ВМ на основі променевого пошуку

Опис основного алгоритму

Вхідними даними алгоритму ПП є список ФС з множини A та список ФС з множини B, в яких є вільні ресурси і є можливість розмістити додаткові завдання у вигляді ВМ.

Ідея алгоритму: перегляд i-го ФС зі списку A та пошук таких або такого ФС зі списку B, куди можливо мігрувати j-ту ВМ з i-го ФС. Якщо вдалося звільнити всі або частку ФС зі списку A, видаємо результат у вигляді матриці U_{ij}, яка є планом міграцій.

Опис алгоритму променевого пошуку:

Перша стадія: підготовка вхідних даних для другої стадії. Формування списку A фізичних серверів, які треба звільнити від ВМ, та списку B фізичних серверів для визначення плану міграцій.

Друга стадія виконується для кожного ФС зі списку A:

1. На кожному кроці обираємо одну ВМ, назначену на i-й ФС і розглядаємо наступні варіанти обміну (міграцій):

a) мігрувати ВМ на інший ФС, в якого залишилось достатньо вільних ресурсів CPU та RAM;

b) перевірити можливість обміну з іншим ФС віртуальною машиною, яка вимагає менше ресурсів (так ми розглядаємо лише стани з кращою

оцінкою та уникаємо можливих зациклювань) і, в результаті, ФС не буде перенавантаженим після обміну.

2. З усіх можливих обмінів обирається n (ширина променю) обмінів з найвищою оцінкою.

3. Завершуємо пошук якщо i-й ФС вдалося звільнити від віртуальних машин або якщо не можна побудувати нові стани (тобто не залишилось жодного варіанту для реалізації допустимого обміну a) або b)).

Порівняння станів виконується за допомогою критерію (1):

$$J = \sum_{i=1}^{m} u_i^2 + \sum_{i=1}^{n} f_i^2, \qquad (1)$$

де u_i – кількість використовуваних ресурсів на i-му ФС, f_i – кількість вільних ресурсів на i-му ФС, m – кількість ФС в списку B, n – кількість ФС в списку A.

Таким чином, алгоритм поступово зменшує використовувані ресурси на ФС, які належать до списку A, та завантажує ФС зі списку B.

Умови завершення алгоритму

Для завершення роботи алгоритму пропонуються наступні умови:

1. Вичерпання списку A.

2. Неможливість мігрувати всі ВМ з певної визначеної кількості ФС Th_A поспіль.

Якщо другу умову не застосовувати, цикли пошуку виконуються занадто довго, якщо співставити їх з часом на створення нової ВМ та часом на міграцію для ВМ з середніми вимогами до ресурсів пам'яті. Для визначення Th_A пропонується застосувати евристику, яка полягає у розгляді певного відсотку фізичних серверів зі списку A, але не менше, ніж α фізичних серверів. В реалізації досліджуваного алгоритму прийнято $Th_A = 0.05$, $\alpha = 10$. З меншими значеннями α алгоритм пропускає значну кількість ФС, що могли бути вимкнуті, але в результаті завершення алгоритму були не звільнені від працюючих на них ВМ. Даний критерій завершення вводиться для зменшення часу виконання алгоритму.

Визначення вхідних даних (опис першої стадії роботи алгоритму)

Отримання списку фізичних серверів, з яких треба мігрувати всі ВМ з метою подальшого вимкнення ФС пропонується здійснювати за допомогою двох методик: *нижньої границі* та *порогу вільних ресурсів*. Розглянемо кожну з методик більш детально.

Методика *нижньої границі* полягає у визначенні такої кількості фізичних серверів, які вимкнути в результаті міграції усіх ВМ, що на них працює, не уявляється можливим. Таке уявлення виникає з гіпотез, що використовуються в роботі алгоритму визначення нижньої границі.

Пошук нижньої границі виконується наступним чином:

1. Знаходимо середній об'єм ресурсів по всім ФС, на яких працюють ВМ, $\sum_{i=1}^{M} C_i^k$. Значення для кожного ресурсу k розраховуються окремо.

2. По кожному ресурсу окремо рахуємо суму необхідних ресурсів для виконання всіх наявних ВМ, $\sum_{j=1}^{N} c_j^k$.

3. Окремо по кожному ресурсу рахуємо відношення необхідних ресурсів до середнього об'єму ресурсів та округляємо значення до більшого цілого.

4. Нижньою границею буде найбільше число з отриманих на кроці 3.

5. Сортуємо фізичні сервери одним з наступних способів:

a) за об'ємом ресурсів ФС, потім за відношенням використаних ресурсів до кількості працюючих ВМ;
b) за об'ємом ресурсів ФС, потім за кількістю назначених завдань, потім відношенням використаних ресурсів до кількості працюючих ВМ.

В результаті досліджень роботи алгоритму з'ясувалося, що варіант b) виявився значно ефективнішим. При його використанні на одному й тому самому наборі даних вдалося вимкнути 82 ФС, тоді як при використанні варіанту a) вдалося вимкнути лише 33 ФС.

В результаті, знаходимо різницю між кількістю ФС, що використовуються, та отриманою нижньою границею. Знайдене число – це кількість ФС списку A на вимкнення, які є першими у відсортованому списку. Решта ФС потрапляє у список B.

Ідея методики *порогу вільних ресурсів* полягає в такому формуванні списку A, при якому до цього списку потрапляють ФС, загальна кількість невикористаних ресурсів яких перевищує об'єм ресурсів одного з ФС кластеру.

Побудова списку A з використанням порогу вільних ресурсів β виконується наступним чином:

1. Встановлюємо значення β.

2. Відбираємо ФС, у яких кожного вільного ресурсу більше за значення β.

3. По кожному ресурсу окремо рахуємо суму вільних ресурсів.

4. Сортуємо обрані ФС аналогічно до варіанту b) з методики *нижньої границі*.

5. Проходимо по відсортованому списку. Поки сума ресурсів більша за об'єм поточного ФС віднімаємо від суми цей об'єм, додаємо поточний ФС в список *A* та переходимо до наступного ФС, інакше завершуємо проходження списку. Таким чином, отримуємо список *A* з ФС, які треба вимкнути, та список *B* з ФС для обміну віртуальними машинами.

6. Оцінка результатів моделювання

Дослідження роботи алгоритму виконане на фрагментах набору вхідних даних GCT [7]. Дані для тестування і дослідження алгоритму обрані з GCT для певного діапазону часу, за який збиралися дані на реальному кластері. Це необхідно, щоб врахувати всі дані за один і той самий період часу в журналі GCT. З набору випадковим чином відібрано 70000 завдань з певними вимогами до ресурсів *k*. Кожному завданню відповідатиме окрема ВМ. Обираємо 6000 фізичних серверів, з яких на 5832 серверах розміщено ВМ, та на 168 серверах завдань немає, але вони доступні для розміщення ВМ. Ширина променю для алгоритму дорівнює 5. В даній роботі дослідження впливу ширини променю на роботу алгоритму не досліджувалося.

В таблиці 1 представлені результати роботи алгоритму променевого пошуку з консолідації ВМ для різних значень β. В таблиці 2 представлені результати роботи алгоритму променевого пошуку при консолідації ВМ з використанням методики *нижньої границі.*

Якісними показниками роботи алгоритму є кількість вимкнених ФС та кількість міграцій ВМ, за допомогою яких ФС вивільнюються від ВМ.

Табл. 1. Результати моделювання консолідації ВМ алгоритмом променевого пошуку з використанням методики порогу вільних ресурсів

β	*B*	*A*	Вимкнено ФС	Міграцій	Час, с
0.005	199	3	0	0	0.53
0.004	299	3	0	0	0.69
0.003	1098	9	0	0	4.05
0.002	1393	10	0	0	6.55
0.001	1868	28	13	219	38.08
0.0009	1943	31	15	263	42.29
0.0008	2002	39	22	349	67.93
0.0007	2081	43	24	399	81.02
0.0006	2204	48	28	432	94.14
0.0005	2302	55	32	504	105.48
0.0004	2407	63	36	554	141.43
0.0003	2565	73	43	683	143.16
0.0002	2764	82	48	782	166.54
0.0001	3706	95	59	992	238.06
0.00005	4323	106	68	1199	330.18
0.00001	5095	116	75	1336	463.36
0	5328	125	84	1459	518.34

Табл. 2. Результати моделювання консолідації ВМ алгоритмом променевого пошуку з використанням методики нижньої границі

B	*A*	Вимкнено ФС	Міграцій	Час, с
5707	125	82	1460	508.09

Вплив значення порогу β на якісні показники роботи алгоритму є суттєвим (Рис.1 та Рис.2), залежність виявилася майже лінійною. Чим менший поріг β тим більше ФС потрапляють на вхід алгоритму.

При $\beta = 0$ деякі ФС (а саме ті, у яких хоча б один ресурс повністю зайнятий) не потрапляють на вхід алгоритму, тому що при перевірці наявності ресурсів використовується строга нерівність. При $\beta < 0.0002$ алгоритм з методикою порогу вільних ресурсів починає працювати більш ефективно і на граничних значеннях працює краще чим з методикою нижньої границі. Для всебічної перевірки роботи алгоритму з різними методиками та їх порівняння планується створити декілька наборів даних з різною кількістю ФС та ВМ.

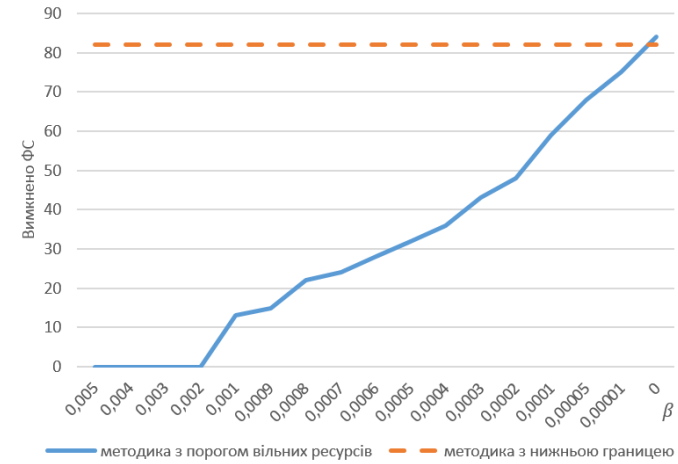


Рис.1. Кількість вимкнених ФС

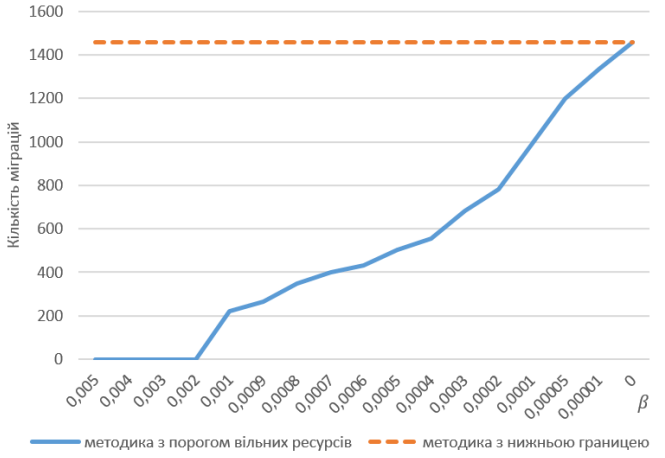


Рис. 2. Кількість міграцій

Перевагою використання методики з порогом вільних ресурсів є можливість адаптуватися до стану кластера, враховуючи інші ресурси, час міграції ВМ та інтенсивність надходження заявок на створення нових ВМ.

Висновки

Для управління процесом консолідації віртуальних машин в статті запропоновано використовувати підхід на базі алгоритму променевого пошуку. Розроблені евристики першої та другої стадій роботи алгоритму, розроблений алгоритм променевого пошуку для консолідації ВМ, розроблені функція оцінювання та умови закінчення роботи алгоритму. Для перевірки роботи алгоритму використані дані про надходження завдань в кластер Google. методики нижньої границі та порогу вільних ресурсів, виконано їх порівняльний аналіз. В результаті пропонується використовувати методику порогу вільних ресурсів, що показала більш якісні результати консолідації ВМ.

Список літератури

1. F. Lopez Pires and B. Baran, “A virtual machine placement taxonomy,” in Proc. of the 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), 2015, pp. 159–168.
2. R. W. Ahmad, A. Gani, S. H. A. Hamid, M. Shiraz, A. Yousafzai, and F. Xia, “A survey on virtual machine migration and server consolidation frameworks for cloud data centers,” *Journal of Network and Computer Applications*, vol. 52, pp. 11–25, 2015.
3. S. Telenyk, E. Zharikov, O. Rolik, “An approach to virtual machine placement in cloud data centers,” In proc. of the 2016 International Conference Radio Electronics & Info Communications (UkrMiCo) 11–16 September, Kyiv, Ukraine. – 2016 pp. 1–6. http://dx.doi.org/10.1109/UkrMiCo.2016.7739645
4. Eucalyptus community. http://open.eucalyptus.com/
5. S. Lee, R. Panigrahy, V. Prabhakaran, V. Ramasubrahmanian, K. Talwar, L. Uyeda, and U. Wieder. Validating heuristics for virtual machines consolidation. Microsoft Research, MSR-TR-2011- 9, 2011.
6. B. Sharma, V. Chudnovsky, J. L. Hellerstein, R. Rifaat, and C. R. Das. Modeling and synthesizing task placement constraints in google compute clusters. In Proceedings of the 2ndACM Symposium on Cloud Computing (SOCC), 2011.
7. C. Reiss, J. Wilkes, and J. L. Hellerstein, “Google cluster-usage traces: format+ schema,” Google Inc., Mountain View, CA, USA, Technical Report, 2011.

УДК 004.031.42

ХАРЛАМБОВ К.К.
ГОЛОВЧЕНКО М.М.

БІБЛІОТЕКА ДЛЯ ПЕРЕНЕСЕННЯ БІЗНЕС-ЛОГІКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ У СПЕЦІАЛЬНО ВИДІЛЕНЕ СЕРЕДОВИЩЕ

В даній статті описані основні принципи роботи та використання даної бібліотеки, її внутрішня структура та функціональні можливості.

This article describes the basic principles of operation and usage of the library, its internal structure and functionality.

1. Введення

Розвиток інформаційних технологій ще давно мав чи не найвищий пріоритет в науковій сфері, основною метою якого було знайти найоптимальніший розв'язок поставленої задачі. Так виникали нові інструменти, мови, методології розробки програмного забезпечення, парадигми програмування тощо.

Дана бібліотека призначена для вирішення проблеми розбиття програми на системний та прикладний код. Вона використовує один з SOLID принципів – інверсію управління, а точніше кажучи, формує спеціальне середовище (контекст), в якому створюються компоненти з модулів або класів, а також залежності між ними (dependency injection), що описані в файлі конфігурації даного контексту у форматі JSON (файлів може бути декілька). Після ініціалізації компонент в контексті програміст-користувач має можливість отримувати вже готові компоненти з контексту (dependency lookup) і використовувати в подальшій розробці свого програмного додатку на рівні прикладного коду.

Окрім виділення контексту бібліотека надає можливість не просто використовувати компоненти, а й можливість керувати їх життєвим циклом. В більш розгорнутому і детальному вигляді це реалізовано в окремому модулі даної бібліотеки – модулі з підтримкою аспектно-орієнтованого програмування, де компоненти вже використовуватимуться на рівні аспектів – класів чи модулів з наскрізною функціональністю, тобто формуватимуть принцип розкиданого коду (виклики компонент в прикладному коді будуть розкидані по всій системі).

Що стосується технічного використання даної бібліотеки, то вона спирається на платформу Node.js і написана на мові Typescript.

2. Внутрішня структура бібліотеки

Як вже було зазначено, бібліотека складається з двох модулів: основного модуля (Library Core Module), в якому знаходиться Inversion of Control Container, тобто контекст, в якому будуть створюватися, видалятися компоненти та встановлюватимуться залежності між ними, тобто наявна підтримка життєвого циклу всіх складових контексту; допоміжного модуля з підтримкою принципів аспектно-орієнованого програмування (Library AOP Module), що надає змогу викликати компоненти в будь-якому місці прикладного коду та розширює можливості контексту з потоком керування набору компонент в системі.

За умови інтеграції Library AOP Module в систему користувача всі компоненти матимуть можливість перейти на рівень аспектів, отримати взаємодію з порадами (advices) та точками зрізу (pointcuts), проте вони все одно будуть проходити ініціалізацію в контексті. Для Library AOP Module

визначений окремий тип JSON-формату, який не порушує конфігурації звичайних компонент, сформованих внаслідок генерації через стандартну конфігурацію. Приклад структури можна подивитися на рис. 1.

3. Архітектура програмного додатку з використанням даної бібліотеки

Для того, щоб використати дану бібліотеку, достатньо розбити систему на логічні програмні блоки (модулі або класи). Після цього потрібно створити конфігураційні файли формату JSON за специфікацією самої бібліотеки, де будуть описані майбутні компоненти. Це надає можливість контексту перехопити керування над об'єктами, які ми позначили як компоненти в файлах конфігурації, і вже використовувати готові об'єкти з відповідною функціональністю. Це як правило, можуть бути сервіси для отримання та обробки даних з бази даних або інші елементи бізнес-логіки системи, адже основна ідея бібліотеки полягає в слабкій зв'язаності між компонентами системи для досягнення масштабованості програмного забезпечення. Основні положення використання бібліотеки представлені на рис.2.

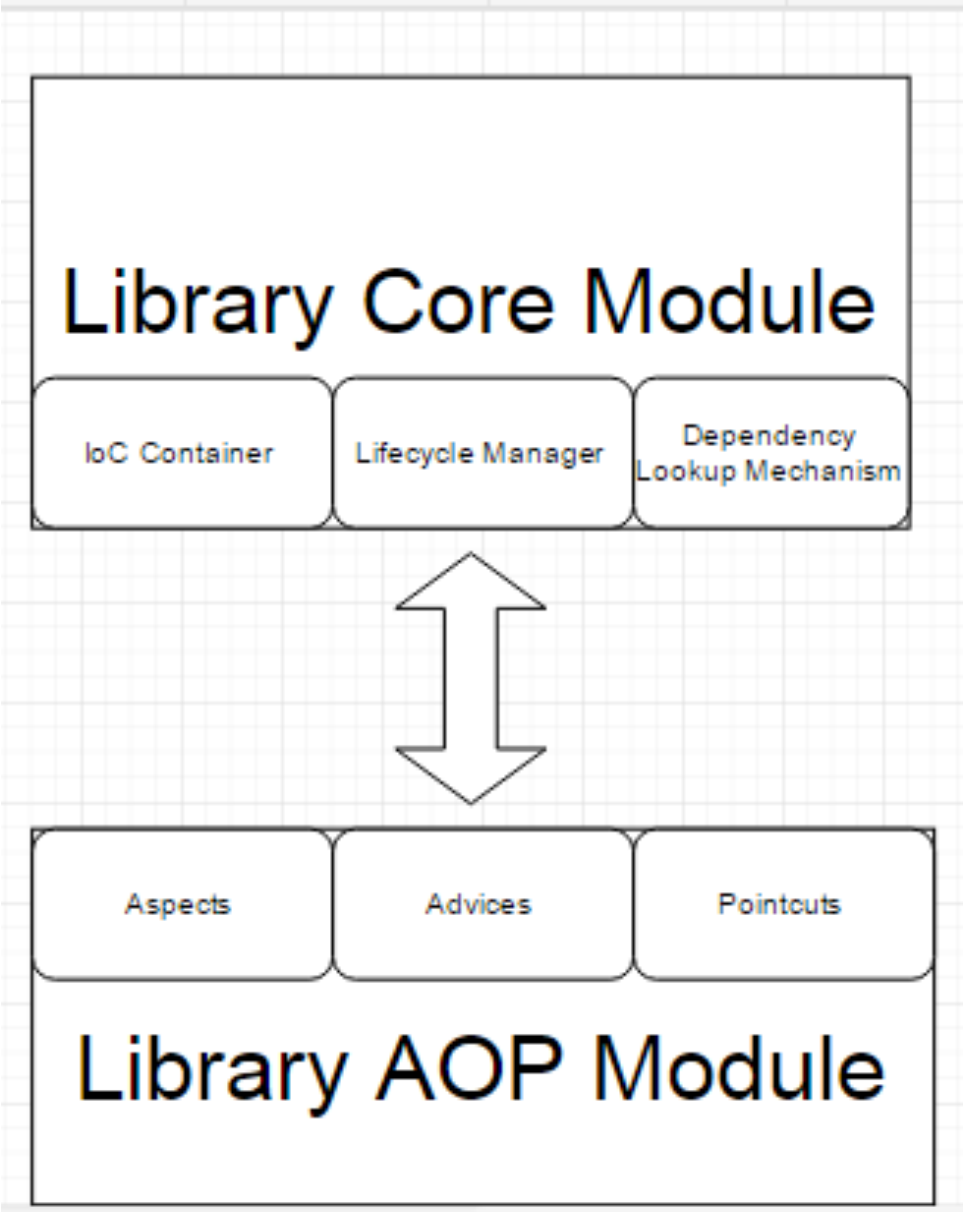


Рис. 1. Схема внутрішньої структури бібліотеки

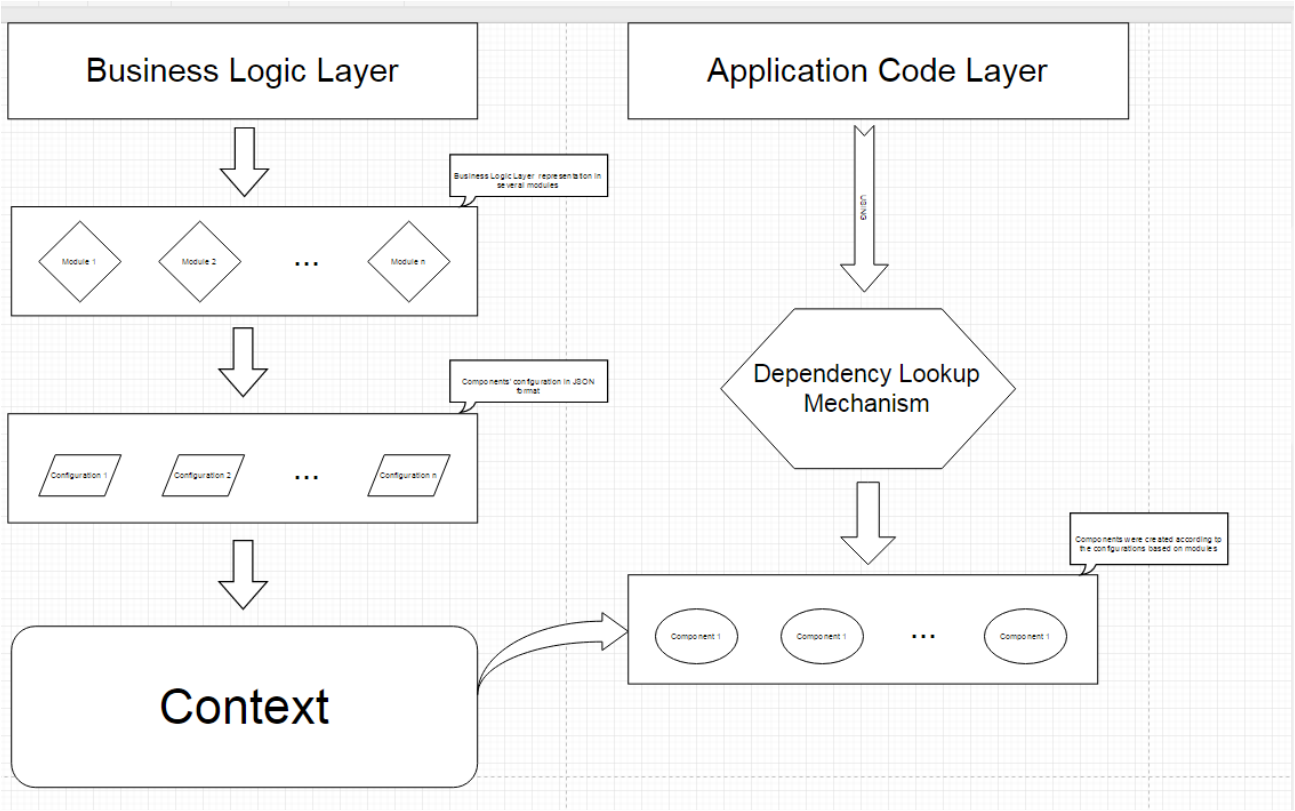


Рис.2. Схема принципу роботи програмного додатку з використанням даної бібліотеки

4. Висновок

Підсумовуючи попередні три розділи, можна зрозуміти, що використання принципів інверсії управління та аспектно-орієнтованого програмування дозволяє зробити програмісту-користувачу більш гнучку та ефективну систему. Ідея розбиття системи на окремі модулі та формування єдиного контексту як групуючого та керівного елементу системи дозволяє вирішити проблему масштабованості програмного забезпечення.

Крім того, це вже було підтверджено Java-фреймворком Spring, який і досі вважається чи не найефективнішим інструментом для розробки Java enterprise-додатків. В Node.js такої парадигми ще не ввели або ввели частково, наприклад бібліотека spring.js, яка імітує механізм роботи фреймворка, проте цей проект призупинився, що і спонукало мене як розробника продовжити втілення ідеї Inversion of Control Container в платформі Node.js.

Список літератури

1. Martin Fowler. Inversion of Control Containers and the Dependency Injection pattern
2. Robert E. Filman; Tzilla Elrad; Siobhán Clarke; Mehmet Aksit. Aspect-Oriented Software Development.
3. Andreas Holzinger; M. Brugger; W. Slany. Applying Aspect Oriented Programming (AOP) in Usability Engineering processes

УДК 004.031.42

ДОВГАЛЬ Д.О.

СИСТЕМА ТЕХНІЧНОЇ ПІДТРИМКИ КОРИСТУВАЧІВ ЛОКАЛЬНОЇ КОМП'ЮТЕРНОЇ МЕРЕЖІ КАФЕДРИ ВНЗ

Пропонується система для технічної підтримки користувачів локальної комп'ютерної мережі кафедри ВНЗ. Дана система повинна полегшити роботу адміністраторів і комунікацію між користувачами і адміністраторами.

Proposed system for technical support department university LAN universities. This system should facilitate the work of administrators and communication between users and administrators.

1. Вступ

Локальна комп'ютерна мережа кафедри ВНЗ складається з певного набору пристроїв, які можна розподілити на групи, а саме персональні комп'ютери, сервери, активне мережеве обладнання тощо. Керування та моніторинг стану мережі та обладнання здійснюється системними адміністраторами і, як правило, використовуються певні засоби віддаленого контролю та моніторингу.

У студентів та викладачів, що є користувачами даної мережі, можуть виникати проблеми з роботою обладнання. Для полегшення інформування системних адміністраторів про виникнення проблем в роботі з обладнанням, розроблено систему технічної користувачів. Вона дає змогу зменшити затрати часу на інформування про проблеми в роботі обладнання, що дасть змогу більш швидко реагувати на проблеми, а також дозволить рівномірно розподілити завдання між системними адміністраторами.

2. Огляд наявних аналогів

В ході пошуку схожих за функціональністю систем було виявлено наступні системи зі схожими функціями:

- Microsoft Ticket System
- Кларис ServiceDesk
- OTRS
- Mobile Asset Inventory

Функціональність Microsoft Ticket System, Кларис ServiceDesk, OTRS направлена на спільне вирішення проблем користувачів, що надходять від них у вигляді заявок. Але в цих системах є ряд відмінностей, а саме:

- Microsoft Ticket System орієнтована тільки на операційну систему Windows;
- Кларис ServiceDesk є комерційним продуктом (сайтом), для роботи з яким потрібне постійне підключення до мережі Інтернет;

Також в даних системах відсутня можливість врахування обліку обладнання та кімнат (аудиторій).

Mobile Asset Inventory призначена тільки для інвентаризації обладнання.

3. Постановка завдання

Веб-ресурс надає змогу зареєстрованим користувачам можливість створення заявки, що описує проблему. Користувач має змогу відслідковувати статус своїх заявок. Головний адміністратор відслідковує заявки, формує задачі та підзадачі для виконання, призначає виконавців та відповідальних, змінює статус заявок. Адміністратори можуть переглядати задачі та терміни для виконання. Адміністратори можуть вести та редагувати облік обладнання, переглядати та редагувати розклад занять на які потрібен проектор. Адміністратори можуть редагувати вже існуючі завдання та створювати свої.

3.1. Призначення системи

Дана система призначена для моделювання наступних процесів у роботі веб-ресурсу:

- подання користувачами заявки на обслуговування/підтримку;
- відслідковування користувачами статусу заявок;
- надання доступу користувачу для зміни контактних даних та паролю;

- перегляд адміністраторами заявок;
- формування адміністраторами завдань та підзадач;
- призначення адміністраторами відповідальних за виконання завдань;
- зміна адміністраторами статусу заявок;
- перегляд адміністраторами розкладу занять в аудиторіях;
- перегляд і редагування адміністраторами обліку обладнання.

3.2. Цілі створення системи

Головними цілями створення даної системи є:

- підвищення швидкості виконання заявок користувачів;
- покращення якості виконання заявок користувачів;
- покращення інформування користувачів про виконання його заявки.

3.3. Задачі розробки

Для досягнення поставлених цілей необхідно вирішати наступні задачі:

- подання користувачами заявки на обслуговування/ підтримку;
- ведення історії заявок;
- відслідковування користувачами статусу заявок;
- перегляд адміністраторами заявок;
- формування адміністраторами завдань та підзадач;
- призначення адміністраторами відповідальних за виконання завдань;
- перегляд адміністраторами розкладу занять в аудиторіях;
- перегляд і редагування адміністраторами обліку обладнання.

4. Опис функціональної моделі

Нижче наведений опис кожного з акторів.

Користувач – користувач кафедри, що має доступ до обладнання. Має змогу створювати заявку, в якій вказує інформацію про проблему, а також відслідковувати стан виконання заявки та коментувати її.

Викладач – викладач кафедри, що має всі можливості користувача та можливість вказувати бажаний час виконання заявки та її пріоритет.

Інтерн – користувач, що проходить практику в підрозділі системних адміністраторів. Окрім функцій викладача, виконує призначені йому заявки, має змогу переглядати інформацію про всі збережені в системі заявки, редагувати розклад пар на які потрібен проектор та інформацію про обладнання.

Адміністратор – системний адміністратор кафедри. Окрім функцій інтерна, має змогу реєструвати та призначати виконавця заявки.

Головний адміністратор - системний адміністратор кафедри. Окрім функцій адміністратора, має змогу змінювати статус заявок.

На рисунку 1.1 наведена діаграма використання системи

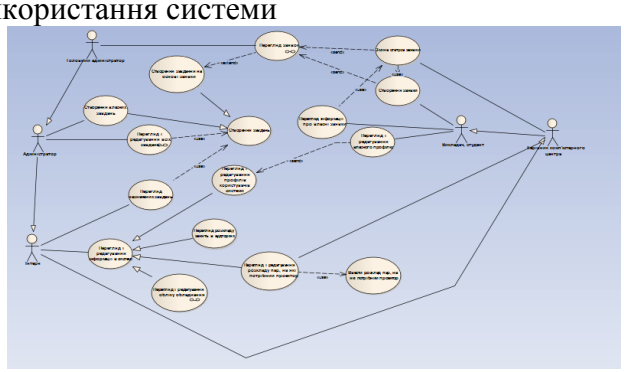


Рисунок 1.1 – Схема структурна варіантів використання

5. Засоби розробки

При створенні програмного продукту були використані такі засоби для програмування на c#/.NET як ASP.NET MVC з використанням СУБД Microsoft SQL Server.

На сьогоднішній момент мова програмування C# одна з найпотужніших і затребуваних мов в IT-галузі. На даний момент на ній пишуться найрізноманітніші програми: від невеликих десктопних програм до великих веб-порталів і веб-сервісів, які обслуговують щодня мільйони користувачів.

Платформа ASP.NET MVC має наступні переваги.

- Вона полегшує управління складними структурами шляхом поділу додатки на модель, вид і контролер.
- Вона не використовує стан перегляду і серверні форми. Це робить платформу MVC ідеальної для розробників, яким необхідний повний контроль над поведінкою програми.
- Вона використовує схему основного контролера, при якій запити веб-додатки обробляються через один контролер. Це

дозволяє створювати додатки, що підтримують розширену інфраструктуру маршрутизації.

- Вона забезпечує розширену підтримку розробки на основі тестування.
- Вона добре підходить для веб-додатків, підтримуваних великими колективами розробників, а також веб-розробникам, яким необхідний високий рівень контролю над поведінкою програми.

Microsoft SQL Server - одна з найбільш потужних систем роботи з базами даних в архітектурі "клієнт-сервер".

У своєму складі система має засоби створення баз даних, роботи з інформацією баз даних, перенесення даних з інших систем і в інші системи, резервного копіювання та відновлення даних, розвинену систему транзакцій, систему реплікації даних, реляційну підсистему для аналізу, оптимізації та виконання запитів клієнтів , систему безпеки для управління правами доступу до об'єктів бази даних та ін.

6. Висновок

Була розроблена система автоматизаії обробки заявок користувачів локальної комп'ютерної мережі кафедри ВНЗ, що складається з задач управління заявками, користувачами та обліком обладнання.

Було розглянуто предметне середовище, наведено опис варіантів використання системи, проведено огляд наявних аналогів, визначено призначення, цілі та задачі розробки підсистеми.

На етапі програмної реалізації підсистеми був обґрунтований вибір програмної платформи, системи управління базами даних та використаних інструментів розробки програмного забезпечення.

Список використаної літератури

1. MSDN [Электронний ресурс] / Режим доступу: http://www.msdn.microsoft.com
2. Castillo, Carlos. Effective Web Crawling [Текст]/ Castillo, Carlos // Santiago, 2004. – 191с.
3. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений [Текст]/ Буч Г., Максимчук Р., Энгл М., Коналлен Д., Хьюстон А. // М.: Вильямс, 2008. – 720с.
4. Ковалюк Т.В. Основи програмування [Текст] – К.: Робоча група BHV, 2005.
5. Help Desk, Service Desk Best Practice Handbook: Building, Running and Managing Effective Support - Ready to use supporting documents bringing ITIL Theory into Practice [Текст] - Emereo Pty Ltd, 2008

УДК 004.031.42

САГАН Б. Г.
КОВАЛЮК Т. В.

ПОБУДОВА ІНДИВІДУАЛЬНИХ ПЛАНІВ НАВЧАННЯ СТУДЕНТА НА ОСНОВІ ЙОГО КОМПЕТЕНТНОСТЕЙ ТА ПОБАЖАНЬ

Пропонується підхід для побудови індивідуальних планів навчання для кожного студента, на основі його компетентностей та побажань. В даному підході враховуються вимоги ринку до спеціалістів, на основі яких формуються набори дисциплін, які будуть максимально їх задовольняти.

The approach to build individual learning plans for each student, based on his competence and wishes. In this approach takes into account business demand for specialists on which are formed set of disciplines that will meet their maximum.

1. Вступ

Одним з найважливіших завдань ВНЗ є задоволення ринку праці в спеціалістах, яких він потребує. При цьому ВНЗ повинні мати високу якість підготовки фахівців в кожній галузі ринку, яка напряму залежить від того, наскільки випускник буде відповідати компетенціям, що були поставлені ринком. Проте, витримувати високу якість підготовки в теперішній час дуже складно, так як наука та техніка стрімко розвиваються і постійно висуваються нові вимоги. Випускники кожного нового року завжди повинні мати вищий рівень підготовки, в порівнянні з їх попередники. Одним із факторів досягення високого рівня є навчальний план ВНЗ. Підготовка фахівців, що будуть відповідати компетенціям, які висуває ринок, тягне за собою безперевне оновлення та вдосконалення навчальних планів. Тобто, навчальний план має бути достатньо гнучким для швидкої адаптації до мінливих вимог, які диктує ринок.

2. Навчальний план

Навчальний план – документ, складений вищим навчальним закладом освіти на підставі освітньо-професійної програми та структурно-логічної схеми підготовки, який визначає перелік і обсяг нормативних та вибіркових навчальних дисциплін, послідовність їх вивчення, конкретні форми проведення навчальних занять та їх обсяг, графік навчального процесу, форми і засоби здійснення поточного й підсумкового контролю.

Окрім вище перечисленого у навчальному плані зазначається обсяг часу, передбаченого на самостійну роботу. На основі навчального плану ВНЗ формує робочий навчальний план, тобто план на поточний навчальний рік. Він конкретизує форми проведення навчальних занять, їх обсяг, форми і засоби поточного та підсумкового контролю за семестрами.

З метою покращення якості навиків випускників у вищих навчальних закладах практикується складання студентами індивідуальних навчальних планів. Такий план є нормативним документом, за яким здійснюється навчання студента з огляду на вимоги освітньо-професійної програми відповідного рівня підготовки та його освітньо-професійних інтересів і потреб.

3. Методи побудови навчального плану

Одним з методів складання навчальних планів і програм є організація модульного навчання. В останні роки в цьому напрямку зроблено безліч розробок[1]. Суть модульного навчання полягає в тому, щоб максимально відокремити окремі блоки (модулі) навчального матеріалу. Кожен модуль при його вивченні забезпечує досягнення певної дидактичної мети. Навчальний матеріал, що охоплюється

модулем, повинен бути настільки закінченим блоком, щоб існувала можливість конструювання єдиного змісту з окремих модулів без порушення логічності викладу матеріалу.

Модульне навчання передбачає максимум самостійної роботи студента. Функції педагога при такому навчанні все більше зводяться до консультативним.

Мета розробок, що ведуться в цьому напрямку - створення гнучкого змісту навчання з можливістю заміни окремих модулів.

При модульній побудові навчання пропонується наступна методика формування змісту модулів. Будується граф логічної структури предмета, в якому вказуються не лише внутрішньо-предметні, але і міжпредметні зв'язки. Потім в окремі навчальні елементи, що складають структуру модуля, вибираються повністю ті теми з графа логічної структури, які необхідні для вивчення конкретного навчального елемента, що дозволяє по можливості забезпечити його велику автономність, досягти повноти змісту навчального матеріалу в ньому. У зв'язку з цим у зміст навчального елемента, крім вищевказаних тем, включаються і теми інших предметів, на які вказали міжпредметні зв’язки.

Така побудова навчання має свої переваги та недоліки. В якості переваг можна вказати те, що досягається певна гнучкість навчання. Можна переміщати в часі окремі блоки модулі навчального матеріалу без аналізу їх зовнішніх зв'язків, так як модулі є максимально відокремленими і закінченими структурами.

Проте значним недоліком такої організації побудови навчального плану є те, що в модулі міститься інформація, що не відноситься безпосередньо до досліджуваної дисципліни. Причому, інформація фундаментальних наук для даної спеціальності (зокрема, для інженерної освіти - математика, фізика та інші загальнотехнічні дисципліни) може дублюватися кілька разів в різних модулях. Це, звичайно ж, позитивно впливає на якість засвоєння матеріалу, але значно скорочує загальний обсяг навчального матеріалу, який можна піднести студенту за термін його навчання у вузі.

Ще одним з напрямків робіт в області вдосконалення підготовки фахівців у вищих навчальних закладах є складання навчальних планів вищих навчальних закладів на основі дерева цілей підготовки фахівця[2, 3].

Коротко про суть методу. Метод реалізується на основі побудови дерева цілей навчального процесу підготовки фахівця[4]. Дерево цілей має кілька ієрархічних рівнів. Різні автори пропонують будувати ієрархію рівнів по-різному.

Основні цілі навчання - що повинен знати і вміти випускник вузу. Кожній цілі у відповідність ставиться одна чи кілька дисциплін навчального плану. Кожну дисципліну, в свою чергу, можна розбити на теми.

Обсяг навчального плану в годиннику відомий заздалегідь, необхідно наповнити цей обсяг найбільш важливим змістом.

За умови, що осінній семестр містить 18 тижнів, весняний - 17 тижнів, кожну дисципліну зручно ділити на 17-годинні елементи.

Таким чином, дерево цілей навчального процесу містить три рівні:
- цілі навчального процесу;
- розділи (блоки дисциплін) навчального плану;
- 17-годинні елементи.

Вхідними даними є коефіцієнти відносної важливості цілей навчального процесу, а також ваги цілей другого рівня щодо цілей першого рівня. Виходячи з цих даних, обчислюються коефіцієнти відносної важливості цілей другого рівня, ваги цілей третього рівня щодо цілей другого рівня і коефіцієнти відносної важливості цілей третього рівня (17-годинних елементів), а також групові ваги елементів навчального плану.

Обсяг навчального плану відомий, можна перевести його в елементи. Тоді, розмістивши елементи в порядку убування групових ваг елементів навчального плану, потрібно відібрати в навчальний план R

перших елементів, де R - обсяг навчального плану в елементах. Потім проводиться експертне опитування по зв'язках між обраними в навчальний план елементами.

При такому алгоритмі роботи не враховуються зв'язки між модулями. Зв'язки між модулями, що потрапили в навчальний план, оцінюються після відбору змісту, тому може проявитися інформаційна недостатність для вивчення деяких модулів, так як необхідні для них в якості інформаційної бази елементи-предки можуть мати недостатньо високу групову вагу.

3. Побудова індивідуального навчального плану з урахування компетентностей та побажань студента

Для досягнення максимальної якості підготовки фахівців запропоновано об’єднати переваги існуючих методів побудови навчальних планів. В даному методі побудови індивідуального навчального плану студента будуть брати участь як ринок, що диктує вимоги до випускників так і сам студент.

На вхід нашої системи буде надходити набір дициплін, який сформовано на основі компетенцій якими має володіти студент для успішного працевлаштування в обраній галузі. Кожна дисципліна буде покривати одну або декільна висунутих ринком компетенцій, а також буде містити оцінки важливості відповідних компетецій. Далі студент буде обирати ті дисципліни які він хотів би опанувати. Слід зауважити, що початковий набір дисципніл більший від того набору який буде вивчатись студентом.

При побудові навчального плану ми будемо дотримватись принципів модульності та дерева цілей підготовки фахівця. Таким чином, ми досягнемо високої гнучкості та зможемо швидко реагувати на зміни вимог ринку або особистих побажань студента.

Також враховується те, що студент був попередньо протестований і має свій власний набір компетентностей, які описують його рівень. Тобто якщо він має набір компетентностей які дозволяють пропустити певні дисципні та більше зосередитись на тих, які його цікавлять, він може це зробити.

Ще одним важливим фактором являється те, що студент може змінювати напрямок свого навчання при бажанні. Після кожного закінченого семестру він буде поповнювати набір власних компетентностей і на основі цього набору може починати опановувати обраний напрямок далі або змінити якого на більш цікавий для самого студента.

В результаті ми отримаємо деяку систему прогресу студента, в якій він буде постійно удосконалювати свої знання. Але також слід зауважити, що постійна зміна напрямків навчання є недопустимою, так як в результаті студент може завершити навчання у ВНЗ із набором компетентностей які будуть не взмозі задовольнити хоча б якусь спеціальність на ринку. В такому випадку нам слід обмежити студента в зміні свого напрямку навчання і дозволяти змінювати лише на той, що не сильно відрізняється. Тобто, при зміні напрямку студент повинен задіяти максимум з тих компетентновтей якими він володіє. Зміна напрямку навчання на той, у якого різниця між компетентностями студента та тими які вимагає від нього цей напрямок дуже велика, можлива лише на початкових курсах. В цьому випадку у студента буде можливість повністю опанувати спеціальність до завершення навчання.

4. Висновок

В завершенні слід сказати, що методи побудови навчальних планів, що використовуються зараз є дещо застарілими і не відповідають тепершнім вимогам. На сьогоднішній день процес складання навчальних планів заснований на досвіді працівників ВНЗ та потребує дуже серйозного вдосконалення, що особливо актуально в умовах постійно зростаючих вимог до випускників. З метою інтесифікації навчального процесу у ВНЗ практикується складання студентами своїх індивідуальних навчальних планів. Тобто актуальною темою для роботи є

вдосконалення процесу побудови навчальних планів з урахуванням компетенцій, що були висунуті ринком, компетентностей студента та його особистих побажань. Це дозволить значно підвищити рівень задоволення ринку в спеціалістах, яких він потребує. Також потрібно зауважити, що даний підхід до побудових індивідуальних навчальних планів є надзвичайно гнучким і може швидко адаптуватись до нових вимог які диктує ринок. Ще одним важливим фактором є те, що при використанні запропонованого підходу значно збільшить рівень комунікації між ВНЗ та ринком, що також позитивно вплине на розвиток освіти.

Список літератури

1. Алексеева Л. Н. Формирование гибкого содержания образования и обучения в средних специальных учебных заведениях. Автореф. дисс. ... канд. тех. наук. Москва, 1997.
2. Горбатова Р. Е. Системный анализ деятельности специалиста и моделирование задач подготовки инженерных кадров. Автореф. дисс. ... канд. тех. наук. Томск, 1981.
3. Кристофидес Н. Теория графов. Алгоритмический подход. Москва, 1978.
4. Московиченко А. Л. Дерево целей инженерной деятельности. //Кибернетика и вуз. Выпуск 13. Томск, 1987, с.123-129.

УДК 004.51 612.881

ЖДАНОВА О. Г.
КАЛЬНИЦЬКИЙ Р. І.
СПЕРКАЧ М. О.

ДОСЛІДЖЕННЯ ВИКОРИСТАННЯ ТАКТИЛЬНИХ ДИСПЛЕЇВ У СИСТЕМАХ СЕНСОРНОГО ЗАМІЩЕННЯ

Розглянуто принцип побудови тактильного дисплею та спосіб передачі за допомогою нього візуальної інформації. Наведено основні патерни передачі інформації про зміну рельєфу. Проведено експерименти із навчання зрячих людей розпізнання нерівності поверхні із використанням статичних та динамічних патернів. Також проведено експерименти із використання дисплею для передачі слів шрифтом Брайля.

Ключові слова: МОБІЛЬНІСТЬ ІЗ ВАДАМИ ЗОРУ, ПАТЕРНИ КОДУВАННЯ ДАНИХ, ВІБРОТАКТИЛЬНИ ДИСПЛЕЙ, СИСТЕМА СЕНСОРНОГО ЗАМІЩЕННЯ, ТАКТИЛЬНА ІНФОРМАЦІЯ, ШРИФТ БРАЙЛЯ

Tactile display construction principle and the way of data transmission of visual information are described. The basic patterns of information transmission about surface unevenness are described. The results of experiments to teach sighted people recognize surface unevenness using static and dynamic patterns. Also experiments to use display to transmit words using Braille font have been done.

Keywords: MOBILITY OF VISUALLY IMPAIRED PATTERNS OF CODING DATA VIBROTAKTYLNY DISPLAY SYSTEM REPLACEMENT TOUCH, TACTILE INFORMATION, BRAILLE

1. Вступ

В повсякденному житті кожен із нас поглинає гігабайти інформації тим чи іншим чином і навіть не задумується про складність природи її збору, аналізу та засвоєння. Насправді мозок замкнений у повній тиші та непроглядній темряві [1]. Він намагається певним чином адаптувати інформацію, що надходить до нього від рецепторів, поєднати її, відкоригувати та побудувати загальну картину всього того, що відбувається навколо. Близько 90% усієї зібраної інформації надходить людині за рахунок зору.

Незважаючи на свій недуг, люди із вадами зору (ЛВЗ) бажають повноцінно самостійно пересуватись вулицями та приміщеннями, отримувати ту ж інформацію, яка є буденною для здорових людей, але при цьому на грані досяжної для них. Допомагають їм у цьому добре відомі способи: тростина, собака-поводир, шрифт Брайля. Ці методи достатньо прості, але при цьому є не достатньо гнучкими чи зручними у використанні при певних обставинах. Саме тому питання створення спеціальних засобів вирішення даних проблем є актуальними і по цей день.

На сьогодні існує досить велика кількість різних пристроїв, які намагаються збирати тим чи іншим способом інформацію та передавати її користувачу за допомогою різного роду інтерфейсів. Предметом дослідження даної статті є процес передачі інформації за допомогою вібротактильного дисплею.

2. Аналіз літературних даних та постановка проблеми

При отриманні значних порушень зору, пацієнт не позбавляється можливості бачити – зазвичай він втрачає лише периферійні сенсори чи шляхи передачі до головного мозку, а не «механізм» в цілому. Через це залишається можливість використовувати ті ж ділянки кори для

аналізу інформації, але вже від інших сенсорів, які подразнюються системами сенсорного заміщення (ССЗ), такі як слух або дотик [2].

У звичайній зоровій системі дані, зібрані сітківкою, перетворюються у електричний імпульс, який подається на зоровий нерв, і ретранслюється в головний мозок, який відтворює зображення і сприймає його. ССЗ базуються на заміні ушкоджених елементів і продовженні використання існуючих областей головного мозку у процесі сприйняття. Під час сенсорного заміщення зору неушкоджена сенсорна система ретранслює інформацію для сприйняття зоровою частиною мозку так, що людина може отримати відчуття, ніби вона може «бачити». За допомогою сенсорного заміщення інформація, отримана від однієї сенсорної системи, може досягати структур мозку, які фізіологічно пов'язані із іншими. Сенсорне заміщення дотик-зір передає інформацію від рецепторів дотику шкіри до зорової ділянки кори для інтерпретації та сприйняття. Даний факт підтверджений за допомогою функціональної магнітно-резонансної томографії, під час якої було виявлено, що під час отримання лише тактильної інформації сліпими людьми, зорова ділянка кори головного мозку теж проявляла активність, відповідну тій, яка проявляється під час обробки зорової інформації [3, 4].

Гнучкість мозку надає можливість йому адаптуватись до зміни умов навколишнього середовища, такі як погіршення або повна відсутність одного із чуттів [2].

Зір і дотик поділяє декілька спільних принципів сприйняття. Обидві сенсорні системи побудовані на обробці двовимірних масивів рецепторів, які активно сканують навколишнє середовище [3], що кодують свої виміри в розрізі простору та часу.

Більшість існуючих тактильно зорових ССЗ (ТЗССЗ) перетворюють візуальну інформацію на тактильну, кодуючи яскравість візуальних образів у вібрацію. Сама ідея передачі візуальної інформації за рахунок масиву віброелементів відома ще із 20-их років XX століття, але перша спроба реалізації даної ідеї була здійснена лише у 1963 році Штарківіцем [5] і у 1969 Бак-і-Ріта [6], який і став батьком поняття «сенсорне заміщення». Під час активної передачі інформації на тактильні рецептори важливу роль у чіткості та якості сприйняття відіграє розширення масиву атракторів (віброелементів).

Даний факт створює найбільшу перешкоду в створенні ССЗ – на сьогоднішній день ТЗССЗ зазвичай містять лише десятки атракторів. Рекордсменом є застосунок, запропонований Віселлем у своїй роботі [7], який використовував двовимірний масив із 1000 атракторів. Роздільна здатність ока значно більша ніж, шкіри, тому не можна покладатись на зменшення розміру атракторів та збільшення їх кількості на удільну площу шкіри.

Професор Бак-і-Ріта показав суттєвий результат у своїй роботі [8], застосовуючи електротактильний дисплей для передачі інформації на поверхню язика. Даний спосіб відкриває широкі можливості для достатньо чіткого розпізнання предметів, але при цьому не підходить для потреб орієнтування в просторі. Це можна обґрунтувати часомістким процесом усвідомлення зображення користувачем та електричною природою передачі інформації.

Використання масиву із соленоїдів для формування реального відчуття дотику досить цікавий спосіб оскільки він надає реальне відчуття дотику. Явним та вагомим недоліком є висока робоча напруга соленоїдів (близько 200V), яка може бути небезпечним при персональному використанні під час пересування вулицею.

Використання сервомоторів у процесі передачі інформації робить тактильний дисплей громіздким (з огляду на розміри та способи сполучення елементів), тому обмежує спосіб використання та зменшує можливу кількість інформації, що може бути передана за одиницю часу.

3. Дослідження ефективності методів кодування візуальної інформації

3.1. Про зміну вигляду поверхні перед користувачем на тактильну

З огляду на наведену вище інформацію, розглядатиметься дисплей на основі віброелементів. Наступним питанням є розмір дисплею та способи кодування інформації.

Спочатку розглянемо можливі одиниці інформації. Припустимо, пристрій розпізнання нерівності поверхні (Пристрій), в якому використовуватиметься даний дисплей, дозволяє виявити зміну абсолютної висоти поверхні в часі.

Таким чином можуть бути наступні основні випадки: поява підвищення, заглиблення, похилого спуску, похилого підйому при русі пристроєм та зменшення чи збільшення відстані при нерухомому. Усі інші випадки утворюються за рахунок композиції наведених. Для простоти розглядатимемо випадок руху Пристроєм зліва направо.

Однією із ключових проблем є час навчання, який необхідний буде надалі для ефективного використання тактильного дисплею. В найкращому випадку інформація, що передається, повинна бути якомога більш інтуїтивною для користувача та простою у сприйманні. З огляду на можливість імітування відчуття тиску за допомогою вібрації, очевидним є імітація відчуття поверхні. Відповідно, патерн, що кодуватиме випадок, матиме динамічну природу, щоб відобразити процес зміни рельєфу. Даний підхід потребує затрат часу для відображення одиниці інформації, що може призвести до зменшення її удільної кількості. Як альтернативу варто розглянути прості кодування кожної ситуації у вигляді статичного символу, що дозволить збільшити удільну кількість інформації.

Форм-фактор розроблюваного дисплею грає ледве не ключову роль – саме від нього залежить роздільна здатність інформації, яку може сприйняти користувач за момент часу. Він повинен бути якомога більш компактним, щоб не сковувати рухи та давати можливість використовувати його на різноманітних ділянках шкіри – від внутрішньої ділянки зап'ястка та бедра до поверхні спини та шиї.

Під час експериментів використовуватиметься дисплей розміром 3x3. Даний розмір було прийнято як мінімально допустимий для передачі різного роду інформації та достатньо ефективний (Лі [9]).

Таким чином, кожен елемент дисплею має 3 основні стани: максимальну вібрацію, середню вібрації та стан спокою. Задля забезпечення плавних переходів, також елемент має змогу поступового переходу між кожним з цих станів. Ілюстрація даних станів зображена на рис. 1. Для простоти надалі множину станів усіх елементів на екрані в певний момент часу називатимемо кадром.

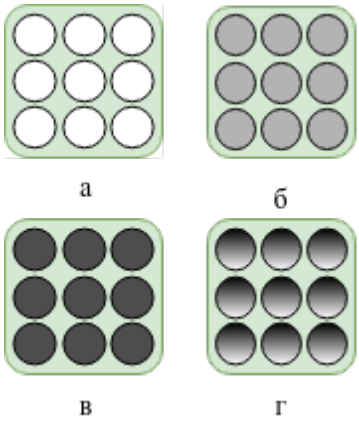


Рис. 1. Вигляд квадратного дисплею 3x3 та його пікселів: а) у стані спокою; б) у середньому стані; в) у стані максимальної вібрації; г) у перехідному стані.

Опишемо патерни кодування даних. При відсутності змін висот дисплей знаходитиметься у стані спокою (всі віброелементи вимкнені – рис. 1 а). При виявленні Пристроєм виступу під час руху ним – у протилежному напрямку відносно руху, поширювалась хвиля вібрації на дисплеї, яка із певним часом після повного його заповнення, поступово рівномірно затухатиме, показуючи перехід на новий рівень висоти та перехід дисплею у стан спокою. Даний процес схематично зображено на рис. 2.

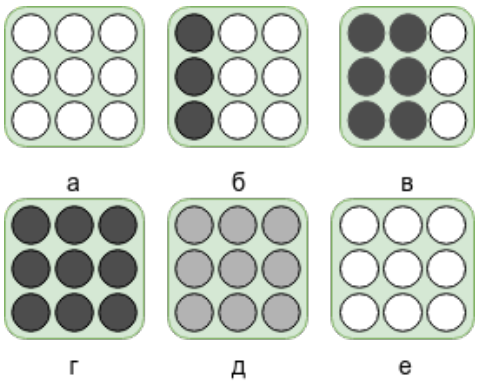


Рис. 2. Зображення кадрів під час виявлення виступу при русі Пристроєм справа наліво: а) стан спокою; б) виявлено виступ, початок поширення хвилі; в) продовження поширення хвилі; г) повне заповнення екрану; д) перехідний процес переходу до стану спокою; е) стан спокою

Аналогічним чином будуються інші динамічні патерни. Усі статичні патерни матимуть вигляд символів на дисплеї. Їхній вигляд залежатиме від вигляду та розміру дисплею. Для прикладу, та ж ситуація, що зображена на рис. 2 матиме статичний патерн як на рис. 3.

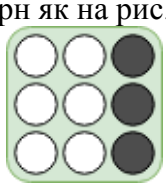


Рис. 3. Зображення патерну виявлення виступу при русі Пристроєм справа наліво

3.2. Дослідження ефективності дисплею для передачі символьної інформації

Сфера застосування даного дисплею не обмежується відображенням інформації про зміну вигляду рельєфу перед користувачем. Окрім цього за рахунок градації вібрації можна відображати достатньо різноманітну інформацію, при цьому роздільна здатність передачі сильно залежатиме від розміру дисплею. Найпростішим, але достатньо істотним та важливим прикладом, є відображення символьної інформації. Передача тексту відбуватиметься на основі використання шрифту Брайля на дисплеї 3x3 із задіянням лише крайніх стовпців. Також за рахунок використання середнього стану можна кодувати додаткові символи, які, зазвичай, потребують додаткового індикатору перед собою, який би позначав їх використання. Приклад наведений на рис. 4.

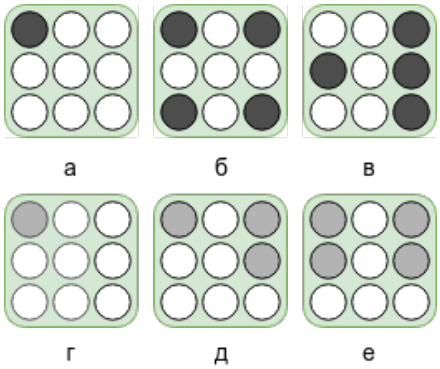


Рис. 4. Приклад використання дисплею для відображення літер шрифту Брайля: а) літера “А”; б) літера “Щ”; в) літера “В”; г) цифра “1”; д) цифра “4”; е) цифра “7”

4. Результати досліджень

У дослідження ефективності описаного підходу брала група людей без вад зору. У якості пристрою було розроблено емулятор, який зчитував напрямок руху руки та запускав відтворення патерну на підключений дисплей згідно нього. Перша група навчалась на основі статичних патернів, друга — динамічних. Обидві групи однаково залучались до експериментів із шрифтом Брайля. Перед проведенням експериментів було описано загальні принципи відображення інформації дисплеєм та продемонстровано декілька разів усі просторові патерни (кожні групі окремо) та перші 10 літер шрифтом Брайля. Кожен учасник проходив експерименти незалежно один від одного і після вільного володіння усіма патернами.

Випробування проводились щодня протягом тижня. Кожне випробування містило всі патерни у 5-ох екземплярах і відтворювались у випадкових послідовностях окремими сесіями для розпізнання просторових патернів та літер шрифтом Брайля. Частка правильно розпізнаних просторових патернів наведено на рис. 5.

Рис 5. Частка правильно розпізнаних патернів усіма учасниками на протязі п'яти днів

По завершенню на шостий день була проведена додаткова сесія для дослідження визначення складних ситуацій (складених патернів): від 1 до 8, кожен містив 5 різних комбінацій. Результати середньої частки правильних відповідей для статичних і динамічних патернів наведено на рис. 6.

По завершенні кожної сесії із розпізнання літер була проведена додаткова сесія із розпізнання слів, що складались із тестових літер.

Рис. 6. Порівняння точності розпізнання статичних та динамічних патернів у складі складних подій

Довжина слів була випадкова від 4 до 6 (кожне слово мало якесь інформаційне навантаження). Результати частки правильно виявлених букв та слів наведено на рис. 7.

Рис. 7. Динаміка росту розпізнання патернів шрифту Брайля

5. Висновки

Необхідно зауважити, що більшість учасників була в тій чи іншій мірі ознайомлена із принципами роботи даного дисплею до його використання. Також усі учасники досконало вивчили вигляд літер шрифтом Брайля перед використання дисплею та могли їх візуально відтворити. Дані факти мають достатньо великий вплив на результати експериментів та повинні братись до уваги при інтерпретації результатів.

Як видно з рис. 5 обидва підходи показують достатньо гарний результат. Більше того — статичні патерни на 4-ий день досягають результату у 100%. Але при цьому, коли на 6-ий день була проведена контрольна перевірка із використанням складних подій, динамічні патерни показали значно більшу стійкість до збільшення складності рельєфу. З огляду на це можна припустити, що використання динамічних патернів для передачі інформації в реальних умовах є більш доцільним.

Отже, даний дисплей можна використовувати для інтеграції із пристроями, що мають текстовий інтерфейс. Точність розпізнання слів в цілому із часом достатньо швидко зросла, що ще раз підтверджує ефективність використання даного дисплею для передачі символьних повідомлень.

Список літератури

1. Can we create new senses for humans? [Електронний ресурс]/D. Eagleman // Нью Йорк. – 2015. Режим доступа: URL: http://www.eagleman.com/blog /eaglemanted2015.

2. Bach-y-Rita, P. Sensory substitution and the human–machine interface [Text]. / P. Bach-y-Rita, S. W. Kercel // TRENDS in Cognitive Sciences. 2003 — N.12(7) — P. 541-546.

3. Bach-y-Rita, P. Brain Mechanisms in Sensory Substitution [Text]. / P. Bach-y-Rita // Academic Press Inc. 1972 — P. 182.

4. Bach-y-Rita, P. Nonsynaptic Diffusion Neurotransmission and Late Brain Reorganization [Text]. / P. Bach-y-Rita // Demos-Vermande. New York, 1995 — P. 215.

5. Starkiewicz W. The 80-channel elektroftalm / W. Starkiewicz, T. Kuliszewski // The 80-channel elektroftalm. Proc. Intern. Congr. Texhnol and Blindness. – New York, 1963. — Vol 1— P. 157.

6. Bach-y-Rita P. Vision substritution by Tactile Image Projection / P. Bach-y-Rita, C. C. Collins, F. A. Saunders, B. White, L. Scadden // Nature.1969. — V. 221. — P. 963-964.

7. Visell, Y. Tactile sensory substitution: Models for enaction in HCI [Text] / Interacting with Computers. Canada, 2008 — P. 38-53.

8. Bach-y-Rita, P. Tactile sensory substitution studies [Text]. / P. Bach-y-Rita // ANNALS-NEW YORK ACADEMY OF SCIENCES. 2004 – P. 83-91.

9. Lee J. Investigating the Information Transfer Efficiency of a 3x3 Watch-back Tactile Display / J. Lee, J. Han, G. Lee // CHI. – Seul, 2015. — P. 1229-1232.

УДК 004.023

СТАРЦЕВ О.Р.

АНАЛІЗ ЗАСТОСУВАННЯ ТЕХНОЛОГІЙ ІНТЕРНЕТУ РЕЧЕЙ ДЛЯ КЕРУВАННЯ ТРАНСПОРТНИМИ ПОТОКАМИ

В даній статті розглянута проблема управління транспортними потоками в умовах мегаполісу та пропонується її вирішення за допомогою використання технологій Інтернету Речей. Досліджується проблема необхідності розробки алгоритму, який дозволить автоматизувати систему транспортної мережі міста та країни в цілому.
Ключові слова: ІНТЕРНЕТ РЕЧЕЙ, ТРАНСПОРТНІ ПОТОКИ, ЕВРИСТИЧНІ АЛГОРИТМИ, АВТОМАТИЗАЦІЯ ПРОЦЕСІВ КЕРУВАННЯ

The article describes the problem of traffic management in the metropolis and offeres its solution through using the Internet of Things's tehnologies. The autor explores the necessity of developing the algorithm for automate transport system.
Keywords: INTERNET OF THINGS, TRAFFIC FLOWS, HEURISTIC ALGORITHMS, AUTOMATION OF MANAGEMENT PROCESSES

1. Вступ

Розвиток автомобільного транспорту України та світу в цілому актуалізують завдання управління рухом автомобільного транспорту. Особливо, в умовах мегаполісів, де кількість автотранспорту різного виду вкрай велика.

Складність завдання управління транспортною системою міста полягає в різноманітності цілей управління. Це і громадський, і вантажний, і приватний автотранспорт. Також необхідно враховувати різні керуючі вказівки, що надходять від різних органів влади і адміністрації, кліматичні умови і стан дорожньої мережі. Величезний вплив на складність завдання надає великий розмір об'єкта управління і робота в режимі реального часу.

З огляду на все вище сказане класичне можна прийти до висновку, що класичне рішення задачі керування тран-спортними потоками в умовах міста в наш час неможливо і недоцільно. У зв'язку з цим все більш цікавими стають рішення, які поєднують сучасні технології збору та обробки інформації. Ці рішення мають бути орієнтовані на використання їх в інтелектуальних транспортних систе-мах.

2. Аналіз транспортного комплексу

Транспортний комплекс є фундаментному для економіки, і він має велике культурне, оборонне, соціальне та економічне значення. І, звичайно ж, він визначає розвитку держави. Адже транспорт використовується в усіх галузях економіки.

Транспорт призначений для фізичного переміщення тіл у просторі. Крім того вагомим фактором для вибору транспортного засобу є вимога до збереження вантажу або пасажирів.

Якість транспортних послуг визначається нормативно-правовими документами та здоровим глуздом. Для кожного типу транспорту існують свої, окремі документи.

Автомобільний транспорт є значною частиною транспортного комплексу країни і світу в цілому. Середня дальність перевезень показують, що переважно автомобільний транспорт використовується для перевезень на короткі відстані.

Техніко-експлуатаційні особливості автотранспорту залежать від його технічних можливостей. Наприклад, таких як автономність руху одиниці рухомого складу, його універсальність і спеціалізація і так далі.

3. Аналіз автомобільного транспорту

Автомобільний транспорт має, як переваги, так і недоліки. До недоліків варто віднести високу аварійність і високу собівартість перевезень.

Структура автотранспорту визначається наступною класифікацією:

- Вантажний рухомий склад;
- автобуси;
- Легкові автомобілі.

Автомобільні перевезення поділяють на два види - вантажні і пасажирські. Вантажні в свою чергу діляться на комерційні та некомерційні.

За територіальною ознакою перевезення бувають, внутріміськими, приміськими, міжміськими і міжнародними.

Пасажирські діляться на перевезення автобусні, легкові, а також вантажні (вантажними автомобілями, які обладнані для перевезення людей). За формою організації перевезення діляться на маршрутні і разові.

Умови функціонування автомобільного транспорту залежать від реальної обставини на дорозі. Сюди входять такі фактори: дорожні умови, транспортний потік, стан навколишнього середовища і так далі. Від цих умов залежить вибір маршруту і, відповідно, швидкість доставки вантажу або пасажирів.

4. Автомобільний транспорт найближчого майбутнього

Уже сьогодні на прикладі розвинених країн ми можемо передбачити подальший розвиток автомобільного транспорту та транспортної системи в цілому.

Окремі автомобільні компанії демонструють успіхи в створенні автомобілів, які здатні переміщатися з пункту А в пункт Б без допомоги людини.

У місті Пало-Альто, США компанія Alphabet проводить тестування автономного транспорту. Ще в 2010 році Google заявляв про подоланих автомобілем 1600км без участі людини.

Компанія Tesla застосують систему автономного управління по всьому світу. У водіїв, які сидять за кермом автомобілів компанії Tesla є можливість включити функцію автопілота і просто стежити за дорогою і поведінкою автомобіля. До кінця 2017 Ілон Маск пообіцяв, що автомобіль їх компанії проїде від Лос-Анджелеса до Нью-Йорка без участі людини.

Зрозуміло, системам автономного управління ще далеко до ідеалу і час від часу вони потрапляють в дорожньо-транспортні пригоди. У систем автономного управління є проблеми з функціонуванням в засніжених умовах або умовах злив.

Всі ці проблеми можна вирішити і кожна подія з такими автомобілями є внеском в розвиток їх систем. Вже зараз, грунтуючись на статистичні дані, ми можемо заявляти, що система автономного управління автомобілем дозволяє значно знизити аварійність на дорогах. Найчастіше реакція людини на аварійну ситуацію нижче, ніж реакція комп'ютера автомобіля. Людині властиво робити помилки. Комп'ютеру - ні. Звісно, це можливо в умовах якісно написаного і належним чином протестованого програмного забезпечення.

Стрімкий розвиток даного виду транспорту призведе до виключення людини з управління автомобільним транспортом. Й через пару десятків років, а то й менше, на дорогах не залишиться автомобілів, якими керують люди.

Це, в свою чергу, вплине на нашу дорожню мережу. Відпаде необхідність в розмітці, дорожніх знаках, світлофорах і інших віджитих

елементів, необхідних водієві-людині, але не комп'ютера.

Автомобілі будуть обмінюватися інформацією один з одним, а також з антенами, які замінять світлофори. Все це можливо буде об'єднано в єдину мережу. Наприклад, за допомогою бездротових коміркових мереж (WMN).

5. Застосування Інтернету Речей

Описана вище система є яскравим прикладом популярної нині методології обчислювальної мережі фізичних предметів - Інтернет Речей (Internet of Things, IoT). Кожен автомобіль в цій методології може бути "річчю".

Інтернет Речей дозволить розділити дорожню мережу на окремі ділянки - осередки. На окремій клітинці рішення про направлення автомобіля може застосовуватися без участі центральних систем управління. Це знизить навантаження на Центр управління, знизить вимоги до потужності обчислювальних засобів і дозволить приймати рішення з більш високою швидкістю.

Застосування методології Інтернету Речей вимагає внесення змін до математичної моделі управління транспортними потоками.

Проведено безліч досліджень і розроблено велику кількість математичних моделей для управління транспортними потоками в умовах мегаполісу. І всі ці моделі можна застосовувати, якщо внести до них невеликі зміни для обліку комірності дорожньої мережі.

Управління транспортною системою можна розділити на стратегічне управління і оперативне. Останнє орієнтоване на управління транспортними потоками на місцях. У нашому випадку, на управління транспортними потоками в окремих осередках.

Застосування методології Інтернету Речей значно вплине на оперативне управління. І в меншій мірі на стратегічне.

Застосування Інтернету Речей вплине на безпеку і аварійність транспортних систем. В ідеальному варіанті зведе аварійність до нуля. Такий вплив можливо завдяки фактично миттєвої реакції мережі на зміни дорожніх умов, а також, завдяки можливому плануванню на короткі і середні терміни поведінки кожного об'єкта управління.

6. Аналіз завдання управління транспортними потоками

Необхідно визначити, що є об'єктом управління, безліччю станів об'єкта управління, безліччю можливих управлінь, критерієм якості і метою управління.

Об'єктом управління, звичайно ж, є транспортна мережа - вулиці, провулки, автомобілі, автобуси і засоби управління. Об'єкт управління зручно представити у вигляді графа, де вершинами є транспортні вузли, а ребрами між ними вулиці, магістралі і так далі. Для кожного графа можна побудувати свій підграф. Як і для кожної транспортної мережі можна за допомогою декомпозиції побудувати транспортну сіть.

Кожну мережу/підмережу можна описати набором її параметрів - пропускна здатність рядності, допустима швидкість руху, завантаженість і так далі. Набір значень параметрів задається вектором стану елемента графа. Безліч таких векторів утворюють безліч станів всієї мережі.

У кожної транспортної мережі або підмережі може бути скінченна безліч станів. Таким чином всю транспортну мережу ми можемо поставити своїм графом і простором станів.

Управління транспортної мережі - це не що інше, як зміна її станів. Наприклад, за допомогою зміни швидкості руху окремого автомобіля, що призведе до зміни швидкості руху всього потоку. Головним завданням

управління є цілеспрямовані зміни стану транспортної мережі доступними засобами управління.

Кожний засіб управління характеризується великою кількістю параметрів. Наприклад, місцем дії. Можливостями використовуваних засобів визначається безліч управлінь транспортною мережею.

Якість управління можна визначити станом дорожньої мережі після застосування певного управління. Чим ближче стан транспортної мережі до цільового, тим вища якісна оцінка.

Метою управління є оптимізація якості роботи мережі. Наприклад, підвищення швидкості руху транспортного потоку, окремого об'єкта або збільшення пропускної спроможності окремої ділянки дорожньої мережі.

7. Висновок

Аналіз статистичних показників використання автономного транспорту на дорогах загального користування і позитивного застосування методології Інтернету Речей в різних сферах діяльності людини дозволяє зробити висновок про необхідність розробки математичної моделі і алгоритму керування транспортними потоками з застосуванням технологій методології Інтернету Речей.

Також була досліджена класифікація об'єктів управління, а саме транспортних засобів та дорожньої мережі.

Був проведений аналіз системи управління дорожньою мережею та об'єктів транспортної системи, який може бути застосований для побудови системи управління із використанням технологій Інтернету речей.

Список літератури

1. Современное управление : энцикл. справ. / под ред. : Д. Н. Карпухина, Б. З. Мильнера. - М. : Издатцентр, 1997. - Т. 1. - 584 с.
2. Шишков В. И. Экспедиционное обслуживание предприятий и организаций автомобильным транспортом / В. И. Шишков, С. У. Пиньковецкий, Ю. В. Калашников. - М. : Транспорт, 1982. - 224 с.
3. Jain, A. K. Data Clustering: A Review / A. K Jain, M. N. Murty, P. J. Flynn // ACM Computing Surveys. - 1999. - Vol. 31, N 3. - P. 264-323.
4. Kerner, B. S. The Physics of Traffic. Empirical Freeway Pattern Features, Engineering Applications, and Theory / B. S. Kerner. - Berlin : Springer, 2004. - 682 p.

УДК 004.02

ТКАЧЕНКО В.Ю.,
ЖДАНОВА О.Г.,
СПЕРКАЧ М.О.

ЗАСТОСУВАННЯ АЛГОРИТМУ ІМІТАЦІЇ ВІДПАЛУ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ МІНІМІЗАЦІЇ СУМАРНОГО ЗАПІЗНЕННЯ РОЗКЛАДУ ВИКОНАННЯ ЗАВДАНЬ ЗІ СПІЛЬНИМ ДИРЕКТИВНИМ ТЕРМІНОМ ПАРАЛЕЛЬНИМИ МАШИНАМИ РІЗНОЇ ПРОДУКТИВНОСТІ

Задачі теорії розкладів - це широке коло, як правило, NP-повних задач комбінаторної оптимізації, що мають застосування в організації виробництва, в бізнесі, в оптимізації комп'ютерних обчислень, в індустрії сервісу і в багатьох інших областях людської діяльності. Це робить розробку алгоритму, здатного вирішити їх за допустимий час та бажаної якості, складним завданням. В роботі розглянуто особливості застосування алгоритму імітації відпалу як ефективного методу побудови розкладу завдань із загальним директивним терміном на паралельних пристроях з різною продуктивністю з метою мінімізації сумарного запізнення.

Ключові слова: СКЛАДАННЯ РОЗКЛАДУ, ПАРАЛНЛЬНІ МАШИНИ, МІНІМІЗАЦІЯ СУМАРНОГО ЗАПІЗНЕННЯ, АЛГОРИТМ ІМІТАЦІЇ ВІДПАЛУ

Scheduling theory tasks - a wide range, usually, NP-complete combinatorial optimization problems, which are widely used in production, business, optimize computing, service industry and in many other areas of human activity. This makes the development of an algorithm that can solve them acceptable and desirable as a really challenging. Therefore, in this paper, the application features simulated annealing algorithm as an effective method of constructing acceptable schedule of tasks with a common prescriptive period of parallel devices with different performance to minimize the total delay.

Keywords: scheduling, PARALNLNI MACHINES, MINIMIZING THE TOTAL DELAY, SIMULATED ANNEALING

1. Вступ

Задача складання розкладів є однією з найбільш поширених задач, що вирішуються кожною людиною (свідомо чи ні) практично щодня. У загальній постановці вона являє собою процес розподілу деякого скінченного набору подій в часі за умови ресурсних та інших обмежень. Таким чином, проста людина, плануючи свій робочий день, і диспетчер, складаючи розклад занять у вузі або графік робіт на підприємстві, вирішують завдання складання розкладу. Але, якщо в першому випадку задача може вирішуватися інтуїтивно на основі життєвого досвіду, то в другому вона може виявитися занадто складною навіть для групи фахівців. Така ситуація виникає через причетність до розкладу великої кількості людей зі своїми інтересами і вимогами, які необхідно врахувати, задоволення таких вимог часто призводить до конфліктних ситуацій.

У деяких часткових випадках вдалося розробити алгоритми, здатні знайти рішення за прийнятний час. У той же час більшість реальних задач складання розкладу відносяться до класу NP-повних.

Тому з розвитком обчислювальних технологій актуальною науковою задачею є розробка ефективного методу побудови допустимого розкладу завдань із загальним директивним терміном на паралельних пристроях з різною продуктивністю з метою мінімізації сумарного запізнення в умовах сучасного виробництва.

Метою дослідження є підвищення ефективності та якості складання розкладів, за рахунок

зменшення сумарного штрафу за порушення директивного терміну шляхом максимізації моменту запуску пристроїв.

2. Постановка задачі та її характеристики

Задано кількість машин m та множину завдань J, для кожного $j \in J$ відомо тривалість виконання t_j. Всі завдання мають спільний директивний термін D. Машини можуть працювати паралельно і є взаємозамінними у тому сенсі, що кожна з машин може виконувати будь-яке завдання з множини J. Машини відрізняються одна від одної продуктивністю виконання завдань. При цьому можна впорядкувати машини за швидкістю виконання завдання і цей порядок однаковий для всіх завдань: для машини i існує коефіцієнт k_i такий, що тривалість виконання завдання j на пристрої i дорівнює $k_i t_j$. «Еталонним» будемо називати пристрій з коефіцієнтом продуктивності $k = 1$. В цьому сенсі величина t_j є тривалістю виконання завдання j на еталонному пристрої. Величину k_i будемо називати *коефіцієнтом продуктивності*. (Якщо $k_i > 1$, то пристрій i менш продуктивний, ніж еталонний, якщо $k_i < 1$ – більш продуктивний).

Необхідно побудувати розклад σ виконання завдань $j \in J$ на m машинах у якому досягає мінімуму функціоналу:

$$F(\sigma) = \sum_{j \in J} \max[0; C_j(\sigma) - D],$$

де $C_j(\sigma)$ – момент завершення виконання завдання j в розкладі σ.

Всі завдання множини J надходять на виконання одночасно і обслуговуються без переривань [1].

Сформульована задача відноситься до класу NP-складних задач. Вона вирішується за псевдополіноміальний час при $m = 2$.

Пронумеруємо завдання множини $J = \{1,2,...,n\}$ по неспаданню значень t_j і розіб'ємо на (необов'язкові непусті) підмножини $J_1, J_2, ..., J_i, ..., J_m$, які попарно не мають спільних елементів, де J_i - множина завдань, що обслуговуються i-м пристроєм, $i = \overline{1,m}$.

Пошук оптимального розкладу відповідно до теореми, наведеної в [2], можна обмежити розглядом розкладів, при яких кожен пристрій обслуговує завдання в порядку зростання їх номерів.

Розіб'ємо множину завдань J_i на підмножини $P_i(\sigma)$, $S_i(\sigma)$, $Q_i(\sigma)$ таким чином щоб:

$P_i(\sigma)$ –завдання, які не запізнюються в розкладі пристрою i;

$S_i(\sigma)$ –завдання, які запізнюються в розкладі пристрою i, для яких виконуються умови:

$$S_i^H < D,\ C_j > D,\ \forall j \in S_i(\sigma),$$

де S_i^H – момент початку виконання завдання j;

$Q_i(\sigma)$ – завдання, які запізнюються в розкладі пристрою i, для яких виконуються умови:

$$S_i^H > D,\ \forall j \in Q_i(\sigma),$$

$$P = \bigcup_{i=\overline{1,m}} P_i;\ S = \bigcup_{i=\overline{1,m}} S_i;\ Q = \bigcup_{i=\overline{1,m}} Q_i$$

R_i – резерв часу пристрою i в розкладі σ

$$R_i = D - \sum_{j \in P_i(\sigma)} k_i t_j;$$

$\Delta_i(\sigma)$ – запізнення виконання завдання $j \in S_i(\sigma)$ відносно директивного терміну:

$$\Delta_i = \sum_{j \in P_i(\sigma) \bigcup S_i(\sigma)} k_i t_j - D$$

На рисунку 1 зображено приклад розкладу виконання деякої множини завдань із вказанням означених вище множин $P(\sigma)$, $S(\sigma)$, $Q(\sigma)$.

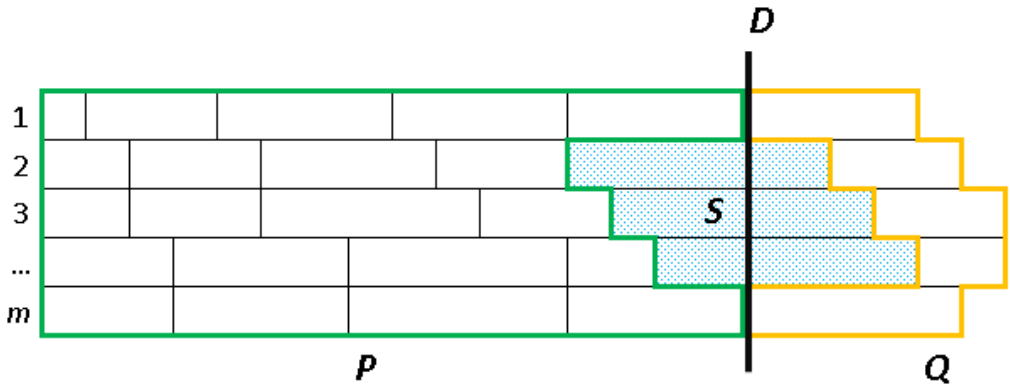


Рис. 1. Приклад розкладу

3.Достатні умови оптимальності

Введемо позначення μ – кількість завдань, що входять у $P \cup S$ ($P \cup S = \{j_1, j_2, ..., j_\mu\}$). Нехай $\overline{\mu}$ ($\underline{\mu}$) – найбільше (найменше) значення μ, при якому $\sigma \in \Psi_{PS}$. Базуючись на методі, розробленому в [3], можна знайти оцінку для величини μ з урахуванням того, що машини мають різну продуктивність.

Введемо позначення k_{t_j} – коефіцієнт продуктивності машини на якій виконується завдання t_j, та μ_i – кількість завдань, що належать $P_i(\sigma) \cup S_i(\sigma)$, звідки $\mu = \sum_{i=1}^{m} \mu_i$.

Також введемо позначення t_{ij} – тривалість виконання завдання j на машині i.

Для оцінки $\underline{\mu}$ уявимо граничний випадок, коли для деякого розкладу σ при $\mu = \mu'$ виконується $\sum_{i=1}^{m} \sum_{j=1}^{\mu_i'} k_i t_{ij} = m \cdot D$, тобто всі завдання із множини $P \cup S$ виконуються у директивний термін (очевидно, що в такому випадку $S = \varnothing$, а отже $P \cup S = P$). Також очевидно, що при $\mu = \mu' + 1$ виконується $\sum_{i=1}^{m} \sum_{j=1}^{\mu_i'} k_i t_{ij} + k_{t_{\mu'+1}} t_{\mu'+1} > m \cdot D$, при цьому $J_{\mu+1} \in Q$.

Логічно припустити, що в такому випадку $\underline{\mu} = \mu'$. Розглянемо $\mu'' = \mu' - 1$:

$$\sum_{i=1}^{m} \sum_{j=1}^{\mu_i''} k_i t_{ij} = \sum_{j=1}^{\mu_i'} k_i t_{ij} - k_{t_\mu} t_\mu;$$

$$t_\mu > 0 \Rightarrow \sum_{i=1}^{m} \sum_{j=1}^{\mu_i''} k_i t_{ij} < \sum_{i=1}^{m} \sum_{j=1}^{\mu_i'} k_i t_{ij} \Rightarrow \sum_{i=1}^{m} \sum_{j=1}^{\mu_i''} k_i t_{ij} < m \cdot D$$

Отримана нерівність протирічить нерівності [1]. Отже, $\forall \mu$ виконується

$$\sum_{i=1}^{m} \sum_{j=1}^{\mu_i} k_i t_{ij} - k_{t_\mu} t_\mu < m \cdot D \le \sum_{i=1}^{m} \sum_{j=1}^{\mu_i} k_i t_{ij}.$$

Введемо позначення $k_{\min} = \min_i \{k_i\}$ та $k_{\max} = \max_i \{k_i\}$.

Очевидно, що для $\forall k_{\min}$ виконується $k_{\min} t_j \le k_i t_j$ для $\forall i = \overline{1, m}$, а для $\forall k_{\max}$ виконується $k_{\max} t_j \ge k_i t_j$ для $\forall i = \overline{1, m}$.

Звідки отримуємо таку оцінку $\underline{\mu}$:

$$\underline{\mu} = \min\left\{ \mu \mid \sum_{j=1}^{\mu-1} k_{\min} t_j < m \cdot D \le \sum_{j=1}^{\mu} k_{\max} t_j \right\}$$

Для оцінку $\overline{\mu}$ проведемо аналогічні міркування і отримаємо:

$$\overline{\mu} = \max\left\{ \mu \mid \sum_{j=1}^{\mu-m} k_{\min} t_j < m \cdot D \le \sum_{j=1}^{\mu} k_{\max} t_j \right\}.$$

Згідно з теоремою наведеною в [2], оптимальним буде розклад $\sigma \in \Psi_P$, у якому завдання з множин $S(\sigma)$, $Q(\sigma)$ рівномірно розподілені між m машинами. Таке можливо за умови кратності m кількості завдань, що належать множинам $S(\sigma)$ та $Q(\sigma)$. Позначимо $|S(\sigma)| = s$, $|P(\sigma)| = p$. Очевидно, що:

$|Q(\sigma)| = n - \mu; \quad \mu = p + s; \ 0 \le s \le m$.

Розклад є рівномірним, якщо $(n - \mu + s) \bmod m = 0$. Під *ідеальним контуром* будемо розуміти такий частковий розклад σ^* виконання завдань з множини $P(\sigma)$, за якого можна гарантувати $\sigma \in \Psi_p$ та його рівномірність. Для часткового розкладу σ^* справедливо наступне: для $m - s$ пристроїв, у яких в повному розкладі $S_i(\sigma) = \varnothing$, в частковому розкладі $\Delta_i = R_i = 0$; для решти s пристроїв у частковому розкладі сумарний резерв $U = m \cdot D - \sum_{j=1}^{p} t_j$. Необхідно забезпечити виконання умови «найменша тривалість виконання завдання $j \in S(\sigma)$ є більшою ніж найбільший резерв машин» для того, щоб повний розклад σ належав класу Ψ_p. При виконанні цієї умови можна побудувати частковий розклад σ^*, що буде ідеальним контуром. В ньому i-та машина $(i = \overline{1,s})$ матиме резерв $1 \le R_i \le U - s + 1$, а також виконується $\sum_{i=1}^{s} R_i = U$.

Тепер наведемо схему спрощення вихідної задачі шляхом побудови ідеального контуру.

Крок 1. Обчислюємо $\underline{\mu}$ та $\overline{\mu}$.

Крок 2. Знаходимо такі пари чисел s та μ, для яких виконується $n - \mu + s \ \vdots \ m$. Обчислюємо $p = \mu - s$.

Крок 3. Для кожної знайденої пари чисел s та μ обчислюємо сумарний резерв

$$U = m \cdot D - \sum_{j=1}^{p} k_{t_j} t_j$$

Якщо $\left\lceil \frac{U}{s} \right\rceil \ge t_{p+1}$, то очевидно, що усі завдання пристрою під номером $p+1$ також входять до множини $P(\sigma)$, тобто поточне значення p змінюється - переходимо до розгляду наступного набору значень. В іншому випадку – переходимо на наступний крок.

Крок 4. Будуємо ідеальний контур. Під ідеальним контуром тут мається на увазі частковий розклад, який складають завдання з множини $P(\sigma) = \{1,2,...,|P(\sigma)|\}$, із деякою конкретною кількістю пристроїв $m - s$, що мають нульові резерви, і s пристроїв, сумарний резерв яких $U = m \cdot D - \sum_{j=1}^{p} k_{t_j} t_j$.

Крок 5. Для того, щоб розклад σ належав класу Ψ_p, формуємо множину $S(\sigma)$ таким чином, щоб виконувалась умова $S_{j_k}^H \le S_{j_l}^H$, якщо $t_{j_k} \le t_{j_i}, \ \forall j_k, \ j_l \in S(\sigma)$.

Крок 6. Формуємо множину $Q(\sigma)$ таким чином, щоб виконувалась умова: Q_i містить ті і тільки ті елементи, котрі відрізняються від $|P \cup S| + i$ на величину, кратну m, $i = \overline{1,m}$.

Якщо буде знайдено такий частковий розклад σ^* виконання завдань з множини $P(\sigma)$, який співпадає з ідеальним контуром, то за побудовою повний розклад σ належатиме класу Ψ_p і буде рівномірним, тобто він буде оптимальним.

Отже, існування такого часткового розкладу σ^* виконання завдань з множини $P(\sigma)$, який співпадає з ідеальним контуром, є достатньою умовою оптимальності повного розкладу $\sigma \in \Psi_p$.

В рамках описаного методу кожному обраному $\mu \in [\underline{\mu}; \overline{\mu}]$ для ідеального контуру можна однозначно поставити у відповідність деякі числа p, s, U. Побудуємо дерево, листами якого будуть можливі ідеальні контури (рисунок 2).

Таким чином, задача звелася до побудови часткового розкладу σ^* виконання завдань з множини $P(\sigma)$, який буде найближчим до деякого ідеального контуру – листа дерева. На практиці така задача матиме набагато меншу розмірність, ніж вихідна. Отже, на її вирішення має бути витрачено менше ресурсів. Тому актуальним є пошук ефективного методу розв'язання отриманої підзадачі.

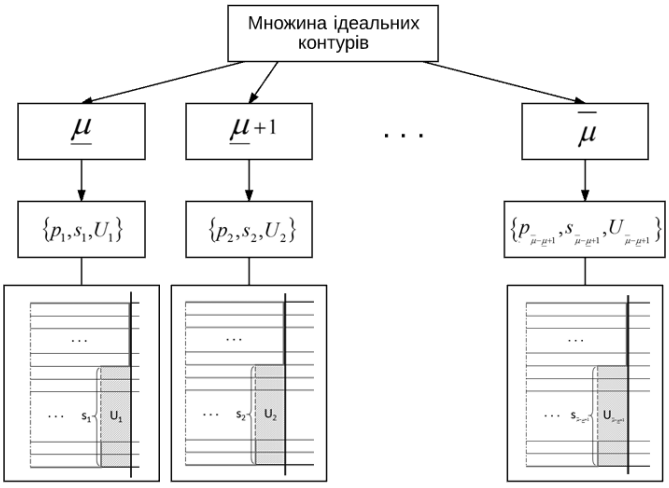


Рис.2. Дерево ідеальних контурів

4.Алгоритм імітації відпалу

Досить актуальним на даний час, для вирішення задачі складання розладу виконання завдань, є метод імітації відпалу (simulated annealing). Алгоритм імітації відпалу - загальний алгоритмічний метод розв'язання задачі глобальної оптимізації, особливо дискретної та комбінаторної оптимізації. У галузі фізики конденсованих середовищ, відпалом називається тепловий процес отримання низьких енергетичних станів тіла в тепловій бані (термостаті). Цей процес складається з двох кроків:

Крок 1. Підвищення температури теплової бані до максимального значення, до твердих розплавів

Крок 2. Повільне зниження температури теплової бані, поки частинки не впорядкуються в основний стан тіла

У рідкій фазі частинки розташовуються випадковим чином, а в основному стані твердого тіла всі частинки розташовані в добре структуровані ґратки, для яких відповідна енергія мінімальна. Основний стан твердого тіла отримується тільки тоді, коли максимальне значення температури досить високе і охолодження здійснюється досить повільно. В іншому випадку, тіло застигне в метастабільному стані, а не в істинному стані. Алгоритм ґрунтується на імітації фізичного процесу, який відбувається при кристалізації речовини з рідкого стану в твердий, у тому числі при відпалі металів. Передбачається, що атоми вже вишикувалися в кристалічну решітку, але ще допустимі переходи окремих атомів з однієї комірки в іншу. Передбачається, що процес протікає при поступовому зниженні температури. Перехід атома з однієї комірки в іншу відбувається з деякою ймовірністю, причому вірогідність зменшується з пониженням температури. Стійка кристалічна решітка відповідає мінімуму енергії атомів, тому атом або переходить в стан з меншим рівнем енергії, або залишається на місці.

Отож, алгоритм імітації відпалу в процесі пошуку оптимального рішення з деякою ймовірністю допускають перехід в стан з більш високим значенням цільової функції. Це властивість дозволяє їм виходити з локальних оптимумів. На рисунку 3 зображено кульку в коробці, внутрішня поверхня якої відповідає ландшафту цільової функції. При сильному струшуванні коробки в горизонтальному напрямку кулька може переміститися з довільної точки в будь-яку іншу.

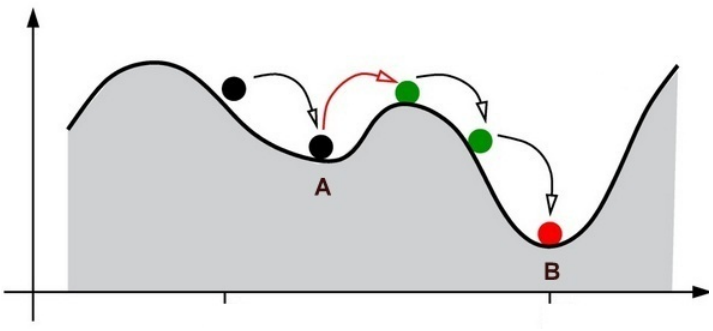


Рис.3. Кулька в коробці

При поступовому зменшенні сили струшування буде досягнута умова, коли ця сила забезпечує переміщення кульки з точки А в точку В, але недостатня для того, щоб кулька могла переміститися з В у А. При подальшому зменшенні сили струшування до нуля кулька зупиниться в точці В – точці глобального мінімуму.

Введемо деякі позначення:

σ_0 – початковий розклад;

σ – поточний частковий розклад;

$K_{\max}$ – максимальна кількість ітерацій;

q – ймовірність переходу в новий стан;

m – кількість пристроїв;

n – кількість завдань;

J – множина всіх завдань;

t_j – тривалість виконання завдання $j \in J$.

Загальну ідею алгоритму імітації відпалу для задачі складання розкладу можна представити наступною схемою:

Крок 1. На першій ітерації генерується деякий коректний початковий розклад σ^0, який вважається поточним розв'язкам задачі ($\sigma = \sigma^0$).

Крок 2. Задаються початкові, високе значення температури T^0 і операція мутації розкладу. Мутація розкладу може бути забезпечена наступними діями:

- зміна порядку виконання завдання на машині;
- зміна машини на якій виконується завдання;
- обмін місцями в розкладі двох завдань.

Крок 3. На основі введених операцій мутації і поточного розв'язку генерується новий коректний розклад σ', який злегка відрізняється від попереднього.

Крок 4. Рахується зміна цільової функції $\Delta F = F(\sigma') - F(\sigma)$:

- якщо $\Delta F <= 0$, новий варіант розкладу стає поточним $\sigma = \sigma'$;
- якщо $\Delta F > 0$ (розв'язок погіршився), то новий розкладу стає поточним тільки з певною ймовірністю. Відповідно в іншому випадку попередній розклад зберігається в якості поточного. Таким чином, допускається збільшення цільової функції, але ймовірність цього зменшується з часом. Такий підхід дозволяє долати локальні екстремуми.

Крок 5. Викликається функція зміни поточної температури. Температура і, відповідно, ймовірність прийняти в якості поточного розклад з більшим значенням цільової функції зменшується з кожною ітерацією. Поступово зі зменшенням температури її значення повинно прямувати до мінімуму.

Крок 6. Якщо не виконано критерій зупинки, то перейти до **Крок 3**. В якості критерію зупинки можуть бути прийняті наступні:

- виконання заданого числа ітерацій;
- виконання заданого числа ітерацій без поліпшення значення цільової функції.

Висновки

У даній статті було розглянуто постановку задачі мінімізації сумарного запізнення розкладу виконання завдань зі спільним директивним терміном паралельними машинами різної продуктивності. З урахуванням особливостей задачі, було наведено достатні умови оптимальності, запропоновано нижню та верхню оцінки кількості завдань μ, а на основі цього було визначено кількість можливих варіантів контурів, які утворюють дерево ідеальних контурів. Враховуючи отриманий результат було розроблено адаптований алгоритм імітації відпалу для вирішення поставленої задачі. Застосування модифікованого алгоритму дозволяє швидше отримувати розв'язок, а також дозволяє при потраплянні у локальні оптимуми продовжувати пошук і прямувати до глобального оптимуму.

Список літератури

1. Основы системного анализа и проектирования АСУ: Учебн. пособие. / А.А. Павлов, С.Н. Гриша, В.Н.Томашевский и др. Под общ. ред. А.А.Павлова, К.: Выща шк.,– 1991.– 367 с.
2. Танаев В. С. Введение в теорию расписаний / В. С. Танаев, В. В. Шкурба. – М.: Наука, 1975. – с. 256.
3. Ткаченко В.Ю., Ткаченко Н.В., Жданова О.Г., Сперкач М.О. Задача мінімізації сумарного запізнення розкладу виконання завдань зі спільним директивним терміном паралельними пристроями//Системний аналіз та інформаційні технології: матеріали 19-ї Міжнародної науково-технічної конференції SAIT 2017, Київ, 22-25 травня 2017 р. – К.: ННК “ІПСА” НТУУ “КПІ”, 2017. – С. 126 — 127.

УДК 519.21

ПРОМСКИЙ И.Л.,
ГУЛЯНИЦКИЙ Л.Ф.

ПРИМЕНЕНИЕ РАЗЛИЧНЫХ РЕАЛИЗАЦИЙ АЛГОРИТМА ОПТИМИЗАЦИИ МУРАВЬИНОЙ КОЛОНИЕЙ В ЗАДАЧЕ ПРОГНОЗИРОВАНИЯ ТРЕТИЧНОЙ СТРУКТУРЫ ПРОТЕИНОВ

Рассматривается оптимизационная задача вычислительной биологии – прогнозирование структуры протеина по последовательности аминокислотных остатков. Рассмотрено использование различных модификаций алгоритма оптимизации муравьиной колонией в применении к гидрофобно-полярной модели на плоской и трёхмерной треугольных решётках. Представлены результаты исследований эффективности данных модификаций.

The article addresses one of optimization problems of computational biology - i.e., the prediction of the protein structure by the sequence of amino acid residues. The research reviews the application of various modifications of the Ant Colony Optimization algorithm in a hydrophobic-polar model on flat and three-dimensional triangular lattices. The study presents the results of effectiveness assessment of the before mentioned modifications.

Ключевые слова: комбинаторная оптимизация, прогнозирование третичной структуры протеина, методы оптимизации муравьиными колониями, эвристики, NP-сложность.

1. Введение

Задача прогнозирования третичной структуры протеина по последовательности аминокислотных остатков – одна из самых важных и сложных задач вычислительной биологии. Поскольку процессы свертывания протеинов изучены не в полной мере, исследователями предложен ряд упрощенных моделей, которые базируются на физических свойствах молекул и среды, в которой происходит процесс свёртки, которые приводят к задачам комбинаторной оптимизации. Но даже в упрощённых моделях предложенные точные алгоритмы не решают задачи за удовлетворительное время даже для последовательностей длины 50, что дает основания для разработки приближенных алгоритмов. В настоящее время для решения возникающих задач предложен ряд таких алгоритмов популяционного типа [1,2,3]. Необходимость в получении более точных решений побуждает к разработке новых детерминированных и метаэвристических алгоритмов, модификации одного из которых, алгоритма оптимизации муравьиной колонией, и являются предметом представленных исследований.

2. Метод оптимизации муравьиной колонией

Одним из современных стохастических методов комбинаторной оптимизации является алгоритм оптимизации муравьиными колониями [4]. Его общая схема показана на рис. 1.

```
procedure ACO
    initialise pheromone trails;
    while (termination condition not satisfied) do
        construct candidate conformations;
        perform local search;
        update pheromone values;
    end
end
```

Рис. 1. Псевдокод классического алгоритма ОМК

В реальном мире муравьи (первоначально) ходят в случайном порядке и по нахождению продовольствия возвращаются в свою колонию, прокладывая феромонами тропы. Если другие муравьи находят такие тропы, они, вероятнее всего, пойдут по ним. Вместо того, чтобы отслеживать цепочку, они укрепляют её при возвращении, если в конечном итоге находят источник питания. Со временем феромонная тропа начинает испаряться, тем самым уменьшая свою привлекательную силу. Чем больше времени требуется для прохождения пути до цели и обратно, тем сильнее испарится феромонная тропа. На коротком пути, для сравнения, прохождение будет более быстрым и, как следствие, плотность феромонов остаётся высокой. Испарение феромонов также имеет свойство избежания стремления к локально-оптимальному решению. Если бы феромоны не испарялись, то путь, выбранный первым, был бы самым привлекательным. В этом случае исследования пространственных решений были бы ограниченными. Таким образом, когда один муравей находит (например, короткий) путь от колонии до источника пищи, другие муравьи, скорее всего пойдут по этому пути, и положительные отзывы в конечном итоге приводят всех муравьёв к одному, кратчайшему, пути.

В основе алгоритма лежит поведение муравьиной колонии - маркировка более удачных путей большим количеством феромона. Работа начинается с размещения муравьёв в вершинах графа, затем начинается движение муравьёв - направление определяется вероятностным методом, на основании формулы вида:

$$P_i = \frac{l_i^q \cdot f_i^p}{\sum_{k=0}^{N} l_k^q \cdot f_k^p}$$

где:

P_i - вероятность перехода по пути i,

l_i - величина, обратная весу -го перехода,

f_i- количество феромона на -м переходе,

q - величина, определяющая «жадность» алгоритма,

p - величина, определяющая «стадность» алгоритма и

$q + p = 1$.

После того, как муравьи закончат движение, происходит обновление матрицы феромонных следов, обычно с использованием формулы вида:

$$f_{i,j} = (1 - \rho) f_{i,j} + \Delta f_{i,j}$$

где:

$f_{i,j}$ – количество феромона на дуге i, j,

ρ – скорость испарения феромона,

$\Delta f_{i,j}$ – количество отложенного феромона.

3. Применение алгоритмов локального поиска в ОМК

Решение, найденное с помощью ОМК, не является точным и даже может быть одним из худших, однако, в силу вероятностности решения, повторение алгоритма может выдавать (достаточно) точный результат.

А так же часто реализации алгоритма ОМК дополняют различными алгоритмами локального поиска для улучшения результата, в пределах некоторой ограниченной окрестности.

Обозначим множество допустимых сверток D. Определим расстояние между свертками $\rho(s, v)$ как количество символов, на которое отличаются их кодировки s и v. Обозначим $O_\delta(s)$ окрестность точки s размера δ , т.е. $O_\delta(s) = \{v | \rho(s, v) \leq \delta\}$ – энергию свертки s. В приведенных обозначениях схему детерминированного локального поиска показано на рис. 2 [5].

```
procedure LocalSearch (s0)
    while (окрестность O1(s0) не просмотрена полностью) do
        v := некоторый вариант из O1(s0);
        if v ∈ D and E(v) < E(s0) then
            s0 := v;
    end while;
    return s0;
end procedure.
```

Рис. 2. Псевдокод алгоритма детерминированного локального поиска

Таким образом, алгоритм гарантирует улучшение текущей свертки в окрестности, если оно существует. Сложности возникают в том, что условие допустимости является достаточно строгим, поэтому, при малых значениях δ в окрестности $O_\delta(s)$ может оказаться малое количество допустимых

вариантов решения (сверток) или не оказаться вообще.

В [6] было предложено использование детерминированного локального поиска единичной глубины для уточнения решения найденного вероятностным алгоритмом. В результате проведенного вычислительного эксперимента было показано, что такой подход может при аналогичных условиях завершения может находить свёртки с меньшими энергиями (до 16%), однако вследствие появления дополнительного стремления к локальным минимумам решение может ухудшиться. Так же в [6] было предложено дополнительно исследовать возможности интеграции алгоритмов ОМК и ДЛП для окрестности $\delta > 1$, а также использование в качестве алгоритма локального поиска других алгоритмов, как детерминированных (метод свёртки стрелой, а также метод ветвей и границ), так и стохастических (таких как алгоритм имитации отжига и G-алгоритм).

4. Изменение методов прокладки и испарения феромона

При реализации метода ОМК [7] в стандартный алгоритм, описанный выше, на каждом цикле расчета добавлены уточняющие элементы – процедуры, осуществляющие модификацию матрицы феромонных следов путем применения весовых коэффициентов, уменьшающих влияние количества отложенного феромона, а также дополнительное обновление указанной матрицы, учитывающее физические свойства рассматриваемой молекулы протеина. Применение весового коэффициента обусловлено необходимостью снизить риск возможной ошибки при нахождении оптимального решения задачи. Вместе с тем, описанный алгоритм содержит ряд параметров, определяемых эвристически. При этом оптимальный выбор таких параметров может привести к существенному улучшению эффективности выбранного метода решения задачи.

С учетом особенностей задачи поиска оптимальной свёртки протеиновой молекулы [7] предлагается альтернативный вариант обновления феромонной матрицы. Его идея состоит в том, чтобы усиливать только те феромонные дорожки, которые влияют на энергию заданной молекулы. Для этого использовалась следующая процедура: был введен массив ϕ длиной $N-2$ и заполнен нулями. Пусть l, m – номера остатков, образующих гидрофобную связь, $l < m$. В образовании этой связи принимают участие элементы $r_l, r_{l+1}, \dots, r_{m-1}$. Производится корректировка массива ϕ для этой связи так:

$$\phi[i] = \phi[i] + 1, i = \overline{\max\{1, l-1\}, m}$$

и операция повторяется для всех гидрофобных связей в структуре $v = r_2 r_3 \dots r_{N-1}$ после чего обновляется феромонная матрица:

$$\Phi[\mathrm{k}_{\mathrm{i}+1}, \mathrm{i}] = \Phi[\mathrm{k}_{\mathrm{i}+1}, \mathrm{i}] + \phi[i]^r, i = \overline{1, N-2}$$

.Согласно [7] Такая схема позволяет более точно учитывать приемлемость той или иной части структуры молекулы.

5. Использование априорных оценок

В [7] также была рассмотрена реализация алгоритма ОМК, в которой, кроме изменённой схемы прокладки феромонного следа, было реализовано использование априорных оценок при создании свёртки. В соответствии с этими оценками вероятность перехода будет определятся по формуле вида:

$$P(r_i = dir_j) = \frac{\Phi[j,i-1]^{\alpha} * Est(r_2 r_3 \dots r_{i-1}, dir_j)^{\beta}}{\sum_{j=1}^{n} \Phi[j,i-1]^{\alpha} * Est(r_2 r_3 \dots r_{i-1}, dir_j)^{\beta}},$$

где α и β – параметры алгоритма, $Est(r_2 r_3 \dots r_{i-1}, dir_j)$ – априорная оценка, а Ф – матрица феромонных следов. Для ее вычисления предполагается, что следующая аминокислота располагается по направлению dir_j и подсчитывается количество гидрофобных (n_H) и полярных (n_P) остатков в соседних с ней узлах. Чтобы образовывались новые связи, гидрофобные остатки целесообразно располагать рядом с гидрофобными или со свободными узлами (есть шанс заполнить их гидрофобными остатками при дальнейшем построении молекулы), а полярные не ставить рядом с

гидрофобными (оставляя место для возможных новых соединений).

Такая интерпретация аналогична естественному поведению молекулы: полярные остатки стремятся к соседству с полярным окружением – водой или другими полярными остатками. Введены параметры

$0 \le \eta_{HP} < \eta_{H0} < \eta_{HH},\ \eta_{HP} + \eta_{H0} + \eta_{HH} = 1,$

$0 \le \eta_{PH} < \eta_{P0} \le \eta_{PP}, \eta_{PH} + \eta_{P0} + \eta_{PP} = 1,$

и определена оценка $Est(r_2 r_3 \dots r_{i-1}, dir_j)$ как показано в (1).

$$Est(r_2 r_3 \dots r_{i-1}, dir_j) = \begin{cases} n_P * \eta_{HP} + n_H * \eta_{HH} + (n - n_P + n_H) * \eta_{H0}, O_{i-1} = H \\ n_P * \eta_{PP} + n_H * \eta_{PH} + (n - n_P + n_H) * \eta_{P0}, O_{i-1} = P \end{cases} \quad (1)$$

Приведенная оценка является обобщением известных ранее – в литературе часто встречается комбинация $est_{PH} = est_{P0} = est_{PP} = \frac{1}{3}$, $est_{H0} = est_{HP} = 0$, $est_{HH} = 1$. В [8] для вычислительного эксперимента были выбраны значения $est_{PH} = 0.2,\ est_{P0} = est_{PP} = 0.4,\ est_{H0} = 0.25, est_{HP} = 0.15$, $est_{HH} = 0.6$.

Из приведённых в [7] результатов вычислительного эксперимента следует, что данная модификация алгоритма прокладки феромонных следов значительно повышает эффективность поиска оптимальной свёртки.

6. Заключение

Как показано в рассмотренных публикациях, метод оптимизации муравьиной колонией может эффективно решать задачу поиска оптимальной свёртки протеиновых молекул. Также, модифицируя различные его элементы, возможно создание ещё более эффективных мета-эвристик.

Открытыми остаются такие вопросы:

- выбор наиболее подходящих для решения данной задачи модификаций;
- подбор оптимального набора значений параметров алгоритма, при котором он максимально эффективен;

Список литературы

1. R.Unger, J.Moult. Genetic Algorithms for protein folding simulations // J. of Molecular Biology.–1993. – 231(1). –P.75–81.
2. A.Piccolboni, G.Mauri. Application of Evolutionary Algorithms to Protein Folding Prediction // Artificial Evolution. – 1997. – P. 123–136.
3. M.Khimasia, P.Coveney. Protein Structure Prediction as a Hard Optimization Problem: the Genetic Algorithm Approach // Molecular Simulation. – 1997. – 19. – P. 205–226.
4. M.Dorigo M.,T. Stützle. Ant Colony Optimization. – Cambridge: MIT Press, MA, 2004. – 348 p.
5. И.В. Сергиенко. Математические модели и методы решения задач дискретной оптимизации. – К.: Наукова думка, 1985. – 384 с.
6. Л.Ф. Гуляницкий, В.А. Рудык. Разработка и исследование алгоритмов решения задачи прогнозирования третичной структуры протеина// Intelligent Support of Decision Making (Eds. K. Markov et al.) / Int. Book Series “Information science and computing”. N 10. – Sofia: ITHEA, 2009, p.97-103
7. В.А. Рудык. Анализ эффективности алгоритмов ОМК для решения задачи о сворачивании протеинов // Теорія оптимальних рішень: Зб. наук. пр. — 2012. — № 11. — С. 66-72. — Бібліогр.: 7 назв. — рос.

УДК 004.9:519.7

СТАНГРІТ Н.С.

ПОБУДОВА РОЗРЯДЖЕНОГО ВОКСЕЛЬНОГО ДЕРЕВА ОКТАНТІВ ДЛЯ ПОШУКУ ШЛЯХУ

У статті описано задачу побудови моделі орієнтації. Формалізовано задачу побудови моделі орієнтації. Розроблено метод розв'язання задачі побудови моделі орієнтації. Розроблено метод розв'язання задачі пошуку шляху у цій моделі. Запропоновано програму реалізації моделі. На основі отриманих теоретичних результатів приведені можливості практичного застосування, а також обчислювальні задачі, які при цьому виникають.

This article describes the task of building the orientation model. The author formalize the task of building the orientation model. The scientist develops a method of decision of the task of building the orientation model. The study works out a method of solution of the search problem in this model. The researcher proposes the model implementation program. With outlined theoretical results, there is a description of possible usages of the solution and the problems and challenges that follow after that.

1. Вступ

Пошук шляху штучним інтелектом доволі актуальне питання. Це завдання виникає у таких галузях як розробка ігор, моделюванні деяких процесів. Слід уточнити, що ми говоримо про пошук шляху у віртуальному середовищі, а не реальному. При збільшені масштабу проекту, можуть збільшуватися розмір мап, та кількість віртуальних агентів.

Більшість сучасних методів, які активно використовуються, розроблені для двох вимірного простору, або, у разі трьох вимірного простору, мають не чітку структуру, що ускладнює пошук шляху. Проте деякі агенти мають діяти у повноцінному трьох вимірному просторі. І хоча деякі моделі навігації дозволяють будувати себе в такому просторі, в такому процесі можуть виникнути проблеми, а утворена структура не є чіткою і збільшує час роботи з нею.

Актуальною практикою є розпаралелення систем, так як обчислювані потужності швидше зростають у напрямку використання декількох елементів одночасно, аніж у напрямку зростання потужності лише одного елементу, а знайти процесор, який має лише один потік, важко. Саме тому висунуто умову роботи використовуючи паралельних підхід, тема стає біль актуальною, так як деякі моделі навігації, наприклад – навігаційна сітка, не спроможні працювати в такому режимі, через свою не структуровану внутрішню ієрархію, тобто неможливість розділити основну свою задачу на менші задачі.

Найпопулярніші алгоритми використовують здобутки полігональної моделі, будуючи ніби трьох вимірну модель вільного від перешкод простору. Така модель називається навігаційна сітка. Ця модель може давати погані результати у випадку необхідності вільного переміщення по трьох вимірному просторі.

На відміну від навігаційної сітки, яка використовує полігони, як засіб позначення вільного простору, ми будемо використовувати альтернативний шлях, вокселі. Вокселі – це технологія яка має багато шансів прийти на заміну класичним полігонам. І хоча зараз цей напрямок тільки розвивається, вже існують дослідження пов'язані з ними, які показують, що цей напрям має як свої плюси, так і мінуси.

Тож, сумуючи наведені вище доводи, можемо сказати, що актуальним питанням є розробка моделі навігації, яка буде спроможна працювати в повноцінному трьох вимірному середовищі, точніше буде навіть орієнтована на роботу тільки з ним. Працювати паралельно, як при будуванні такої моделі, так і при використанні її в реальному часі. Підтримувати динамічну перебудову, не гублячи при цьому деталізованість, чи загальну структуру.

2. Постановка задачі

Існують різні моделі, для вираження простору навігації, який використовується для пошуку шляхів. Пошук шляху по таким моделям

поділений на декілька виді: глобальний пошук шляху, та локальний. Глобальний пошук дозволяє агенту рухатись з початкової позиції до кінцевої по оптимальному шляху. Цей оптимальний шлях зазвичай є найкоротшим. Глобальний пошук шляху вимагає інформацію про усі перешкоди на локації. Тоді як локальний пошук шляху дозволяю лише уникати перешкод, на шляху до цілі, і не завжди буває оптимальний.

В усіх моделях існують як плюси, так і мінуси. Можна побачити що для повноцінних випадків трьох вимірного простору наявно менше варіантів, і більшість з них має проблеми з швидкодією та витратами пам'яті. Також, у випадку Об'ємної сітки навігації лише її варіант з об'ємними многогранниками дає можливість оперувати повним простором, на відміну від варіанту, де полігони можуть лежати в різних площиннах. Варіант Воксельної сітки виглядає більш оптимістичним над Сіткою навігацію, то му що дає змогу створювати структури с більш жорсткою ієрархією, аніж динамічне розташування елементів в Сітці навігації та біль підходить до вимоги динамічної зміни.

Тож, якщо провести дослідження по цій моделі, і змінити деякі її принципи, такі як: розташування вузлів; обробка не вільного, а зайнятого простору; обробка пустих просторів не в режимі графу, можна очікувати отримувати результати, які будуть ліпшими за стандартну модель.

Таким чином, наша мета - пошвидшення пошуку шляху в моделі навігації, зменшення її розміру, можливість динамічної зміни.

Для досягнення мети необхідно виконати наступні завдання: розробити модель навігації, яка буде відповідати меті; розробити програмну реалізацію алгоритмів та моделей у вигляді, що може використовуватися при пошуку шляху для штучного інтелекту; виконати аналіз отриманих результатів.

3. Обґрунтування розв'язання

Щоб реалізувати заявлені задачі необхідно визначити в якому виді буде представлена модель. Так як ми будемо використовувати Воксельну сітку, за основу можна використати дерево октанів, чи k-мірне дерево.

Основна ідея , це ділення вузла дерева на 8 потомків, розташованих, як октанти в Декартовій системі координат. Так як нас хвилює використання пам'яті, слід одразу взяти модифіковану версію такого дерева – Розряджене дерево октанів, яка не створює потомків, коли немає необхідності і такі потомки не несуть корисної інформації. [1].

K-мірне дерево – структура даних, яка служить для організації точок в k-мірному просторі. Являє собою варіант бінарного дерева пошуку.

Така структура дозволяє швидку шукати сусідів елементу, або сам елемент по його позиції. Так пошук йде в вузол, центр якого знаходиться найближче до позиції, по якій ми шукаємо. Складність такого пошуку буде наступним:

$$O\left(k * n^{1\frac{1}{k}}\right), \text{де } k - \text{розмірність простору}, n - \text{кількість вузлів} \quad (1)$$

Для того, щоб можна було розташовувати модель в просторі під будь-яким кутом, та зменшити витрату пам'яті на зберігання координат, необхідно використовувати матрицю трансформації. Вона дозволить зберігати усі координати цілочисенними.

Одна з основних відмінностей досліджуваної моделі в тому, що сама вона містить інформацію лише про перешкоди, а не вільний простір. Тож методи побудови графу, які використовуються в інших моделях тут не допоможуть.

Розробка моделі навігації вкрай необхідна, оскільки дозволяє проводити роботу в трьохвимірному середовищі, що надає можливість використовувати більш структуровану ієрархію та можливість динамічної зміни. Для реалізація запропонованих алгоритму та моделі доцільно використовувати спеціальне програмне забезпечення, яке повинно відповідати наступним вимогам. По перше – кроссплатформенність. Штучний інтелект зазвичай працює на операційних системах, які не використовуються в повсякденному користуванні звичайних користувачів. Зазвичай це різні дистрибутиви Linux, або навіть біль спеціалізовані системи. В вихідному варіанті буде підтримуватися лише двигун Unity, іншим системам можна буде використати API, та утворити свій графічний інтерфейс, за такої необхідності. Unity допоможе продемонструвати роботу система на різних середовищах, показати приклад використання API, та звільнити нас від необхідності власних систем, які не є частиною цієї роботи (фізика, графіка, UI).

По друге – многопоточність. Зараз вже майже відсутні системи, які працюють на одноядерних

процесорах. Тож важливим фактором є можливість системи працювати паралельно, як в плані роботи з основним потоком, так і в плані внутрішньої роботи.

По трете – генерація моделі на льоту. Генерація має займати прийнятний, для роботи в реальному часі час, та дозволяти використовувати так звану ліниву підгрузку. Лінива підгрузка означає, що в деякий момент часу генерується лише та частина моделі, яка необхідна, з подальшим обчисленням усієї моделі

4. Проведення експерементів

Отже, маємо V, який представляє собою куб, де ми знаємо дві вершини – V^A, V^B, що розташовані на протилежних вершинах довільної діагоналі цього кубу. Центр V буде наступним:

$$V^C = \frac{V^A + V^B}{2} \quad (2)$$

А розмір:

$$V^S = |V^A - V^B| \quad (3)$$

V може містити, або не містити, 8 менших V. Менші V будемо записувати як V[1..8]. Індекси цих вершин строго відповідають їх розташуванню, як показано на малюнку 3.1.

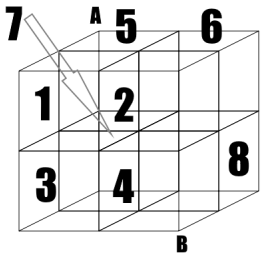


Рисунок 1. Розбиття V на менші V

Також маємо множину C, яка включає в себе всі коллайдери довільної форми. Умова, яка необхідна, щоб поділити V на менші V наступна:

$$V \subset \forall C \quad (4)$$

Тож, якщо V перетинається з будь-яким елементом множини C, і:

$$|V^S| > S_{min} \quad (5)$$

де S_{min} – мінімальний розмір будь-якого вокселя, то воксель розбиватися на 8 менших вокселів за наступним правилом.

V[5]: $V[5]^A = V^A$, $V[5]^B = V^C$;

V[4]: $V[4]^A = V^C$, $V[4]^B = V^B$;

V[7]: $V[7]^A = V^A - (0, V_Y^S/2, 0)$, $V[7]^B = V^C - (0, V_Y^S/2, 0)$; (

V[2]: $V[2]^A = V^C + (0, V_Y^S/2, 0)$, $V[2]^B = V^B + (0, V_Y^S/2, 0)$;

V[1]: $V[1]^A = V^A - (0, 0, V_Z^S/2)$ $V[1]^B = V^C - (0, 0, V_Z^S/2)$;

V[8]: $V[8]^A = V^C + (0, 0, V_Z^S/2)$ $V[8]^B = V^B + (0, 0, V_Z^S/2)$;

V[3]: $V[3]^A = V^A - (0, V_Y^S/2, V_Z^S/2)$ $V[3]^B = V^C - (0, V_Y^S/2, V_Z^S/2)$;

V[6]: $V[6]^A = V^C + (0, V_Y^S/2, V_Z^S/2)$ $V[6]^B = V^B + (0, V_Y^S/2, V_Z^S/2)$; (6)

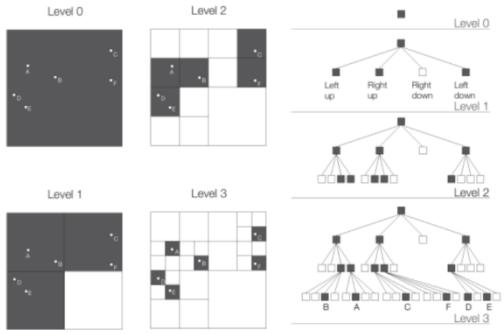


Рисунок 2. Приклад розбиття на двовимірному просторі

Таким чином розділення кожного вокселя виконується рекурсивно, доки воксель не буде перетинатися з будь-яким коллайдером, чи його розмір не буде менше за деякий заданий мінімальний розмір.

Для побудови графу необхідні найменші вокселі, тобто такі, у яких V[1..8] = ∅. Якщо воксель не моє перетину з жодним коллайдером, то в залежності від обраної моделі він може не прийматися до уваги. Кожна сторона вокселя обробляється в залежності від обраної моделі побудови. На виході обробки маємо множини N[1..8], які зберігають сторону S, та утворені нею вузли S[1..5](Кожна сторона утворює п'ять вузлів). Основним параметром моделі є вектор гравітації, чи його відсутність, та максимальний кут нахилу площини до нього.

Якщо довжина вектору гравітації G не дорівнює нулю, або максимальний кут нахили до гравітації$\alpha_{max} > 0$, виконується тест сторін:

$$acrcos\left(\frac{NG}{|N||G|}\right) > \alpha_{max}, \quad (7)$$

де N – нормаль деякої сторони.

Утворивши вершини методом, який наведено вище, можуть утворюватися зайві вузли. Їх очистка виконується наступним чином.

Усі вокселі сортуються по розміру, від найбільшого, до найменших, - B[1...∞]. Починаючи з найбільших вокселів знаходяться дублікати вершин, до уваги беруться протилежні сторони вокселів, такі дублікати видаляються.

Коли усі дублікати вершин, що знаходяться в одно розмірних вокселя були видалені, починається видалення вузлів, незважаючи на направлення сторони та розмір вокселя. Усі

вершини тестуються на дублікати, та зайві видаляються.

Такий дворазовий прохід дозволяє очистити вузли від зайвих дублікатів на першому проході, який більш швидкий, ніж повний пошук дублікатів по всіх вузлах. Та видалити вузли, розташовані в середині перешкод.

Для застосування рейкастів та покращення швидкодії алгоритму побудови графу, нам необхідно знаходити сусідів деякого вокселя. Метод знаходження сусідів базується на роботі Vörös[35], який пропонує знаходження сусідів використовуючи їх спільних батьків. В цій роботі використовуються деякі модифікації алгоритму Ворупа. По-перше, Ворус обчислює сусідів, які менше, рівні чи більші.

У цьому дослідження лише сусіди, які рівні, чи більші за даний. По-друге, в метод Ворупа два вузла є сусідами, якщо вони спільно використовують одну грань. У цьому дослідженні, два вузла можуть бути сусідами, якщо вони мають спільну: грань, ребро чи вершину. Так що метод Ворупа розширюється, щоб обчислити усі 24 сусіда, рис.3.

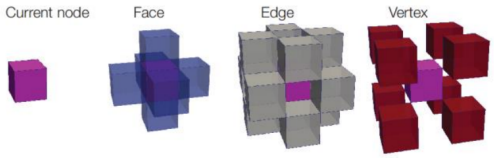


Рисунок 3. Приклад сусідів вокселя по спільним елементам.

Основна ідея для знаходження сусідів по грані: сусіди по грані вузла V мають спільні сторони з сусідами, які мають спільні сторони з основним вузлом. На першому етапі, знаходяться чотири сусіди по стороні. На другому етапі знаходимо сусідів по стороні, які розташовані по осям X і Z. На третьому кроці сусіди по Z і X обчислюють своїх сусідів по стороні за напрямками Y та X.

Метод для обрахування сусідів по вершинам схожий на пошук сусідів по граням. Вершинні сусіди вокселя мають зв'язок сторони з сусідами по граням основного вузла. На першому етапі, обчислюються сусіди по грані. Потім вибираються сусіди по грані, які були обчислені в напрямку X. З обраного вузла, обчислюються сусіди по стороні в напрямку Y.

Спочатку розглянемо двовимірний метод рейкасту, який надалі переведемо в тривимірний. Промінь r задається як (p, d), де p = (p_x,p_y) – точка в початку проміння, і d = (d_x, d_y), який є нормальним вектором напрямку проміння. Для кожного не цілочисленного t > 0 існує точка (x_r(t), y_r(t)) на промені. Де x_r і y_r дві функції, які визначаються наступним чином:

$$x_r(t) = p_x + td_x;$$
$$y_r(t) = p_y + td_y; \quad (8)$$

Воксель *o* в *a* дереві це множина точок в квадраті. Формально, *o* це набір точок (x, y), таких, що $x_0(o) \le x \le x_1(o)$ і $y_0(o) \le y \le y_1(o)$, де x_0, x_1, y_0 і y_1 скалярні велечини, які визначають позицію кожного вокселя. Також введемо функцію *s(o),* яка визначає довжину грані вокселя.

З наведених вище позначень виведемо, що перетин між *r* і *o* виникає, якщо є хочь один *t*, при якому

$$x_0(o) = x_r(t) < x_1(o);$$
$$y_0(o) = y_r(t) < y_1(o); \quad (9)$$

Запропонований алгоритм називається параметричним алгоритмом, тому що всі обчислення використовують значення *t*, такі як $(x_r(t), y_r(t))$, це точка на межі вокселя. Для вокселя *o*, та проміння *r*, $t_{x0}(o,r)$,$t_{y0}(o,r)$, і $t_{x1}(o,r)$,$t_{y1}(o,r)$ визначенні як параметри проміння, при яких він перетинає границі вокселя. Формально ці значення визначаються наступним чином:

$$\left.\begin{aligned} x_r\left(t_{xi}(o,r)\right) &= x_i(o) \\ y_r\left(t_{yi}(o,r)\right) &= y_i(o) \end{aligned}\right| \; i \in \{0,1\} \quad (10)$$

Використовуючи зворотні функції x_r і y_r параметричні величини можуть бути визначені наступним чином:

$$\left.\begin{aligned} t_{xi}(o,r) &= (x_i(o) - p_x)/d_x \\ t_{yi}(o,r) &= (y_i(o) - p_y)/d_y \end{aligned}\right| \; i \in \{0,1\} \quad (11)$$

Де p_x s p_y початок половини лінії, а d_x і d_y нормалізований вектор напрямку. Ці величини обчислюються для кожного єлемента дерева. Спочатку її отримує базовий вузол, а потім інкрементні обчислення виконуються для кожного дочірнього вузла.

Знаючи визначення вузла і променя, легко отримати перетин між променем *r* і *o*. Якщо перетин відбувається інтервал закрит з ліва, та відкрит справа. Беручи до уваги цю інформацію, ми можемо змінити використовуючи параметри променя. Це визначення можна скоротити ще більше, якщо визначити t_{min} і t_{max} для вузла *o* і променя *r* як:

$$\begin{aligned} t_{min}(o,r) &= \max(t_{x0}(o,r), t_{y0}(o,r)) \\ t_{max}(o,r) &= \min(t_{x1}(o,r), t_{y1}(o,r)) \end{aligned} \quad (12)$$

Отже, за наведеною моделлю запропоновано програму, яка буде поставлятися двома способами, як dll бібліотеки, та в відкритому коді.

Dll дозволять користувачам використовувати програму без зайвих труднощів, викликаних необхідності компілювати вихідний код. Тоді як постачання програми в стані відкритого коду дозволить змінити програму під себе.

Програма ділиться на наступні модулі:

- VoxelMovement.Core – головний модуль програми, який надає базові класи, для інтеграції програми до різних середовищ. Та відповідає за найважливіші функції програми, такі як побудова дерева, пошук шляху;
- VoxelMovement.Colliders – модуль програми, який містить в собі визначення підтримуємих коллайдерів, їх поведінку, та інтеграцію з модулем пошуку перетинів;
- VoxelMovement.Physic – містить логіку пошуку перетинів коллайдерів, та вокселів;
- VoxelMovement.Domain.Text – модуль програми, якій містить спеціальні тести, для перевірки правильності відпрацювання інших модулів, надається лише з відкритим кодом.

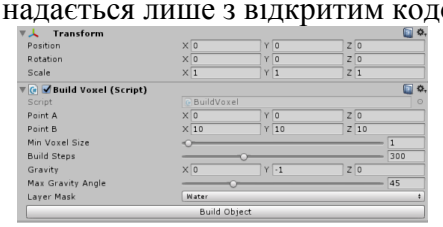


Рисунок 4.. Інтерфейс програми в Unity. Панель налаштувань.

На рис. 4 показано інтерфейс, в разі використання програми з двигуном Unity, а семе показана панель налаштувань, де: Transform – визначає центр початкового вокселя(Position), його кут(Rotation); PointA визначає першу точку вокселя; PointB визначає другу точку вокселя; Min Voxel Size – мінімальний розмір вокселя; Build Steps – кількість ітерацій, який виконає алгоритм будівництва дерева; Gravity – задає вектор гравітації; Max Gravity Angle – максимальний кут сторони вокселя до вектору гравітації; Layer Mask – слої, які будуть задіяні при будівництві моделі; Build Object – кнопка для початку будівництва моделі.

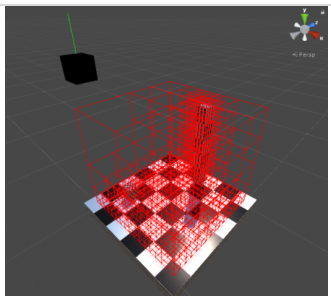


Рисунок 5. Демонстрація вікна редактора.

На рис.5 показана демонстрація відпрацювання програми. Червоним кольором показані вокселі. Чорний куб і зелена лінія демонструють напрям гравітації.

5. Практичне застосування

Запропонований підхід цілком вирішує поставлену задачу. Дослідним шляхом визначено, що одна з основних відмінностей досліджуваної моделі полягає в інформованості моделі лише про перешкоди, а не вільний простір. Тож методи побудови графу, які використовуються в інших моделях тут не допоможуть.Тому визначено, що існує 3 варіанти побудови графа: повне трьох мірне переміщення;можливість переміщення по перешкодам з будь-якої сторони; переміщення за умов гравітації. В першому випадку нічого не блокує переміщення в будь-якому напрямку, в другому – можливо рухатися лише там, де є перешкодо, наприклад, можна рухатися по полу, стелі, стінах. В третьому випадку тільки по перешкодам, де нормаль їх поверхні має деякий максимальний кут до вектору гравітації. Була розроблена програмна реалізація, яка підпорядковується таким вимогам: кроссплатформенність, многопоточність, генерація моделі на льоту та поставлятися двома способами, як dll бібліотеки, та в відкритому коді. Практичним шляхом буда перевірена робота програми на

6. Висновки

Отже, розроблено модель навігації, яка, на відміну від інших, відповідає таким вимогам: при побудові надається інформація про перешкоди, які можуть бути виражені у виді простих геометричних об’єктів та полігональної сітки; модель зберігається для її повторного використання; при використанні, подаючи початкову точку шляху і кінцеву, маємо отримувати повний шлях; можна

використовувати як локальний пошук шляху, так і глобальний, або їх комбінацію.

Варто відмітити, що проведені дослідження по розробці моделі навігації, а також шляхів її реалізації у вигляді програми, вказують н можливість зміни такиї її принципів як розташування вузлів, обробка не вільного, а зайнятого простору, обробка пустих просторів не в режимі графів.

Удосконаливши таким чином модель, можна очікувати, що отримані результати будуть ліпшими за стандартну модель

Список літератури:

1. Geometric Modeling Using Octree Encoding, DONALD MEAGHER, Rensselaer Polytechnic Institute, Troy, New York 12181, June 19, 1981

2. P. Hubbard, “Interactive collision detection,” Proceedings of IEEE Symposium on Research Frontiers in Virtual Reality ’93, 1993

УДК 519.854.2

АНТИКОВ Д.В.
ХАЛУС О.А.

ЕФЕКТИВНІСТЬ ПОЛІНОМІАЛЬНОГО АЛГОРИТМУ ПОБУДОВИ ДОПУСТИМОГО РОЗКЛАДУ ДЛЯ ОДНОГО ПРИЛАДУ З ВІДОМИМИ ДОВІЛЬНИМИ ДИРЕКТИВНИМИ ТЕРМІНАМИ З МІНІМАЛЬНИМ СУМАРНИМ ВИПЕРЕДЖЕННЯМ В ЗАДАЧАХ ТЕОРІЇ РОЗКЛАДІВ У ВИПАДКУ НЕВЕЛИКОЇ КІЛЬКОСТІ РОБІТ НА ПРИЛАДІ

Досліджується розроблений у [1] алгоритм А2 впорядкування допустимого розкладу виконання робіт за критерієм мінімального випередження для задач одного приладу з відомими директивними термінами в теорії розкладів. У ході досліджень було розроблено програмну систему для перевірки достовірності роботи алгоритму. Порівнюються розклади складені алгоритмом та сформовані повним перебором. Для різних характеристик розкладу підраховано відсотковий фактор помилкового складання розкладу досліджуваним алгоритмом.

Ключові слова: КАЛЕНДАРНЕ ПЛАНУВАННЯ, РОЗКЛАД, ОДИН ПРИЛАД, МІНІМІЗАЦІЯ СУМАРНОГО ВИПЕРЕДЖЕННЯ, ПДС-АЛГОРИТМ.

In this article we study the algorithm from [1] that solves a single machine scheduling problem to minimize total earliness. Created programmed test system to check the validity of algorithm. We compare the algorithm output and the schedule found by exhaustive search. For different schedule characteristics we counted the failure factor of algorithm.

Keywords: SCHEDULING, SCHEDULE, ONE DEVICE, MINIMIZING THE TOTAL ADVANCE, THE PDS ALGORITHM.

Вступ

У статті розглядається задача побудови допустимого розкладу (у якому не перетинається жоден з директивних термінів) для одного приладу з різними директивними термінами виконання робіт з мінімальним сумарним випередженням. Розглядається наближений ефективний ПДС-алгоритм, запропонований в [1]. Для практичного застосування алгоритму, необхідно кількісно визначити ефективність алгоритму, за можливістю виділити окремі класи розкладів, для яких алгоритм не дає оптимального рішення, оцінити відносну похибку у випадку неточності вихідного розкладу алгоритму. Для цього формовано систему генерації допустимих розкладів з різним директивним вікном, оцінено результат роботи алгоритму на генерованих допустимих розкладах.

Постановка задачі

Задано множину незалежних завдань $J = \{1, 2, \dots, n\}$, кожне з яких складається з однієї операції. Для завдання $j \in J$ відома протяжність виконання l_j та директивний термін виконання d_j. Завдання поступають до системи одночасно. Переривання у виконанні завдань не допускаються. Процес виконання завдань є неперервним: після виконання по порядку першого завдання одразу ж починається виконання другого і т.д. до тих пір, поки не будуть виконані усі завдання. Необхідно знайти впорядкуванням такий розклад, що сумарне випередження директивних термінів буде мінімальним серед усіх допустимих розкладів.

Теоретичні відомості

Введемо позначення:

C_j – момент завершення виконання завдання j;

E_j – значення випередження для j-го завдання: $E_j = max\{0, d_j - C_j\}$;

$E(\delta)$ – значення сумарного випередження розкладу δ;

T_j – значення запізнення для j-го завдання: $T_j = max\{0, C_j - d_j\}$;

l_j– час виконання завдання, що стоїть у розкладі на позиції j;

r – момент початку виконання завдань.

У допустимому розкладі $d_j \geq C_j \ \forall j$, тобто $E_j = d_j - C_j$.

Генератор розкладів

Введемо позначення:

n - кількість робіт у розкладі, що буде згенеровано.

α - максимальне значення продовжності роботи, що може бути згенерована.

k - коефіцієнт директивного вікна - в згенерованому допустимому розкладі різниця між директивним терміном кожної роботи та відповідним часом закінчення не буде перевищувати αk.

S_i - кінець роботи перших i завдань.

Генерація допустимого розкладу ведеться на основі лінійного розподілу. Роботи для приладу генеруються так, щоб задовольнялася Теорема 1[2]. Алгоритм генерації виглядає наступним чином:

Алгоритм Г

Вважаємо $i = 0$, $S_i = 0$.

КРОК 1.

1.1. $i = i + 1$.

1.2. Згенерувати продовжність роботи l_i з лінійного проміжку $[1..\alpha]$.

1.3. $S_i = S_i + l_i$

1.4. Згенерувати директивний термін роботи d_i з лінійного проміжку $[S_i..S_i + \alpha k]$

1.5. **ЯКЩО** $i < n$,

ТО перейти до 1.1.

ІНАКШЕ КІНЕЦЬ – отримано, згідно Теореми 1 допустимий розклад.

Приклади некоректної роботи алгоритму А2

У [1] запропоновано алгоритм А2 для впорядкування розкладу за критерієм мінімального випередження, що дозволяє отримати впорядкування за алгоритмічний час $kO(n^2)$, де k - кількість змін структури розкладу. Його ідея базується на твердженні, що покращення розкладу за критерієм мінімального випередження в обраний момент початку виконання робіт не можна досягти, якщо кожна попарна зміна місцями веде до порушення допустимості або до погіршення значення критерію. Проте є приклади, що спростовують цю ідею.

Під час опису Алгоритму А1[1] мінімальне випередження для конкретного часу початку r визначається підстановкою на останнє невизначене місце мінімально-продовжної роботи, що не порушує допустимість розкладу. Проте це не завжди вірно.

Приклад 1. Нехай маємо розклад, як показано в Таблиці 1.$r_{max} = r = 0$ Аглгоритм А1 згенерує розклад, як показано в Таблиці 2. Проте оптимальним буде розклад, показаний у Таблиці 3.

Неможливість визначити алгоритмом А2 оптимального випередження для конкретного r в деяких випадках дозволяє поставити під сумнів визначення оптимального мінімального випередження для розкладу в цілому.

Перевірка ефективності алгоритму

Для перевірки за допомогою генератора створюємо розклад. Запускаємо на ньому Алгоритм А2 і паралельно для цього ж розкладу шукаємо оптимальне значення мінімального випередження повним перебором. Якщо Значення критеріїв не співпадають - алгоритму А2 зараховується помилка. У таблиці 4 статистично пораховано вірогідність помилки алгоритмом А2 для відповідних n та k при кількості згенерованих розкладів 10^5.У таблиці 5 показано отриманий за допомогою генератора розклад для $n = 6, k = 2$, $\alpha = 30$, на якому А2 видає хибний розклад, що показаний у таблиці 6. Оптимальний розклад, отриманий перебором, показаний у таблиці 7.

Висновки

Аналізуючи таблицю 4, можна сказати, що зі збільшенням кількості завдань у розкладі, вірогідність помилки алгоритму збільшується. Зі збільшенням директивного вікна, вірогідність помилки спочатку збільшується, досягає своїх максимумів на $(k = 1{,}5, k = 2{,}0)$,а далі поступово наближається до 0 на нескінченності. Зі збільшенням кількості робіт, екстремум по k досягає більших значень. Для більшості помилкових результатів алгоритму А2, різниця між оптимальним та знайденим значенням критерію не перевищувала 10%, проте зафіксовані одиничні випадки, коли алгоритмічне значення було вдвічі більше за оптимальне. Алгоритм дійсно наближено знаходить оптимальний за критерієм мінімального випередження розклад і може бути використаний для практичних задач. Найкращі результати видає на невеликих розкладах з невеликим директивним вікном.

Таблиця 1. Приклад розкладу, на якому A1 видає помилку

d_i	14	6	6	6	14
l_i	8	1	1	1	3

Таблиця 2. Результат роботи A1 на розкладі з таблиці 1

d_i	6	6	6	14	14
l_i	1	1	1	8	3
C_i	1	2	3	11	14

$E = 15$

Таблиця 3. Оптимальний за критерієм мінімального випередження розклад, отриманий на основі розкладу з таблиці 1

d_i	14	6	6	6	14
l_i	3	1	1	1	8
C_i	3	4	5	6	14

$E = 14$

Таблиця 4.Помилки A2 для різних значень k та n

k\n	4	5	6	7	8	9
0.25	0.013	0.032	0.051	0.039	0,074	0,091
0.5	0.096	0.212	0.360	0,430	0,581	0,741
0.75	0.405	0.668	0.841	1,297	1,721	2,314
1	0.659	1.368	2.049	2,781	3,561	4,329
1.25	1.069	1.709	2.820	3,911	5,417	6,863
1.5	1.014	2.338	3.385	5,055	6,170	8,419
2	1.039	2.505	3.886	6.031	8,226	10,026
2,5	1,017	2,435	3,921	6,584	9,067	11,131

Таблиця 5. Генерований розклад, на якому A2 видає помилку

d_i	32	59	63	65	72	102
l_i	2	6	7	9	29	1

Таблиця 6. Результат роботи A2 на розкладі з таблиці 5

$r = 19$						
d_i	65	32	63	59	75	102
l_i	9	2	7	6	29	1
C_i	28	30	37	43	72	73

$E = 110$

Таблиця 7. Оптимальний за критерієм мінімального випередження розклад складений на

$r = 12$						
d_i	32	72	63	59	65	102
l_i	2	29	7	6	9	1
C_i	14	43	50	56	65	66

$E = 99$

основі розкладу з таблиці 5.

Список літератури

1. Павлов А.А. Составление допустимого расписания выполнения работ на одном приборе с целью минимизации суммарного опережения работ / А.А. Павлов, Е.А. Халус // Вісник НТУУ “КПІ”. Серія «Інформатика, управління та обчислювальна техніка». – К.: “ВЕК+”, 2014. – №61. – с.27–34 Режим доступу: http://it-visnyk.kpi.ua/?page_id=2463
2. Павлов А. А. Исследование свойств задачи календарного планирования для одного прибора по критерию минимизации суммарного опережения заданий при условии допустимости расписании / А. А. Павлов, Е. Б. Мисюра, Е. А. Халус // Вісник Національного технічного університету України "Київський політехнічний інститут". Сер. : Інформатика, управління та обчислювальна техніка. - 2012. - Вип. 56. - С. 98-102. - Режим доступу: http://nbuv.gov.ua/UJRN/Vkpi_iuot_2012_56_15.

УДК 004.492.3

ОХРІМЧУК Є.В.
СЕЛІН Ю.М.

АНАЛІЗ ОСНОВНИХ ПІДХОДІВ І ТРЕНДІВ В ПРОЕКТУВАННІ ГРАФІЧНИХ ІНТЕРФЕЙСІВ ДЛЯ МОБІЛЬНИХ ДОДАТКІВ

В даній статті розглянуто основні директиви мобільного дизайну: поточна популярність, особливості реалізації та нові тренди. Описані ключові особливості для найпопулярніших платформ. Проведено аналіз основних факторів проектування. Виділено найтиповіші помилки. За результатами аналізу створено перелік універсальних правил, необхідних при проектуванні графічних інтерфейсів мобільних додатків на різних платформах.

In this article are considered the basic directions of mobile design: current popularity, implementation features and new trends. The key features of the most popular platforms are described. The analysis of the basic factors was done. Based on the analysis results, was created a universal rules list for designing of the mobile applications graphic interfaces on various platforms.

1. Вступ

Про дизайн мобільних додатків написано вже доволі багато інформації. Проте в цій сфері є величезна кількість нюансів. Деякі відомі не всім, інші швидко забуваються. У цій статті я постарався описати основні причини виникнення проблем, та методи роботи з ними, а також кілька прийомів проектування додатків, які можна назвати ефективними і перевіреними.

2. Основні директиви мобільного дизайну

За останні роки мобільний ринок змінився настільки сильно, що сьогодні навіть найпотужніший смартфон п'ятирічної давності виглядає безглуздо і смішно. Розвиток мобільних ОС рухався величезними кроками, і від пануючих колись операційних Symbian і Windows Mobile не залишилося нічого, крім спогадів. Само собою, розвиток смартфонів відбувався як з апаратної частини, так і з програмної. Особливості дизайну на мобільних пристроях, як і на будь-яких інших платформах, напряму залежать від конкретних видів операційний систем і базуються на їх вимогах. На даний момент основну долю ринку мобільних пристроїв займають телефони на базі Android (80,7%) та iOS (17,7%), а це як-не-як 98,4% від усіх, тому розглянуті були саме вони.

Інтерфейс користувача (user interface, UI) — засіб зручної взаємодії користувача з інформаційною системою. Сукупність засобів для обробки та відображення інформації, максимально пристосованих для зручності користувача.

При розробці мобільного інтерфейсу обов'язково використовуються гайдлайни. Guidelines - набір рекомендацій від творців платформи, завдяки яким програми сторонніх розробників виглядають однаково. На сьогоднішній день всі популярні платформи мають у відкритому доступі свої гайдлайни. Гайдлайн в загальному розумінні - це документ, який випускає компанія, і за яким дизайнери і розробники розуміють принцип побудови взаємодії додатка з користувачем. Умовно кажучи, для iOS кнопки треба робити круглими, а для Windows Phone - квадратними. Гайдлайни містять в собі інформацію по відступах, розмірах, візуальних ефектах, механіці анімації та ін.

3. Material Design

У 2014 році була представлена нова дизайн-система для ОС Android, яка отримала назву Material Design. Новий підхід дозволяє створювати сумісний користувацький досвід на всіх екранах: десктоп, смартфон, планшети, годинник, телевізори, машини. У багатьох сенсах це більш гнучка система, яка створювалася з урахуванням того, що користуватися нею може будь-хто.

Material Design дозволяє більш об'єктивно підходити до прийняття

дизайн-рішень: як щось виглядає, як щось працює, як здійснюється анімація і так далі. Вона задає розумні рамки, але не зайві обмеження.

Основні принципи:

- Тактильні поверхні

У Material Design інтерфейс складається з відчутних шарів так званого «цифрового паперу». Ці шари розташовані на різній висоті і відкидають тіні один на одного, що допомагає користувачам краще розуміти анатомію інтерфейсу і принцип взаємодії з ним.

- Поліграфічний дизайн

Якщо вважати шари шматками «цифровий паперу», то в тому, що стосується «цифрового чорнила» (всього того, що зображується на «цифровому папері»), використовується підхід з традиційного графічного дизайну: наприклад, журнального і плакатного.

- Осмислена анімація

У реальному світі предмети не виникають нізвідки і не зникають в нікуди - таке буває тільки в кіно. Тому в Material Design ми весь час думаємо про те, як за допомогою анімації в шарах і в «цифрових чорнилах» давати користувачам підказки про роботу інтерфейсу.

- Адаптивний дизайн

Йдеться про те, як застосовуються попередні три концепції на різних пристроях з різними дозволами і розмірами екранів.

4. Apple Human Interface Guidelines

HIG - документ, який містить рекомендації для розробників програмного забезпечення. Служить для створення найбільш інтуїтивних, легких для сприйняття і логічних інтерфейсів взаємодії з користувачами. Зазвичай уніфікує зовнішній вигляд додатків, а також вказує на необхідність використання технологій, що дозволяють робити додаток доступним на різних мовах і для людей з обмеженими фізичними можливостями.

Apple Human Interface Guidelines (AHIG) регулює буквально все, починаючи від назв і розташування пунктів меню та елементів керування у вікні (аж до пікселів) і закінчуючи розташуванням тимчасових файлів.

Apple позиціонує AHIG не тільки як керівництво до дій для розробників, але також як енциклопедію для користувача Mac OS. Адже більшість додатків відповідає Apple HIG, тобто ведуть себе однаково. Таким чином, освоївши одного разу основні принципи і сформувавши звичку, користувачу надалі буде легко працювати з будь-якими продуктами.

Основні принципи:

- Повага

Інтерфейс повинен служити прошарком між користувачем і програмною частиною, допомагати розуміти і взаємодіяти з контентом. Інтерфейс ніколи не повинен забирати на себе увагу від контенту, не повинен конфліктувати з ним.

- Чистота

Інтерфейс - прошарок між користувачем і програмною частиною, помічник по взаємодії з контентом. Інтерфейс не забирає уваги, не конфліктує з контентом. Прагніть до чистоти інтерфейсу.

- Глибина

Візуальна ієрархія шарів і реалістична поведінка просування користувача крізь ці шари повинні благотворно позначатися на розумінні користувачем поточного становища. Це принцип єдиного інтерфейсного простору користувача.

Особливості, напряму не відмічені в гайдлайнах:

- Відкладайте процедуру реєстрації

Відкладайте процедуру реєстрації як можна довше. Найкраще, коли користувачі можуть досліджувати ваш додаток практично порожниною і використовувати деякі функції без входу

- Контент завжди важливіший інтерфейсу

Не відводьте увагу людини від контенту, яким він цікавиться.

- Спілкуйтеся зі своїм користувачем так, немов ви редактор газети

Використовуйте неформальний і дружній тон

- Зробіть так, щоб вийти було неможливо

У додатку на iOS ніколи немає варіанту закрити його або вийти. Люди припиняють використовувати додаток, коли вони переключаються на інший додаток або повертаються на головний екран, або перемикають пристрій в сплячий режим

5. Сучасні тренди мобільних інтерфейсів в 2017 році

Тенденції в сфері дизайну мобільних додатків змінюються постійно. Якісь тренди зникають з часом, інші ж продовжують рости і розвиватися. Більшість сучасних UI-тенденцій гідно розгляду, адже за кожною новою «фішкою» цілком можуть ховатися інновації. Однак в дизайні для користувача інтерфейсу не так важливий «вау-ефект», як простота і ефективність. Саме тому дизайнери прагнуть до ідеальної функціональності - вони хочуть, щоб навігація додатки була настільки інтуїтивно зрозумілою, щоб користувачі взагалі не замислювалися, як вони взаємодіють з пристроєм.

- Невидимі меню

Невидимі або приховані меню не є якимось нововведенням. До цього давно все йшло, просто настав час, коли цей тренд зможе проявитися в повну силу. Дизайнери, нарешті, вже готові прийняти цей як даність і почати використовувати приховані меню - адже такий підхід відповідає очікуванням користувача.

- Розмиті і дифузійні фони

Сучасні мобільні пристрої можуть мати зовсім невеликий дисплей, однак розмір екрану не впливає на прагнення дизайнера будь-яким способом привернути увагу користувачів. Розмиті фони в дизайні користувальницьких інтерфейсів були дуже популярні в 2016 році і цей тренд поки ще залишається популярним. Пояснюється це просто - розмиття робить дизайн більш «живим», воно, при правильному підході привносить в інтерфейс обсяг і глибину. Крім того, розмитий або дифузне тло створює візуальний контраст, змінює перспективу і дозволяє зробити акцент на елементах першого плану.

- Card Design стане більш актуальним

Картки або плитки мали істотний вплив на веб-дизайн. Даний підхід дозволяє представити контент найбільш зручним способом для користувачів мобільних пристроїв. Не дивно, що плитка зацікавила дизайнерів додатків. Плитка є ефективним інструментом, перш за все тому, що вона дозволяє розбити контент на більш дрібні частини і потім зв'язати кожен елемент з відповідним змістом.

- Розваги і персоналізація

На дизайн інтерфейсів мобільних додатків впливає безліч факторів, у тому числі і культура. Останнім часом з'явилося безліч додатків, в які звичні або навіть консервативні елементи істотно змінилися. У дизайні з'явилися сміливі кольори, в інтерактивних елементах з'явився натяк на гру і діалоги стали більш дружніми. Все частіше зустрічаються діалогові вікна з кумедними повідомленнями або повідомленнями, в яких оптимістичний текст посилюється кумедними картинками.

- Неяскраві колірні палітри

У дизайні нерідко трапляється, що здавалося б взаємовиключні тренди залишаються однаково актуальними. Інтерфейс мобільного додатка повинен бути в першу чергу функціональним, а ось колірна палітра може змінюватися, в залежності від призначення програми. В якомусь випадку дизайнер вибере яскраву колірну палітру і буде правий, так як яскраві кольорові схеми сьогодні в тренді, але він також буде правим, коли злегка приглушить колір, якщо вважатиме це за необхідне.

6. Найтиповіші помилки в проектуванні мобільних інтерфейсів

- Нестача продуманої архітектури і навігації

Неможливо побудувати будинок без креслення, таке ж правило справедливе до UX-дизайну.

- Надлишок дизайну

Основна ціль графічного інтерфейсу - допомогти донести інформацію, не відволікаючи і не заплутуючи. Якщо мета - хороший інтерфейс, потрібно робити все по можливості простіше.

- Складні інтерфейси

Закон Мерфі для мобільних додатків можна перефразувати приблизно так:

«Все, що користувачі можуть неправильно зрозуміти, вони і зрозуміють неправильно».

- Пріоритети

Візуальна ієрархія - доволі ефективний концепт, що дозволяє зробити основні деталі помітніше, так щоб вторинні елементи не сильно кидалися в очі. Потрібно визначитись з чим користувачу доведеться взаємодіяти частіше і підкреслити ці компоненти. Навігація в такому інтерфейсі помітно спроститься.

7. Універсальні правила успішного проектування

- Цілеорієнтованість

Створювати дизайн потрібно для конкретного користувача. Зараз в мережі велика кількість даних про різні категорії користувачів, причому багато матеріалів - дослідження, огляди - доступні безкоштовно. Вивчення цієї інформації допоможе створити додаток, який відповідає потребам цільової аудиторії.

- Юзабіліті

Інтерфейс повинен бути зручним і інтуїтивно зрозумілим. Наприклад, якщо необхідно розмістити посилання для переходу на сторонній ресурс, то оформляти його потрібно звичним чином - за допомогою підкресленого блакитного тексту. Зручність і практичність - це перший крок на шляху до того, щоб програма стала бажаною для користувача.

- Здатність до навчання

В ідеалі користувач повинен без праці здогадуватися, як працювати з програмою. Тут приходять на допомогу знайомі і звичні схеми оформлення програми. Вони повинні допомогти людині без проблем звикнути.

- Фідбек і час відповіді.

Відгук додатку повинен давати користувачеві уявлення про те, виконана задача чи ні. Це може бути звичайний звуковий сигнал або щось складніше - наприклад, модальне вікно. Необхідно переконатися в тому, що фідбек додатку відповідає положенням, встановленим Nielsen Norman Group.

8. Висновки

Сучасні мобільні технології вже впродовж десяти років стабільно набирають оберти, будучи під тиском стрімкого інформаційного прогресу і високої ринкової конкуренції. Відповідно, за цей час відбулись безліч концептуальних змін та інновацій в сфері графічних інтерфейсів. І хоч технології змінюються так швидко, що дуже важко передбачити, як будуть виглядати інтерфейси мобільних додатків уже через рік, можна з впевненістю сказати, що незалежно від актуальних моделей смартфонів і операційних систем, гайдлайнів чи цільових призначень додатків, є доведені на практиці основи проектування. По-перше, додатки створюються для юзерів, а отже графічний інтерфейс має повністю враховувати їх смаки і потреби. По-друге, користувач завжди хоче бути зацікавлений чимось новим і нестандартним.

Список літератури

1. Фисун А. П., Гращенко Л. А. и др. Теоретические и практические основы человеко-компьютерного взаимодействия: базовые понятия человеко-компьютерных систем в информатике и информационной безопасности / А. П. Фисун. — Деп. в ВИНИТИ 15.10. 2004 г. № 1624 – В 2004. — Орел: Орловский государственный университет, 2004. — 169 с. — (Рукопись).
2. Kyle Mew, “Master Material Design and create beautiful, animated interfaces for mobile and web applications”, December 2015, Packt Publishing Ltd, 186 c.

УДК 519.854.2

КЕЛЛЕР Р.В.

ЗАДАЧА ОПТИМІЗАЦІЇ РОЗКЛАДУ ВИКОНАННЯ ЧАСТКОВО ВПОРЯДКОВАНИХ РОБІТ НА МАШИНАХ РІЗНОЇ ПРОДУКТИВНОСТІ ЗА НАЯВНОСТІ ПРОСТОЇВ

В даній статті розглянуто задачі теорії розкладів та наведено методи їх вирішення. Побудовано математичну модель для задачі оптимізації розкладу виконання частково впорядкованих робіт на машинах різної продуктивності за наявності простоїв та розроблений алгоритм отримання початкового допустимого розв'язку.

The subject of this article is the job shop scheduling problem and methods for solving this problem. It contains model for the task of schedule optimizing for partially ordered jobs on machines with different productivity in the presence of idle time and algorithm for getting initial permissible solution.

Вступ

Теорія розкладів займається “впорядкуванням” та “складанням розкладів”. Впорядкування – це формування черги операцій, які виконує одна машина, а складання розкладів – це завдання послідовності дій для декількох машин. У даній науковій роботі розглядається “складання розкладів”, а саме клас задач в яких ми маємо систему обслуговування, яка складається з машин. Кожна машина виконує операції (якісь елементарні задачі). Ці операції є частково впорядкованими, тобто виконання однієї операції може як залежати від виконання попередньої так і не залежати. При цьому, тривалість операції не є сталою величиною, вона залежить від машини на якій вона буде виконуватися. На осі часу можуть бути “пропуски” тобто вимушені періоди простою машини, коли вона не може здійснювати жодних операцій. Сукупність машин, операцій та дисциплін призначення операцій відповідним машинам називається процесом обслуговування.

Складання розкладу для цього процесу означає, що для кожної операції на часовій осі відповідає підмножина, на якій ця операція виконується відповідною машиною. Головним завданням такого класу задач є зменшення часу обслуговування.

Існуючі підходи до розв’язання задач планування розкладу

Метод гілок та меж

Метод гілок і меж (англ. Branch-and-Bound) — один з поширених методів дискретної оптимізації. Метод працює на дереві рішень та визначає принципи роботи конкретних алгоритмів пошуку розв’язків, тобто, є мета-алгоритмом. Для різних задач комбінаторної оптимізації створюють спеціалізовані алгоритми гілок та меж.

Результатом роботи алгоритму є знаходження максимуму функції на допустимій множині. При чому множина може бути як дискретною, так і раціональною. В ході роботи алгоритму виконується дві операції: розбиття вихідної множини на підмножини (гілки), та знаходження оцінок (меж). Існує оцінка множини згори та оцінка знизу. Оцінка згори — точка що гарантовано не менша за максимум на заданій підмножині. Оцінка знизу — точка що гарантовано не більша за максимум на заданій підмножині. Множина що має найбільшу оцінку зверху зветься рекордною. На початку вся множина вважається рекордною.

Генетичні алгоритми

Генетичні Алгоритми (далі ГА) - адаптивні методи пошуку, які останнім часом часто використовуються для вирішення задач функціональної оптимізації. Вони засновані на генетичних процесах біологічних організмів: біологічні популяції розвиваються протягом декількох поколінь, підкоряючись законам природного відбору і за принципом "виживає найбільш пристосований" (survival of the fittest), відкритого Чарльзом Дарвіном. Наслідуючи цьому процесу генетичні алгоритми здатні "розвивати" вирішення реальних завдань, якщо ті відповідним чином закодовані. Наприклад, ГА можуть використовуватися, щоб проектувати структури моста, для пошуку максимального відношення міцності / ваги, або визначати найменш марнотратне розміщення для нарізки форм з тканини. Вони можуть також використовуватися для інтерактивного управління процесом, наприклад на хімічному заводі, або балансуванні завантаження на багатопроцесорному комп'ютері. Цілком реальний приклад: ізраїльська компанія Schema розробила програмний продукт Channeling для оптимізації роботи стільникового зв'язку шляхом вибору оптимальної частоти, на якій буде вестися розмова. В основі цього програмного продукту і використовуються генетичні алгоритми. ГА також використовуються для задач складання розкладу.

Табу-пошук

Табу-пошук (ТП) — метод локального пошуку для математичної оптимізації. Створений Фредом У. Гловером в 1986 році і формалізований в 1989.

Табу-пошук (ТП) є мета-евристичним алгоритмом, який веде локальний пошук, щоб запобігти його від попадання в пастку

в передчасних локальних оптимумах, забороняючи ті переміщення, які змушують повертатися до попередніх рішень і циклічної роботи. ТП починається з вихідного рішення. На кожній ітерації генерується околиця рішень, і найкраще з цієї околиці вибирається як нове рішення. Певні атрибути попередніх рішень зберігаються в табу-списку, який оновлюється в кінці кожної ітерації. Вибір найкращого рішення в околиці відбувається таким чином, що він не приймає жодного з заборонених атрибутів. Найкраще допустиме рішення в даний час, оновлюється, якщо нове поточне рішення краще і допустиме. Процедура триває, поки не виконається будь-який з двох критеріїв зупину, якими є максимальне число виконуваних ітерацій і максимальне число ітерацій, під час яких чинне рішення не поліпшується.

Алгоритм імітації відпалу

Алгоритм імітації відпалу (англ. *Simulated annealing*) - загальний алгоритмічний метод розв'язання задачі глобальної оптимізації, особливо дискретної та комбінаторної оптимізації, в якому процедура пошуку глобального розв'язку імітує фізичний процес відпалу.

Алгоритм Metropolis може бути використаний для генерування послідовності рішень задачі комбінаторної оптимізації, припускаючи еквівалентності між фізичною системою багатьох частинок і завданням комбінаторної оптимізації:

- розв'язання задачі комбінаторної оптимізації еквівалентні стану фізичної системи
- вартість рішення еквівалентно енергії стану системи.

Крім того, ми вводимо керуючий параметр, який відіграє роль температури. Таким чином Simulated annealing алгоритм, можна розглядати як ітерації алгоритму Metropolis, виконані на зменшення значення керуючого параметра. Значення чотирьох функцій в процедурі SIMULATED_ANNEALING очевидні:

- INITIALIZE обчислює і задає початкові значення параметрам c та L
- GENERATE вибирає сусідній розв'язок відносно поточного розв'язку
- CALCULATE.LENGTH і CALCULATE_CONTROL обчислює нові значення параметрів L та c відповідно

Порівнюючи simulated annealing з ітераційним поліпшенням, очевидно що simulated annealing можна розглядати як узагальнення. Алгоритм імітації відпалу стає ідентичним ітераційному покращенню в разі, коли значення керуючого параметра приймається як нуль. Що стосується порівняння продуктивність обох алгоритмів, для більшості завдань simulated annealing працює швидше, ніж ітераційне покращення.

Доцільність проведення дослідження

Цікавим прикладом є цехові завдання теорії розкладів з маршрутизацією, які узагальнюють цехові завдання з широко відомою метричною задачею комівояжера. Дивно, що постановки таких проблем незалежно з'явилися при розгляді задач, що виникають як на виробництві, так і в індустрії обслуговування. Більшість завдань теорії розкладів є NP-складними.

Експонентні алгоритми переборного типу вимагають значних обчислювальних витрат навіть при вирішенні прикладів середньої розмірності. Тому одним з важливих напрямків досліджень є пришвидшення алгоритмів для NP-складних задач. В даний час цей напрям завоювало величезну кількість прихильників в середовищі дослідників, що займаються комп'ютерною математикою. Перелічимо основні причини такої популярності:

- існує величезна кількість оптимізаційних задач, які вимагають рішення, і більшість з них NP-складні. Для багатьох з них необов'язково знаходити точні рішення, а досить побудувати приблизне, проте за короткий час, з чим успішно справляються наближені алгоритми;
- На практиці більшість оптимізаційних задач дуже складні,

тому що включають в себе багато додаткових обмежень, які не дозволяють знайти задовільне наближене рішення. Але зазвичай наближені алгоритми для більш простих варіантів тих же завдань підказують нові ідеї для евристик, які потім успішно застосовуються і для практичних завдань;

- З точки зору знаходження точних рішень майже все NP-складні завдання еквівалентні один одному. Побудова наближених алгоритмів або доказана неможливості їх побудови при розбіжності класів P і NP дає можливість порівняти між собою NP-складні завдання по тому, наскільки вони можуть бути апроксимовані.

Таким чином, ми бачимо, що сучасні алгоритми можуть бути покращені, за рахунок їх пришвидшення

Постановка задачі

Досліджується задача призначення частково впорядкованих робіт для обробки їх на машинах різної продуктивності, що містять визначені періоди простою за наявності простоїв з метою мінімізації моменту виконання всіх робіт.

Нехай маємо множину робіт $J = \{J_j\}, i = \overline{1,n}$ і множину машин $M = \{M_j\}, j = \overline{1,m}$. Час виконання кожної роботи на кожній із машин визначається матрицею $(x_{ij})_{n\times m}$. Також рахується заданою множина періодів простою $H = \{H_l\}, l = \overline{1,k}$, разом з трьома векторами $(\beta_1, \dots, \beta_k)$, $(r_1, \dots, r_k)$, $(\tau_1, \dots, \tau_k)$, які визначають тривалість періоду простою, номер машини, до якої він належить, і час його початку відповідно.

Рішення задачі можна представити двома векторами — $(s_1, \dots, s_n)$, $(z_1, \dots, z_n)$, що задає номер машини на якій заплановано роботу і час початку її виконання.

Виходячи із зазначених вище позначень, формальна модель задачі може бути представлена наступним чином:

$$x_{ij} > 0, i = \overline{1,n}, j = \overline{1,m},$$
$$\beta_l > 0, l = \overline{1,k},$$
$$s_i \in \{1, \dots, m\}, i = \overline{1,n},$$
$$r_l \in \{1, \dots, m\}, l = \overline{1,k},$$
$$z_i \geq 0, i = \overline{1,n},$$
$$\tau_l > 0, l = \overline{1,k},$$

Для всіх $i \in \{1, \dots, n\}, \nu \in \{1, \dots, n\}$ таких що $s_i = s_\nu,\ z_i < z_\nu$:

$$z_i + x_{is_i} \leq z_\nu, \quad (1)$$

Для всіх $i \in \{1, \dots, n\}, \nu \in \{1, \dots, k\}$ таких що $s_i = r_\nu,\ z_i < \tau_\nu$:

$$z_i + x_{is_i} \leq \tau_\nu, \quad (2)$$

Для всіх $i \in \{1, \dots, n\}, \nu \in \{1, \dots, k\}$ таких що $s_i = r_\nu,\ \tau_\nu < z_i$:

$$\tau_\nu + \beta_\nu \leq z_i, \quad (3)$$
$$\max_{i=1,n}(z_i + x_{is_i}) \to min. \quad (4)$$

Обмеження (1) задає не перетинання запланованих робіт на одній машині, обмеження (2) і (3) — не перетинання робіт і періодів простою. Обмеження (4) — цільова функція, яка забезпечує мінімізацію часу завершення найбільш пізньої роботи (т. н. гільйотинний розкрій).

Алгоритм отримання початкового допустимого рішення

Крок 1. ОБРАТИ машину з найбільшою потужністю.

Крок 2. ПЕРЕЙТИ на **Крок 4**.

Крок 3. ОБРАТИ машину з меншою потужністю.

Крок 4. ОБРАТИ задачу з найбільшою тривалістю меншою чи рівною інтервалу до зупинки машини.

Крок 5. ВИЛУЧИТИ обрану задачу зі списку доступних.

Крок 6. ВСТАНОВИТИ обрану задачу в обробку машиною.

Крок 7. ЯКЩО поточна машина має не найменшу потужність ТО ПЕРЕЙТИ на **Крок 3**.

Крок 8. ОБРАТИ першу машину

Крок 9. ПЕРЕЙТИ на **Крок 11**.

Крок 10. ОБРАТИ наступну машину

Крок 11. ОБРАТИ задачу з найбільшою тривалістю меншою чи рівною інтервалу до зупинки машини.

Крок 12. ЯКЩО задача не обрана ТО ОБРАТИ наступний інтервал після зупинки машини ІНАКШЕ ПЕРЕЙТИ на **Крок 14**.

Крок 13. ПЕРЕЙТИ на **Крок 11**.

Крок 14. ВСТАНОВИТИ час кінця роботи машини рівним часу кінця обробки задачі.

Крок 15. ЯКЩО поточна машина не остання ТО ПЕРЕЙТИ на **Крок 10**.

Крок 16. ОБРАТИ машину з найменшим часом кінця роботи.

Крок 17. ВИЛУЧИТИ обрану задачу для даної машини зі списку доступних.

Крок 18. ВСТАНОВИТИ обрану задачу в обробку даною машиною.

Крок 19. ЯКЩО список доступних задач не пустий ТО ПЕРЕЙТИ на **Крок 8**.

Висновок

Як показує наведений вище огляд, всі відомі методи поставлену задачу планування розкладу будуть виконувати з NP-складністю. Враховуючи сучасні тенденції розвитку обчислювальної техніки, варто зосередити увагу саме на зменшенні складності існуючих алгоритмів, що призведе до пришвидшення їх роботи і підвищення продуктивності програмного забезпечення, що буде їх використовувати.

Були розглянуті відомі методи роз’язання згаданої вище задачі та запропонований алгоритм отримання початкового допустимого рішення.

Список літератури

1 Гуляницкий Л.Ф., Туринський В.В. Математическая модель одного класса задач планирования работы независимых машин // Компьютерная математика. Киев: Институт кибернетики им. В.М. Глушкова НАН Украины, 2012. – №2. – С. 1-6.

2 Stavrov G.Kolliopoulos, George Steiner. Approximation algorithms for minimitation the total weighted tardiness on single machine – 2006 – p.1-10

3 А. А. Лазарев, Е. Р. Рашидович Теория расписаний [Текст]/ Москва, 2011 – 222 с.

4 О. А. Щербина Метаэвристические алгоритмы для задач комбинаторной оптимизации (обзор) [Текст]/ Севастополь, 2014 – 17 с.

УДК 004.93(015.7)

ЄРШОВ П. С.
ГРИША О. В.

ЗБІЛЬШЕННЯ ЕФЕКТИВНОСТІ ПРОЦЕСУ ДІЯЛЬНОСТІ НА ОСНОВІ PROCESS MINING

Розглянуто задачу отримання актуальної моделі процесу діяльності у вигляді мережі Петрі, її аналізу та побудови ефективного розподілу ресурсів для реалізації цього процесу .

Сучасні інформаційні системи підтримки процесів діяльності в більшості своїй забезпечені журналами реєстрації подій. В журналах інформація автоматично зберігається і може бути використана для отримання актуальних знань про процеси діяльності. Пропонується використати Process Mining для виявлення процесів а також перевірки відповідності, виявлення відхилень, та надання рекомендацій що до підвищення ефективності процесів.

Виявлено основні проблеми в області Process Mining.

Розглянуто задачу побудови актуальної моделі процесу діяльності, ідентифікації її параметрів та надання рекомендації що до покращення параметрів процесу. Апаратом дослідження обрано модель у формі мережі Петрі. Запропоновано два актуальних критерії покращення процесу діяльності: зменшення вартості виконання процесу та зменшення залучення більш кваліфікованих ресурсів (при обмеженні на вартість).

В результаті створюється актуальна модель для діагностики процесів діяльності складної організаційної системи та збільшення ефективності розподілу людських ресурсів для різних вхідних потоків.

Is offered a task of founding a process activities model as a Petri nets from the event log, its analysis and building an effective resources sharing for the implementation of this process.

Modern information systems supporting the business processes are mostly provided with the event log. Logs information is automatically saved and can be used to obtain relevant knowledge about the processes activity. It is proposed to use the Process Mining for mining to-date information about workflow and verifying and recommending for improve the efficiency of processes.

The paper discovers basic problems in the field of Process Mining.

It is analyzed the task of constructing an actual model of the process activity, identification of its parameters and providing recommendations to improve the process parameters. We have used the Petri nets modeling as device for research. Two effective criterion of process parameters improvement are proposed: reducing the cost of implementation of the process and reducing the attraction of more qualified resources (until don't go beyond the cost limit).

As a result is created the model for the diagnosis of complex business processes of organizational system and improvement the efficiency of allocation of human resources for the different input streams.

1. Вступ

Дослідження процесів діяльності компаній для їх аналізу та оптимізації є надзвичайно важливою, проте досить трудомісткою процедурою, яка вимагає узгодженої діяльності, як системних аналітиків, так і фахівців предметної області.

З іншого боку сучасні інформаційні системи підтримки процесів діяльності в більшості своїй забезпечені журналами реєстрації подій. В журналах інформація автоматично зберігається і може бути використана для отримання актуальних знань про процеси діяльності. Наприклад, ERP-системи, такі як SAP реєструють всі транзакції. Бізнес для бізнесу (B2B) системи реєструють обмін повідомленнями з іншими сторонами, системи CRM загального призначення реєструють взаємодії з клієнтами [3]. У журналі подій зазвичай міститься інформація про події з посиланням на вид діяльності і випадок як екземпляр процесу. Розрізняють три ракурси подання процесу в журналі: Ракурс процесу «Як?», Організаційний ракурс «Хто?» і Ракурс примірника процесу «Що?» з фіксацією часу операції.

Процес видобування знань про процеси діяльності з журналів подій називається Process Mining. Він спрямований на використання даних журналів для побудови актуальних моделей процесів діяльності, які в подальшому можна застосувати для діагностики та поліпшення бізнес-процесів компаній. Ця інформація є доступною у режимі реального часу, що наразі дуже важливо для підтримки конкурентоздатності у сучасному турбулентному оточенні бізнесу. Process Mining, таким чином, лежить між інтелектуальним аналізом даних та моделюванням і аналізом процесів.

Process Mining не обмежується виявленням процесів [1]. Зіставляючи дані про події та моделі процесів, можливо перевіряти відповідність, виявляти відхилення, здійснювати підтримку прийняття рішень і давати рекомендації. Таким чином, можна виділити три ключові підходи, які застосовує Process Mining:

- Видобування (Discovery). Передбачає побудову моделі на підставі записів про те, що відбувалося фактично.
- Перевірка відповідності (Conformance checking). Допомагає знайти відповіді на питання: де реальний процес відхиляється від очікуваного? Чому відбуваються подібні відхилення?
- Удосконалення (Enhancement). Відповідає на запитання: що потрібно поміняти в моделі, щоб покращити певні показники.

2. Постановка задачі

Розглянемо задачу побудови актуальної моделі процесу діяльності, ідентифікації її параметрів та надання рекомендації що до покращення параметрів виконання процесу .

Основні проблеми, що існують досі в області Process Mining-у викладені ван дер Aalst (2004a), як:

1. Шум. Зареєстровані дані можуть бути неправильні або неповні через проблеми зі створенням або виникають в момент видобування.
2. Приховані завдання. Завдання, які існують, але не можуть бути знайдені в даних.
3. Повторювані завдання. Два вузла процесу можуть стосуватися однієї і тієї ж моделі процесу.
4. Невільний вибір. Керований вибір, який залежить від вибору, зробленого в іншій частині моделі процесу.
5. Видобуток петель. Процес може бути виконаний в кілька разів, петлі можуть бути простими за участю одного або декількох подій або більш складним.
6. Різні перспективи. Події процесу можуть бути додані з додатковою інформацією для процесу виявлення.
7. Дельта Аналіз. Порівняння моделі процесу і еталонної моделі для перевірки схожість / невідповідність.
8. Візуалізація результатів. Результати виявлення процесу можуть бути представлені в графічній формі в термінах панелі менеджменту.

9. Гетерогенні результати. Доступ до інформаційних систем, заснованим на різних платформах.
10. Одночасні процеси. Видобуток процесів, що відбуваються в той же самий час.
11. Локальний / глобальний пошук. Локальні стратегії обмежують простір пошуку і менш складні, глобальні стратегії є складними, але мають більше шансів знайти оптимальне рішення.
12. Процес повторного виявлення. Вибір алгоритму виявлення, який може повторно виявити клас моделей процесу з повного журналу процесу.

Не дивлячись на те, що багато які з завдань процесу виявлення можуть бути вирішені за допомогою комбінації модифікованих підходів інтелектуального аналізу даних і призначених для користувача вбудованих алгоритмів, не існує єдиного підходу, який може вирішити всі проблеми, з якими стикаються в процесі виявлення. Багато існуючих алгоритмів інтелектуального аналізу користувальницьких процесів, як правило, вирішують тільки одну або дві проблеми. Найбільш широко до задач Process Mining застосовуються генетичні алгоритми. Такі підходи показали хороші результати в області зниження шуму і виявлення прихованих операцій. Спроб використовувати технології нейронних мереж для подібних завдань виявлення було менше, можливо, через їх складність.

На ринку представлений ряд комерційних інструментів для глибинного аналізу процесів. У їх числі Futura Reflect, Disco, ARIS Process Performance Manager, Interstage Business Process Manager, QPR ProcessAnalysizer.

Оберемо для дослідження найбільш відомий фреймворк, спеціально розроблений для Process Mining-у – ProM. Він має відкритий вихідний код і в якості вхідного формату використовує MXML. Отримуємо актуальну модель процесів в термінах мережі Петрі, а також модель організаційного виконання процесів. Мережі Петрі дозволяють із заданою точністю представляти розгалужені, паралельні, циклічні процеси, володіють засобами аналізу, а також моделювання в реальному масштабі часу.

У якості приклада предметної області для дослідження обрано систему виконання проектів з виготовлення програмного забезпечення, що виконується спеціалістами, що мають нечітку спеціалізацію та кваліфікаційне розмежування (senior, middle, junior) і ці ресурси є частково взаємозамінні.

Задачею є динамічна побудова розкладу призначень, який модна будувати за одним з наступних критеріїв: мінімальна вартість виконання завдінь чи мінімальне навантаження на більш дорогі ресурси при дотриманні обмеження по прогнозованому поточному стану часової затримки на виконання наявної черги до операцій.

3. Методи розв'язку задачі

Для дослідження та оптимізації побудуємо часову функціональну мережу Перрі наступного вигляду[2] по даним Process Mining.

$R =< P, T, G, Q, \mu_0 >$,

Де $P = \{p_i\}$ - непуста множина позицій $i = \overline{1,n}$;

$P` = \{p`_i\}, P` \subset P$ підмножина позицій, що позначають готовність завдань для операції $i = \overline{1,i`}$;

$P`` = \{p``_i\}, P`` \subset P$ - підмножина позицій, що позначають наявні для виконання операцій ресурси $i = \overline{1,i``}$. $P``$ I $P` = \otimes$.

$P``` = \{p```_z\}, P``` \subset P``$ - множина наявних ресурсів по категоріях $z = \overline{1,z``}$, де $z``$ - загальна кількість категорій. $P`` = UP```, I\ P``` = \otimes$;

$T = \{t_j\}$ - непуста множина переходів $j = \overline{1,m}$, де m - кількість операцій процесу діяльності. Операції можуть бути

згрупованими у блоки за вимогою до залучення ресурсів.

$G = \{G_{p-t}, G_{t-p}\}$ - множина дуг мережі, $G_{p-t} = (p \times t), G_{t-p} = (t \times p)$ - відповідно від позицій до переходів та від переходів до позицій; $G_{p-t} \; I \; G_{t-p} = \varnothing$;

$Q = \{q_g\}$ - ваги дуг $q_g = \overline{1,|G|}$.

Дуги, типу G_{p-t} , що йдуть від позицій типу $P`$ мають вагу 1, що відповідає готовності заявки на операцію.

Дуги, типу G_{p-t} , що йдуть від позицій типу $P``$ до переходів позначають вимогу до ресурсів (людино-дні) і мають нормативну вагу $q`_{gj}$, яку визначають як $q`_{gj} = q^{min}_{gj}$ мінімально необхідне залучення $p```_i$-го типу ресурсу на операцію j. При виконанні процесу на моделі обирається конкретне залучення необхідного ресурсу $q_{gj} \geq q^{min}_{gj}$. Вважатимемо, що планування виконується до десятих частин ресурсу, що дає можливість залучити працівника на часткову участь. Залучення більше за 1 визначає необхідність залучити більше ніж одного працівника. У цьому випадку залучається необхідна кількість працівників повністю і додатково працівник частково. Кожному з ресурсів ставиться у відповідність вартість r_g в залежності від приналежності до категорії $p```_z$. При залученні на виконання операції n працівників до терміну виконання операції враховуємо затримку на комунікації між ними (Ричард Хекман) $\lambda * n(n-1)\Gamma\ 2$, де λ - середній коефіцієнт витрат часу на одну комунікацію у відсотках до загального часу .

Для переходів додатково завдано значення терміну спрацьовування Vt_{gj} для кожної з можливих категорій ресурсів при їх повному виключному застосуванні. При застосуванні декількох одиниць ресурсів та з врахуванням різної категорії термін спрацьовування переходу буде $Vt_j = \max_{g \in P```_j}(q_{gj} Vt_{gj}) + \max_{g \in P```_j}(q_{gj} Vt_{gj}) * \lambda * n_j (n_j - 1)\Gamma\ 2$, де $n_j = |P```_j|$ – потужність підмножини ресурсів, залучених для виконання j-ї операції;

μ_0 - початкове маркування мережі. Для позицій $P`$ початкове маркування відповідає наявності завдання на початку процесу. Маркування початкової позиції процесу діяльності збільшується за рахунок динаміки потоку заявок ззовні і зменшується за рахунок їхнього поступового відпрацювання. Для позицій $P``$ початкове маркування відповідає наявним резервам ресурсів по категоріям.

Розглянемо типовий вузол моделі процесу діяльності, поданий на рис. 1. При кожному такті роботи моделі такий вузол вирішує задачу призначення на одну з технологічно наступних за поточною операцією.

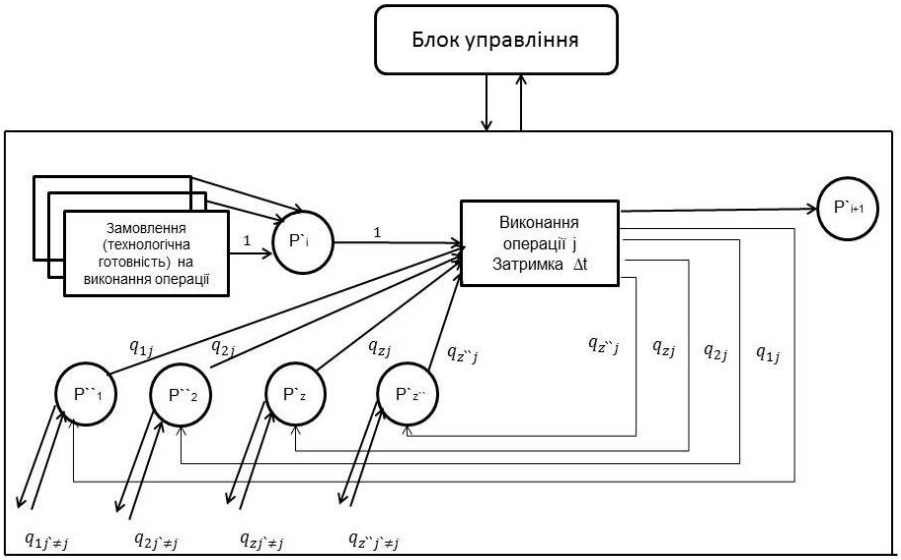


Рис. 1. Елемент моделі процесу діяльності.

Для досягнення необхідного значення показника якості роботи системи пропонується крім моделі об'єкта управління створити модель системи управління. Модель системи управління включає в себе блок перевірки умов спрацьовування переходів і блок визначення значень параметрів проходження переходів. за інформацією про поточний стан елементів моделі об'єкта управління. Для цього в моделі системи управління передбачається зворотний зв'язок. Система управління отримує дані про кількість спрацьовувань кожного переходу ($s[j]$), про сумарний час активності кожного переходу ($t[j]$) і розподіл ресурсів для кожного переходу ($Q[j]$) .

Блок управління діє на кожному такті k роботи моделі і виконує оптимальне призначення $Q_k = F(s[t], t[j], Q[j])$ за критерієм мінімальної вартості виконання процесу діяльності

$$\sum_{j=1}^{m} s_j \Delta t_j \sum_{g=1}^{n_j} q_{gj} r_{gj} \to \min$$

При розрахунку призначення для кожного такту k роботи моделі враховується обмеження

$\max_{p_i` \in P`}(\mu_{p_i`k} * \sum_{j=1}^{m} s_j t_j) \le \Delta t^{max}$ , де - $\Delta t^{max}$ - максимально допустима черга на операцію в часовому вимірі, $\mu_{p_i`k}$ - маркування позиції $p_i`$ на такті k роботи моделі.

Також актуальним при наявній легкості залучення більш дешевого ресурсу і складності отримання високо-кваліфікованого ресурсу можна скористатися критерієм

$$\sum_{j=1}^{m} s_j \Delta t_j \sum_{g=1}^{n_j} q_{gj} r_{gj} \to \min V t_{z_1}(\min V t_{z_2}(...(\min V t_{z``})...))$$

при врахуванні обмеження на вартість виконання процесу.

5. Висновки

В результаті створюється актуальна модель для діагностики процесів діяльності складної організаційної системи та збільшення ефективності розподілу людськ

их ресурсів для різних вхідних потоків.

Список літератури

1. Van der Aalst, W. M. P. 2011. Process Mining: Discovery, Conformance and Enhancement of Business Processes, (1st Edition) Berlin; Heidelberg: Springer.

2. Котов, В. Е. Сети Петри / В. Е. Котов. – М.: Наука, 1984. – 160 с.

УДК 004.93'12

ЗАНЧУК О. Й.
БАКЛАН І.В.

ОПТИЧНЕ РОЗПІЗНАВАННЯ МАТЕМАТИЧНИХ ВИРАЗІВ

В даній статті розглянуто питання та проблеми, що виникають при оцифровці зображень математичних виразів. Демонструється спосіб визначення границь розбиття зображень на атомарні одиниці для подальшого розпізнавання та формування ієрархічного графа, що представляє структуру оброблюваного зображення.

This article deals with issues and problems that arise during optical recognition of mathematical expressions. Demonstrated method of determining the boundaries splitting images to atomic unit for further recognition and formation of hierarchical graph representing the structure of the processed image.

1. Вступ

Ця стаття відноситься до автоматичної обробки зображень відсканованого документа і інших зображень, що містять текст, а зокрема до способів і систем перетворення зображень і фрагментів зображень документів, що містять математичні вирази, в електронні документи. Розглядається зокрема практичне застосування методу рекурсивно-блочного і заснованого на графі підходу до розпізнавання математичних виразів під час OCR-обробки зображення документа.

У пропонованому способі відбувається ітераційний поділ зображення математичного виразу документа на складові вирази і подальше розпізнавання цих складових. Відмінною ознакою є використання рекурсивно-блочного і заснованого на графі підходу до розпізнавання математичних виразів під час OCR-обробки зображення документа, що дозволяє вибрати найбільш оптимальний вид компоновки результатів розпізнавання в математичний вираз.

2. Сучасні рішення OCR

Друковані, машинописні і рукописні документи протягом довгого часу використовуються для запису і зберігання інформації. Незважаючи на сучасні тенденції відмови від паперового діловодства, друковані документи продовжують широко використовуватися в комерційних організаціях, установах і в домашніх умовах. З розвитком сучасних комп'ютерних систем формування, зберігання, пошук і передача електронних документів перетворилися в надзвичайно ефективний і економічно рентабельний альтернативний носій запису інформації і зберігання інформації. Внаслідок переваг по відношенню до ефективності та економічної рентабельності, які забезпечуються сучасними засобами зберігання і передачі електронних документів, друковані документи часто перетворюють в електронні документи за допомогою різноманіття способів і систем, включаючи конвертацію друкованих документів в цифрові зображення відсканованих документів з використанням електронних оптико-механічних скануючих пристроїв, цифрових камер, а також інших пристроїв і систем, і подальшу автоматичну обробку та зображень відсканованих документів для отримання електронних документів, перетворених в цифрову форму відповідно до одного або декількох стандартів кодування електронних документів.

Хоча сучасні OCR-програми еволюціонували до такої міри, що дозволяють автоматично перетворювати в електронні документи зображення складних документів, які включають картинки, рамки, лінії та інші нетекстові елементи, а також текстові символи будь-якої з безлічі поширених алфавітних мов, залишаються невирішеними проблеми щодо перетворення

зображень документа, що містять математичні вирази.

У патенті США 7181068 розкрито систему розпізнавання математичних виразів, спосіб розпізнавання математичних зображень, система розпізнавання символів і методу розпізнавання символів. Пристрій розпізнавання математичних виразів включає модуль, який розпізнає символи на зображенні документа, словник, який зберігає пару оціночних балів для кожного типу слова, бал, що відображає ймовірність приналежності до тексту і бал, що відображає ймовірність його приналежність до математичного виразу, оцінний модуль, який отримує оціночні бали , що відображають ймовірність приналежності до тексту і бал, що відображає ймовірність його приналежності до математичного виразу для кожного зі слів, включених в розпізнані символи з посиланням на словник, і модуль виявлення математичного виразу, який шукає оптимальний шлях, що з'єднує слова шляхом вибору одного з тексту і математичного виразу на основі формативної граматики і оціночних балів, що відображає ймовірність приналежності до тексту і його приналежності математичному виразу, тим самим детектуючи символи, що належать математичного виразу. Елементи математичного виразу перевіряються на факт того, чи є вони символами на базовій лінії, надрядковими символами чи підрядковими символами. Діаграма розсіювання розмірів символів, яка надає дані, відображає розмір нормалізації послідовних символів і розподіл їх можливих центральних позицій.

У пропонованому способі відбувається ітераційний поділ зображення математичного виразу документа на складові виразу і подальше розпізнавання цих складових. Відмінною ознакою є використання рекурсивно-блочного і заснованого на графі підходу до розпізнавання математичних виразів під час OCR-обробки зображення документа, що дозволяє вибрати найбільш оптимальний вид компоновки результатів розпізнавання в математичний вираз.

3. Проблеми, пов’язані з обробкою зображень, що містять математичні вирази

На Рис. 1-2 представлений ряд прикладів математичних виразів, а також вказівок елементів в математичних виразах, які є складними і являють труднощі для використовуваних в даний час OCR-методів, що застосовуються до фрагментів зображень документа, що містить математичні вирази. На Рис. 1 представлені два математичні рівняння 102-104, обрані з підручників з математики. Читач, знайомий з курсами математики і фізики на рівні перших курсів ВНЗ, легко інтерпретує всі рівняння, представлені на Рис. 1. Однак з точки зору автоматичних OCR-методологій розпізнавання цих математичних виразів на зображеннях і фрагментах зображень документів представляє безліч проблем.

$$V = \frac{1}{4a}\iint_R r^2\,dA = \frac{1}{4a}\int_0^{\pi} d\theta \int_0^{2a\sin\theta} r^3\,dr \qquad 102$$

$$f_k(z) = (-1)^{k-1} \prod_{i \le k} \langle i \rangle g(z)^{-\chi(k)}\, \Delta(z)^{-m(k)}\, E_{q^k-1}(z) \qquad 104$$

Рис. 1

На Рис. 2 проілюстровані деякі з проблем, що виникають при застосуванні автоматичних OCR-способів до математичних виразів, представленим на Рис. 1. В якості одного прикладу, досить часто в математичних виразах заданий тип символу може мати безліч накреслень, або зразків, в різних виразах. Наприклад, розглянемо символ інтеграла 212 і символ подвійного інтеграла 214 у виразі102. Символ інтеграла може зустрічатися один або кілька разів в залежності від розмірності інтегрування, яке представляє символ або символи інтеграла. З символом інтеграла можуть бути пов'язані нижня і верхня межі інтегрування, як у випадку символу інтеграла 212 в рівнянні 102. Однак положення нижньої і верхньої меж щодо

символу інтеграла ∫ в різних виразах можуть відрізнятися. У деяких випадках межі можуть бути під і над символом інтеграла, а в інших випадках вони можуть з'являтися праворуч від верхнього і нижнього фрагментів символу інтеграла, як у випадку символу інтеграла 212 у виразі 102. У випадку невизначених інтегралів ні нижня, ні верхня межі інтегрування в символ не включені. В інших випадках, таких як подвійний інтеграл 214 у виразі 102, межі інтегрування представлені більш абстрактно, в цьому випадку – великою літерою, яка з'являється під парою символів інтеграла. Як правило, межі інтегрування виражаються відповідно до розмірів шрифту, які істотно менше розміру шрифту символу інтеграла, але в різних математичних виразах відмінність шрифтів може бути різною. Межі інтегрування можуть являти собою окремі символи чи можуть складати цілі математичні вирази. Прості способи признакового розпізнавання і розпізнавання шляхом накладення еталона складно застосовувати до математичних виразів, в яких використовуються набори символів з добре визначеними значеннями, але у великій кількості різних потенційних зразків.

Іншою проблемою математичних виразів є наявність підрядкових і надрядкових символів. Наприклад, позначення функції 216 в рівнянні 104 представлено функцією f_k, яка приймає один параметр z. Однак існують і додаткові можливі інтерпретації. Наприклад, вираз може означати множення числа f_k на укладений в дужки вираз z або альтернативно він може бути інтерпретований як добуток трьох чисел f, k і z. У багатьох випадках різниця в розмірах шрифту між основним символом і надрядковим в різних виразах може варіюватися в широких межах, і це ж відноситься до відносних положень основного і надрядкового символів. Ще більші труднощі можуть виникати, коли підрядкові і надрядкові написання основних символів самі по собі представляють математичні вирази. Наприклад, підрядковий символ «E» 218 в рівнянні 204 в якості підрядкового символу має вираз q^k-1. Однак автоматичній OCR-системі може бути невідомо, чи є все цей вислів підстрочним, або ж підстрочним є q, а значення k-1 являє собою множник «E» або потенційно будь-яким чином пов'язано з наступним символом (z).

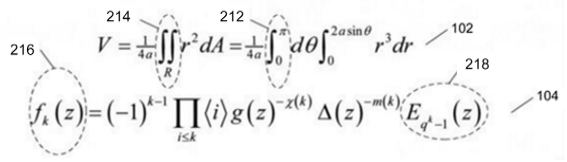


Рис. 2

Зазначені вище приклади являють собою лише невелику частину з безлічі проблем, створюваних математичними виразами для OCR-системи. Математичні вирази по суті мають довільне число рівнів вкладення, в них використовується безліч типів спеціальних символів, що мають різні значення і використання в різних контекстах, і застосовується безліч специфічних умовних позначень для представлення конкретних концепцій. Коли математичні символи представлені в межах електронних документів, наприклад, з використанням засобів Microsoft Equation Editor, Math Type Equation Editor або LaTeX, необхідно розуміти точне значення кожного символу, а також рівні вкладення, на яких знаходяться символи. В якості одного прикладу, у багатьох системах уявлення математичних виразів в цифровому вигляді парні квадратні, фігурні або круглі дужки вказуються шляхом одного вихідного введення, причому символи з'являються в межах парних квадратних, фігурних або круглих дужок після введення пари дужок. Щоб правильно уявити символ, OCR-система повинна розпізнати парні фігурні, квадратні або круглі дужки на кожному рівні вкладення.

4. Вирішення проблеми розділення атомарних одиниць математичного

виразу та розпізнавання усього виразу

4.1 Пошук границь розбиття за допомогою розбиття зображення на масив суміжних паралельних смуг

На Рис. 3-4 проілюстрований один із способів розбиття на блоки математичного виразу, який розділяє зображення або фрагмент зображення, що містить математичний вираз, на блоки або розбиття нижчого рівня. На Рис. 3 представлено рівняння 102, зображене на Рис. 1, на яке накладено безліч вертикальних паралельних ліній. Дані вертикальні лінії, такі як вертикальна лінія 302, ділять зображення математичного виразу 102 на безліч суміжних паралельних вертикальних смуг. Для ясності ілюстрації ступінь деталізації розбиття на смуги, або ширина смуг, в представленому на Рис. 3 прикладі відносно велика. Однак в деяких способах реалізації можна використовувати смуги шириною в один або два пікселя. У способі розбивки на блоки після розбиття зображення рівняння на вертикальні смуги можна врахувати кількість пікселів в межах кожної смуги, відповідної символам, щоб побудувати гістограму послідовних інтервалів уздовж горизонтальної осі зображення виразу, яка показує кількість пікселів, відповідних символів для кожного інтервалу осі x, представляючи область перетину вертикальної смуги з горизонтальною віссю або альтернативно частку пікселів, відповідних символів, в кожній вертикальній смузі. Потім в способі виявляються всі поодинокі вертикальні смуги і суміжні множини вертикальних смуг, в яких кількість пікселів, відповідних символів, менше порогового кількості пікселів, в якості потенційних границь розбиття.

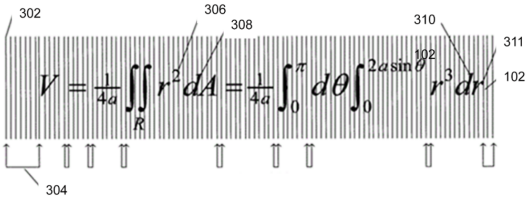


Рис. 3

На Рис. 4 проілюстровано горизонтальне розбиття першого рівня зображення математичного виразу 102, зображеного на Рис. 1. Лінії розбиття, такі як лінія розбиття 460 центровані в межах фрагментів зображення, через які можна побудувати вертикальні або похилі білі смуги (пропусків). У деяких випадках, таких як у випадку розбиття 462 і 464, кожне розбиття, або блок, містить одиночний символ. В інших випадках, наприклад при розбитті 466, в результаті горизонтального розбиття не можуть бути отримані поодинокі символи через наявність горизонтальної дробової риски$\frac{1}{4a}$.

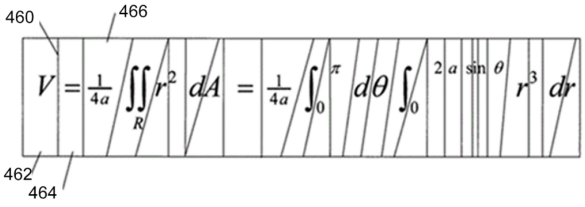


Рис. 4

Як представлено на Рис. 5, аналогічний спосіб розбиття можна використовувати для вертикального розбиття зображення або фрагмента зображення, що включає математичний вираз. На Рис. 5 фрагмент зображення 570, що містить дріб, $\frac{1}{4a}$ можна вертикально розділити на три розбиття 572-574. Застосування етапу вертикального розбиття до розбиття 566, отриманому при горизонтальному розбитті початкового зображення, дозволяє утворити розбиття другого рівня, два з яких включають поодинокі символи, а один 574 включає два символи. Потім горизонтальне розбиття розбиття другого рівня 574 дозволяє утворити два розбиття третього рівня, кожен з яких містить одиночний символ. Таким чином, почергове рекурсивне застосування горизонтального і вертикального розбиття, або розбиття на блоки, можна використовувати для рекурсивного розбиття зображення або фрагмента зображення, що містить математичний вираз, на ієрархічну множину блоків зображення, причому найменший блок завжди містить одиночний символ.

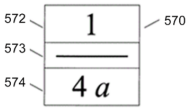


Рис. 5

4.2 Загальний алгоритм і рекурсивна підпрограма

На Рис. 6-7 представлені блок-схеми, що ілюструють один із способів обробки зображення документа, що містить математичний вираз. На Рис. 6 представлена блок-схема для підпрограми «математичний вираз», яка в якості аргументу приймає фрагмент зображення документа, що містить математичний вираз, і в результаті генерує представлення математичного виразу, що міститься на фрагменті зображення, у вигляді графа. На етапі 602 підпрограма «математичний вираз» приймає фрагмент зображення, що містить математичний вираз. Фрагмент зображення може бути вказаний посиланням на бінарне зображення документа, а також позначеннями точок в межах бінарного зображення документа, які вказують на прямокутник або багатокутник, що містить фрагмент зображення, що містить математичний вираз. На етапі 604 підпрограма «математичний вираз» привласнює локальної змінної напрямок «горизонтальне», а для локальної змінної список - нульове значення. На етапі 606 підпрограма «математичний вираз» викликає підпрограму «рекурсивне розбиття на блоки» для виконання рекурсивного розбиття на блоки отриманого фрагмента зображення, як описано вище. Підпрограма «рекурсивне розбиття на блоки» також перетворює списки фрагментів зображень, що містять символи, в уявлення у вигляді графа фрагментів вираження в межах математичного виразу і повертає підсумкове представлення у вигляді графа всього математичного виразу. На етапі 608 уявлення математичного виразу у вигляді графа, повернене підпрограмою «рекурсивне розбиття на блоки», використовується для породження уявлення математичного виразу в цифровому вигляді, відповідного математичного виразу, яке міститься в отриманому фрагменті зображення.

Рис. 6

На Рис. 7 представлена блок-схема підпрограми «рекурсивне розбиття на блоки», що викликається на етапі 606, представленому на Рис. 7. На етапі 710 підпрограма «рекурсивне розбиття на блоки» приймає фрагмент зображення s, напрямок d і посилання на список l. На етапі 712 підпрограма «рекурсивне розбиття на

блоки» привласнює локальної змінної localL нуль. На етапі 714 підпрограма «рекурсивне розбиття на блоки» викликає підпрограму «розбиття на блоки» для проведення розбивки фрагмента зображення на блоки в напрямку, зазначеному отриманим аргументом d. Як описано вище, d може вказувати або на горизонтальне, або на вертикальне розбиття на блоки. Потім в циклі for на етапах 716-722 розглядається кожен блок, що згенерував підпрограмою «розбиття на блоки». На етапі 717 поточний аналізований блок b піддається розпізнаванню символів за допомогою OCR. Якщо розпізнавання символів за допомогою OCR успішно розпізнає символ в блоці b, як визначено на етапі 718, то на етапі 720 підпрограма «рекурсивне розбиття на блоки» формує одноелементний список, який включає один елемент, що представляє розпізнаний символ. В іншому випадку на етапі 719 підпрограма «рекурсивне розбиття на блоки» рекурсивно викликає саму себе для проведення наступного рівня розбиття на блоки в напрямку розбиття на блоки, протилежному напрямку розбиття на блоки, отриманому в якості аргументу d, зазначеного на Рис. 7 позначенням !d. Іншими словами, коли аргумент d має значення «горизонтальне», то !d має протилежне значення «вертикальне», а коли аргумент d має значення «вертикальне», то !d має протилежне значення «горизонтальне». Список, згенерований на етапі або 720, або 719, додається на етапі 721 до списку, на який посилається локальна змінна localL. Після того як всі блоки, згенеровані в результаті виклику підпрограми «розбиття на блоки» на етапі 714, були розглянуті в циклах for на етапах 716-722, підпрограма «рекурсивне розбиття на блоки» викликає на етапі 724 підпрограму «обробка на основі графа» для проведення на основі графа обробки списку, на який посилається локальна змінна localL. Нарешті, на етапі 726 список, на який посилається локальна змінна localL, додається до списку, на який посилається аргумент l.

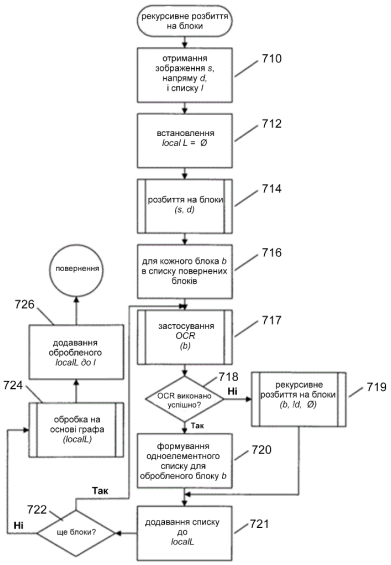


Рис. 7

5. Висновок

В даній статті було представлено ряд складностей, що виникають при перетворенні зображень документа, що містить математичні вирази та способи вирішення цих проблем.

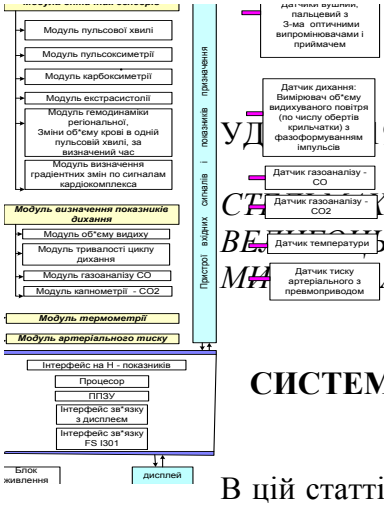
Для вирішення проблеми розбиття зображення математичного виразу на атомарні блоки для подальшого використання їх у OCR-розпізнаванні було продемонстровано застосування рекурсивно-блочного і заснованого на графі підходу.

Знаходження границь розбиття здійснюється за допомогою розбиття усього зображення на масив суміжних паралельних смуг. До усіх отриманих об'єктів застосовується OCR-розпізнавання. До отриманих графічних об'єктів, OCR-розпізнавання яких не дає результатів необхідно застосувати повторний пошук границь розбиття за допомогою масиву суміжних палалельних смуг перпендикулярної орієнтації.

Список літератури

1. Фу К. Структурные методы в распознавании образов. Издательство "МИР", Москва, 1977.
2. Nicholas Matsakis. Recognition of handwritten mathematical expressions. Master’s thesis, Massachusetts Institute of Technology, Cambridge, MA, May 1999.

3. Ernesto Tapia: Understanding Mathematics: A System for the Recognition of On-Line Handwritten Mathematical Expressions, Berlin, 2004
4. Eva Gallardo Perez: 2D Grammar Extension of the CMP Mathematical Formulae On-line Recognition System, Research Reports of CMP, Czech Technical University in Prague, No. 3, 2009



УД[illegible]19

СТ[illegible]УК О. П.,
ВЕ[illegible]ЬКИЙ Д, В.,
МИ[illegible]А.Г.

СИСТЕМА МОНІТОРИНГУ ТА ПРОГНОЗУВАННЯ СТАНУ ДИНАМІЧНИХ ОБ'ЄКТІВ

В цій статті розглянуто практичне застосування методу моніторингу та прогнозування стану динамічних об'єктів на прикладі функціонування портативного приладу для моніторингу показників стану людей, інтоксикованих чадним газом (моноксидом вуглецю) чи іншими випарами, шкідливими для людини в польових умовах із забрудненим довкіллям. Демонструється розроблені алгоритми і програмні засоби та формальні процедури для неінвазивного отримання кількісних показників карбоксигемометрії, пульсоксиметрії в потоках крові, пульсової хвилі і її кровонаповнення, дихання, температури тіла, діяльності серця та інших показників, важливих для оцінки стану людини, їх візуалізації на екрані монітора.

Ключові слова: ДИНАМІЧНИЙ ОБ'ЄКТ, МОНІТОРИНГ, ПОКАЗНИКИ СТАНУ ОРГАНІЗМУ, АЛГОРИТМИ, ПРОГРАМИ, МІКРОПРОЦЕСОР, ДИСПЛЕЙ, КЕРУВАННЯ

This article discusses the practical application of the method of monitoring and forecasting the state of dynamic objects by the example of the functioning of a portable device for monitoring indicators of the state of people who are intoxicated with carbon monoxide or other fumes harmful to humans in the field with polluted environments. The developed algorithms and software tools and formal procedures for non-invasive quantification of oximetry, pulse oximetry in blood flows, pulse wave and blood filling, breathing, body temperature, heart activity and other indicators important for assessing a person's condition and their visualization on a monitor screen are demonstrated.

Keywords: DYNAMIC OBJECTS, MONITORING INDICATORS OF THE BODY, ALGORITHMS, PROGRAMS, MICROPROCESSORS, DISPLAY, MANAGEMENT

.

1. Функціональне призначення.

Гостре отруєння монооксидом вуглецю є однією з найбільш розповсюджених і важких форм інтоксикації організму людей, які змушено попадають в зони пожеж, їх гасіння, чи у працівників, зайнятих у промисловості, побуті, або через забруднення довкілля. Аварійно-рятувальні служби, швидка медична допомога, опікові центри, інші медичні і ветеринарні заклади виявляють потребу в новітніх технологіях і приладах для раннього діагностування, лікування і зниження ризику летальної дії на організм людини перебування в середовищах з інтенсивним виділенням моноксиду вуглецю, температурним враженням, іншими чинниками. Мета роботи – створити програмне забезпечення монітора стану інтоксикованих чадним газом в зонах враження (Монітор МР-01)

.

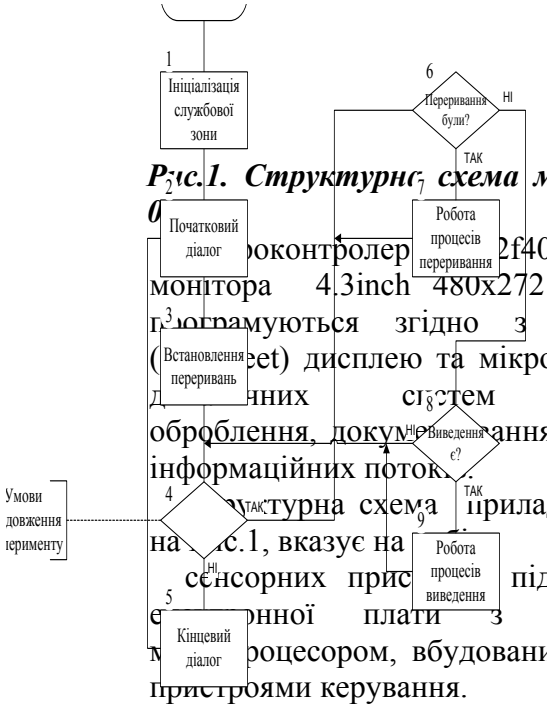


Рис.1. Структурна схема монітора МР-01

...роконтролер ...2f407 discovery і монітора 4.3inch 480x272 Touch LCD, ...ограмуються згідно з інструкціями (...eet) дисплею та мікропроцесору як ...ючних систем отримання, оброблення, докум...ування і візуалізації інформаційних потоків.

...структурна схема ...приладу, приведена на ...с.1, вказує на ... сенсорних прис... підключених до ...онної плати з встановленим ...роцесором, вбудованим дисплеєм і пристроями керування.

2. Алгоритм програмного забезпечення.

Алгоритм програми передбачає підтримку функціонування апаратної частини виробу «МР-01» по введенню сигналів від датчиків, збору і накопиченню вхідної інформації, для діалогу з користувачем і керування процесом проведення вимірювання обстежуваного об'єкта, а також забезпечення зберігання обробленої інформації. Програма може функціонувати у виробі «МР-01» як з повним набором функціональних модулів, що входить до складу, так і з певною його частиною.

Програмний продукт складено по модульному принципу. Окремі групи модулів об'єднані в бібліотеки, з яких власне і формується операційна система.

Для узгодження часу роботи кожної із складових частин виробу «МР-01» в реальному масштабі часу застосовано метод формування черги запитів обробки інформації, надходить по принципу "перший зайшов– перший оброблено" із відповідним пріоритетом оброблення черги

Пріоритет обробки черги запитів наступний:

- черга запитів від модулів, що формують вхідну інформацію;
- чергу запитів виводу інформації на екран дисплею;
- чергу запитів на формування вихідної інформації вимірювання і моніторингу.

Перехід до обробки наступної черги запитів здійснюється тільки після завершення

обробки всіх елементів черги з більш високим пріоритетом.

Структурна загального алгоритму роботи програмного забезпечення «МР_01» та зв'язки з функцій блоками і їх функціями показана на Рис. 2.

1. Блок 1 ініціює включення модулів приладу, очищує службову зону ОЗУ з певними адресами і заповнює її векторами переривання системних функцій «МР-01», забезпечуючи тим самим зупинки роботи виробу при порушенні режиму роботи.

2. Блок 2 керує діалогами з оператором, який користуючись підказками програмного забезпечення (ПО) встановлює необхідні параметри майбутнього вимірювання.

3. Блок 3 ініціює роботу всіх модулів блоку електроніки виробу в реальному масштабі часу.

Рис. 2. Структура алгоритму програмного забезпечення виробу «МР-01»

4. Блок 4 перевіряє чотири умови перебігу вимірювання: закінчення необхідного числа повних хвилин вимірювання, набір достатньої статистики

по системі дихання, по кардіосистеми, а також наявність сигналу про примусове закінчення вимірювання оператором. Перші три умови об'єднані по логічній «І», а четверта умова - по логічному "або" з першими трьома, при позитивному рішенні умов відбувається перехід на блок 6, а при негативному - на 5 Блок.

5. Блок 5 здійснює діалог з оператором по завершенню вимірювання і збереження його результатів. Після закінчення роботи блоку управління передається на блок 2.

6. Блок 6, де переглядається черга сигналів «МР - 01»: при порожній черзі відбувається перехід до блоку 8, а при наявності елементів у черзі - перехід до блоку 7.

7. Блок 7 по параметрах, отриманих з черги запитів, ініціює один з п'яти процесів

обробки переривання з наступним пріоритетом: - обробка по дихальному

процесу; - обробка переривання по інтервалу пульсової хвилі RR; - обробка

секундних переривань і переривання від регістра запитів; - обробка хвилинних переривань від АЦП.

При порожніх чергах відбувається перехід на блок 4, а при наявності запитів відбувається перехід на блок 9.

8. Блок 8 при черзі системних запитів послідовно перевіряє наявність запитів в трьох наступних чергах: черги виводу діагностик, черги виводу інформації ("картинок» з поточного моніторингу).

9. У блоці 9 виконується обробка одного з елементів черги запитів відповідно до пріоритетості, зазначеного в пункті 8. при обробці черги діагностик формується висновок на екрані дисплея поточної діагностики і перехід на блок 4.

При порожній черзі діагностик послідовно обробляється черга формування вихідної інформації або висновок сформованої інформації за запитом оператора. Після задоволення чергового запиту відбувається перехід на блок 4.

Повний опис програми обмежується визначеним обсягом публікації. Тому як приклад на мал. 4 приведено алгоритми контролю за змінами показників окремих функцій: дихання, концентрації карб оксигемоглобіну, температури та інших.

3.Вимірювання показників дихання.

Вхідною інформацією є два сигнали,

що надходять від тахометричного датчика об’єму дихання, тривалості фаз видиху і вдиху.

Базовими є сигнали від двох фототранзисторів, причому якщо першими

приходять сигнали від фототранзистора №1, то це фаза видиху, якщо першим буде

сигнал від фототранзистора №2. то це фаза вдиху.

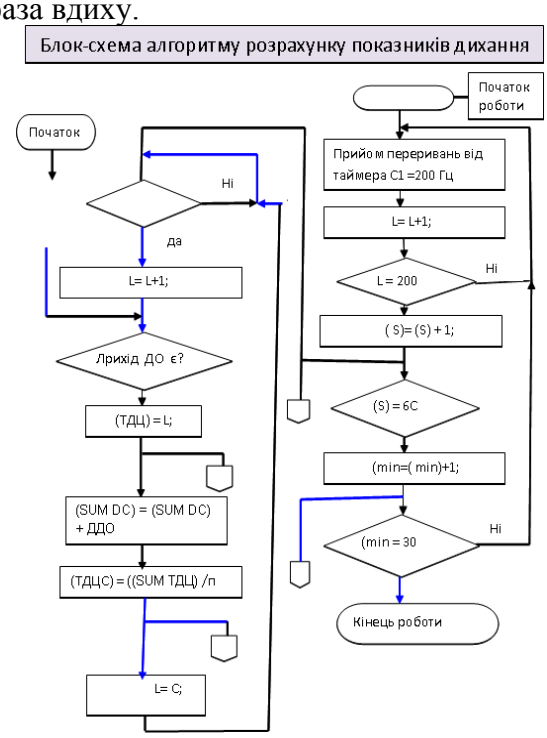


Рис. 3. Схема алгоритму вимірювання показників дихання.

Водночас реєструться тривалість часу між запуском фототранзисторів № 1 і № 2.

Вихідні сигнали з фото транзистора нормуються конкретним імпульсом, сталої амплітуди і тривалості для обліку швидкості руху і числа обертів крильчатки, причому тільки до того часу поки фіксується приріст швидкості видиху чи вдоху, поки значення тривалості інтервалу часу між імпульсами від

фототранзисторів №1 и №2 зростає. до часу Коли ж приріст зупиняється, число обертів стале, або починає зменшуватись їх облік припиняється. Надалі кількість імпульсів NR множиться на нормоване значення об'єму одного оберту SR, і визначається об'єм дихання (видиху) ДО, мл, та інші показники. Результати вимірювання показників дихання виведено на вбудований дисплей у формі числа, гістограми, графіки за вибором.

Вимірювання показників сатурації кисню і карбоксигемоглобіну в потоках крові здійснюється за алгоритмом, приведеним на мал. 5.

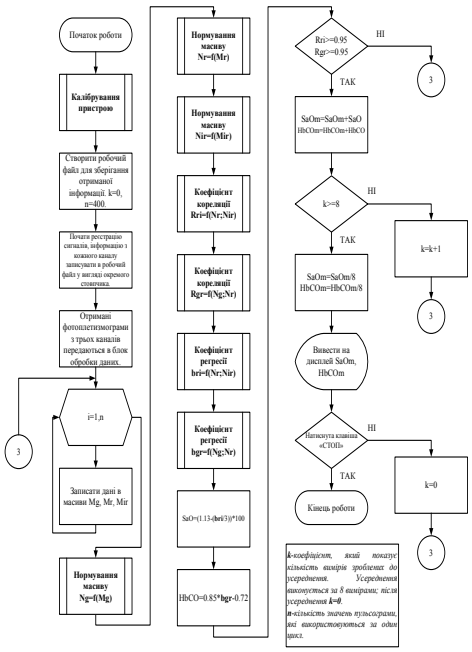


Рис. 4. Схема алгоритму визначення сатурації потоків крові киснем і карбоксигемоглобіном SpO_2 і НвСО.

До інформації, що пов'язана з карбоксигемометрією, відносяться:

SpO_2 - величина сатурації артеріальної крові киснем;

∇SpO_2 - градієнт величини сатурації артеріальної крові киснем;

SpCO - величина відносної концентрації карбоксигемоглобіну в артеріальній крові;

$\nabla SpCO$ - часовий градієнт величини відносної концентрації карбоксигемоглобіну в артеріальній крові.

Наведені окремі результати проведеної роботи по створенні програмного продукту для чисельного аналізу динамічного об'єкту, якими є потоки інформації в моніторі показників стану людей, інтоксикованих чадним газом (моноксидом вуглецю) чи іншими випарами, шкідливими для людини в польових умовах із забрудненим довкіллям та вказують на

суттєву роль якості програмного продукту,

в поєднанні можливостей мікропроцесорної техніки, сенсорних пристроїв і сучасних засобів документування і візуалізації для аналізу діяльності складних динамічних систем, в тому числі біологічних.

Список літератури.

1. Б.І. Мокін, В.Б. Мокін, О.Б.Мокін Математичні методи ідентифікації динамічних систем. Навчальний посібник, Вінниця: ВНТУ, 2010, 260 с.

2. Кость Я., І. Хвищун. Розробка програмного забезпечення для моделювання динамічних систем із жорсткими математичними моделями. //Електроніка та інформаційні технології. 2012. Випуск 2. С. 184–196

3. Велигоцький Д.В., Стельмах Н.В., Єсьман С.С., Мамілов С.О. (Україна); Інститут прикладних проблем фізики і біофізики НАН України. – № 201111247; Заявл. 22.09.11; Опубл. 12.03.12.

4. Мамілов С.О., Єсьман С.С., Велигоцький Д.В., Голуб Д.В. «Спосіб дезактивації карбоксигемоглобіна крові» Патент на корисну модель №86015. від 10.12.2013.

5. Велигоцький Д.В., Стельмах Н.В., Єсьман С.С., Мамілов С.О. (Україна); Інститут прикладних проблем фізики і біофізики НАН України. – № 201111247; Заявл. 22.09.11; Опубл. 12.03.12.

УДК 519.854.2

ЖУРАКОВСЬКА О.С.
ХІЛЬЧЕНКО І.О.

МЕТОДИ ВИРІШЕННЯ ЗАДАЧІ МІНІМІЗАЦІЇ СУМАРНОГО ЗВАЖЕНОГО ВІДХИЛЕННЯ ВІД ЗАДАНИХ ДИРЕКТИВНИХ СТРОКІВ ВИКОНАННЯ РОБІТ НА ОДНОМУ ПРИЛАДІ

В даній статті розглядається задача мінімізації сумарного зваженого відхилення виконання робіт від директивних строків на одному приладі. Наведено опис евристичного алгоритму розв'язання задачі. Для оцінки його ефективності використовується алгоритм заснований на методі гілок та меж. Наведено план проведення експериментів.

This paper addresses minimizing total weighted deviation penalty for direct due date single-machine scheduling problem. The heuristic algorithm is proposed for solving this problem. To evaluate its performance is used an algorithm based on the method branches and bounds. An outline of experiments is shown.

1. Вступ

В даній роботі розглядається задача мінімізації сумарного зваженого відхилення виконання робіт від директивних строків на одному приладі. В цій задачі при впорядкуванні робіт враховуються штрафи як за випередження термінів виконання робіт, що може призвести до збільшення складських витрат, так і за запізнення. У статті наведено основні методи, які використовуються для вирішення поставленої задачі, та проаналізовано їх переваги та недоліки. Приведений евристичний алгоритм рішення задачі та алгоритм заснований на методі гілок та меж. Наведено план проведення комп'ютерних експериментів.

2. Огляд методів вирішення задачі

Задача мінімізації сумарного зваженого відхилення виконання робіт від директивних строків на одному приладі відноситься до класу важкорозв'язуваних задач, і тому не відомо поліноміального алгоритму вирішення цієї задачі. Слід зазначити, що часткові випадки цієї задачі, такі як мінімізація сумарного зваженого запізнення (випадок з нульовими штрафами за випередження), також відносяться до класу NP-задач [1]. У зв'язку з цим основними підходами для вирішення поставленої задачі являються: евристичні алгоритми, метод гілок та меж.

Метод гілок та меж [2] широко використовується для знаходження оптимального розв'язку задач дискретної оптимізації. Ідея методу гілок та меж базується на впорядкованому переборі можливих розв'язків задачі та відсіканні тих розв'язків, що являються безперспективними з точки зору цільової функції. В процесі роботи алгоритму гілок та меж будується дерево рішень, шляхом розбиття множини допустимих розв'язків задачі на підмножини, які далі перевіряються, щоб з'ясувати чи містить дана підмножина оптимальний розв'язок. Щоб побудувати алгоритм заснований на методі гілок та меж необхідно визначити спосіб побудови дерева рішень та спосіб оцінки. Недоліком методу гілок та меж є те, що, як правило, трудомісткість цього методу експоненційно росте з ростом розмірності задачі.

Так Абдул-Разак та Поттс [3] у своїй роботі запропонували алгоритм гілок та меж для рішення задачі сумарного зваженого відхилення виконання робіт від директивних строків на одному приладі. Для обчислення нижньої оцінки вони запропонували алгоритм динамічного програмування.

У роботі [4] описано алгоритм гілок та меж для цієї задачі у випадку, коли всі роботи мають спільний директивний строк.

Евристичні алгоритми базуються на так званому прийомі зниження вимог. При цьому використовуються певні раціональні міркування без строгого обґрунтування і не гарантується, що знайдений розклад є оптимальним. Серед евристичних алгоритмів широкого поширення набули методи локального пошуку, які базуються на ітеративному покращенні початкового рішення доки це можливо. Серед методів локального пошуку виділяють табу-пошук, метод імітації відпалу. Іншим різновидом евристичних алгоритмів являються жадібні алгоритми, в яких з множини можливих альтернатив обирається та, що дає найкращий приріст критерію. Перевагою евристичних алгоритмів є швидкість їх роботи, що дозволяє використовувати їх для розв'язку великих задач, та зручність реалізації на ЕОМ. Однак евристичні алгоритми не можуть гарантувати, що знайдений розв'язок буде оптимальним.

Ов та Мортон [5] запропонували сімейство евристичних процедур для рішення цієї задачі без простоїв приладу.

Павлов, Місюра та Мельников [6] розробили алгоритм локального пошуку для цієї задачі. Їх алгоритм базується на використанні резервів незапізнених завдань, визначення моменту часу запуску робіт на виконання.

3. Постановка задачі

Нехай задано множину незалежних завдань. Для кожного завдання $i, i = \overline{1,n}$ відомо: p_i - тривалість виконання завдання; d_i - директивний строк; α_i - штраф за виконання раніше директивного строку; β_i - штраф за виконання пізніше директивного строку. Завдання надходять у систему одночасно. Заборонені переривання та одночасне виконання завдань. Необхідно побудувати розклад виконання завдань одним приладом за критерієм мінімізації сумарного штрафу за відхилення від директивних строків:

$$\sum_{i=1}^{n} (\alpha_i E_i + \beta_i T_i) \to \min, \qquad (3.1)$$

де $E_i = \max(0, d_i - C_i)$ - випередження виконання завдання i відносно директивного строку;

$T_i = \max(0, C_i - d_i)$ - запізнення виконання завдання i відносно директивного строку;

C_i - момент завершення виконання завдання i.

4. Метод гілок та меж

Схему методу гілок та меж можна описати за допомогою таких понять: галуження, оцінка, рекорд.

Множина всіх допустимих розв'язків X розбивається на підмножини $X_i \subseteq X, i = \overline{1,r}$ таким чином, щоб $\bigcup_{i=1}^{r} X_i = X$. Таким чином початкова задача розбивається на підзадачі, які в свою чергу теж розбиваються на підзадачі. Цей рекурсивний процес називається галуженням. В процесі галуження будується дерево пошуку оптимального рішення.

1. ініціалізація початкового розкладу і розбиття розкладу на блоки;
2. впорядкування робіт в множинах *E* та *T* в блоках;
3. впорядкування робіт за допомогою *ET*-перестановок.

Для кожної з підмножин множини допустимих розв'язків визначається оцінка $\gamma(X)$ (нижня/верхня межа) значень цільової функції. Для обчислення оцінки може розв'язуватися деяка оптимізаційна задача простої структури.

Рекордом f^* називається найкраще знайдене значення цільової функції. При чому підмножина допустимих розв'язків X_i, оцінка якої гірша за поточний рекорд, вилучається з подальшого розгляду, оскільки вона не містить оптимального розв'язку.

Опишемо спосіб галуження та оцінки для задачі мінімізації сумарного зваженого відхилення виконання робіт від директивних строків на одному приладі. Галуження будемо виконувати по компонентам, тобто поступово будувати розклад, додаючи до поточного розкладу одну з невпорядкованих робіт. Таким чином на кожному етапі галуження вихідна множина розв'язків X' буде розбиватись на $n-k$ підмножин, де k - кількість впорядкованих робіт в вихідній

множині $X^{'}$. Для кожної множини розбиття обчислюємо оцінку. В якості оцінки множини X використаємо значення цільової функції для впорядкованих робіт:

$$\gamma(X) = \sum_{i=1}^{k} \alpha_{[i]} E_{[i]} + \beta_{[i]} T_{[i]} = \\ = \gamma(X^{'}) + \alpha_{[k]} E_{[k]} + \beta_{[k]} T_{[k]}, \quad (4.2)$$

де $\gamma(X^{'})$ - оцінка вихідної підмножини;

$[i]$ - порядковий номер роботи в розкладі.

5. Евристичний алгоритм

Введемо деякі визначення, що знадобляться для опису евристичного алгоритму.

Множина E – послідовність робіт в розкладі, що випереджають директивні строки (або виконуються точно в строк). Така послідовність може складатись з однієї роботи.

Множина T – послідовність робіт в розкладі, що запізнюються відносно директивних строків. Така послідовність може складатись з однієї роботи.

Блок – послідовність робіт, що складається із множини *E* та множини *T*. Таким чином розклад можна представити послідовністю блоків.

Розглянемо властивості задачі, які використовуються для побудови алгоритму.

Властивість 1. Якщо в множині *E* існують завдання i та j, що займають сусідні позиції (*i* знаходиться перед *j*) і якщо після їхньої перестановки вони також залишаються в множині *E* (тобто випереджаючими), то для того, щоб перестановка не привела до покращення критерію, необхідно, щоб завдання в множині *E* були впорядковані за зростанням пріоритетів α / p.

Властивість 2. Якщо в множині *T* існують завдання *g* та *k*, що займають сусідні позиції (*g* знаходиться перед *k*) і якщо після їхньої перестановки вони також залишаються в множині *T*, то для того, щоб перестановка не привела до покращення критерію, необхідно, щоб завдання в множині *T* були впорядковані за спаданням пріоритетів β / p.

Властивість 3. Розглянемо дві суміжні роботи i та j, $d_i \le d_j$. Якщо при спробі розташувати ці роботи, щоб час закінчення їх виконання співпадав з директивним строком, тобто $C_i = d_i$ та $C_j = d_j$ існує конфлікт (не вдається одночасно виконати ці умови), то існує оптимальний розклад, в якому $C_i = d_i$ або $C_i = d_i$ [7].

Евристичний алгоритм можна розбити на такі етапи:

1. ініціалізація початкового розкладу і розбиття розкладу на блоки;
2. впорядкування робіт в множинах *E* та *T* в блоках;
3. впорядкування робіт за допомогою *ET*-перестановок.

5.1 Ініціалізація початкового розкладу

Опишемо процедуру ініціалізації початкового розкладу. Для кожної роботи встановимо пріоритет $P_i, i = \overline{1,n}$, який вираховується таким чином: порівнюємо усі можливі пари робіт, і, якщо робота *i* передує роботі *j*, то P_i збільшується на 1 та P_j зменшується на 1. Щоб дізнатись, яка з двох робіт має виконуватись раніше розглянемо ці роботи при припущені, що вони є суміжними. Тоді потрібно розглянути 2 випадки:

1. Робота i передує j, причому $d_i \le d_j$. Нехай $F(i, j)$ мінімальний штраф при умові, що робота *i* передує *j*. Оскільки за властивістю 3 одна з цих робіт має завершуватись точно в строк, то обраховуємо штрафи для 2 випадків $C_i = d_i$ та $C_j = d_j$ і тоді покладемо, що $F(i, j)$ дорівнює мінімальному з них.
2. Робота j передує i, причому $d_i \le d_j$. Аналогічно до першого випадку знаходимо $F(j, i)$.

Якщо $F(i, j) \le F(j, i)$, то P_i збільшується на 1 та P_j зменшується на 1, інакше навпаки. Після обчислення пріоритетів робіт впорядковуємо їх за не зростанням пріоритетів. Якщо існують роботи з однаковим пріоритетом, то впорядковуємо такі роботи за зростанням директивних строків. Отриманий розклад розбиваємо на блоки. Трудомісткість

описаної процедури складає $O(n(n-1)/2) = O(n^2)$.

5.2 Впорядкування робіт в множинах *E* та *T*

Після розбиття розкладу на блоки впорядковуємо роботи всередині множини *E* та *T* в кожному блоці.

Роботи в множині *E* повинні бути впорядкованими за зростанням пріоритетів α_i / p_i. Це правило базується на такій властивості задачі (властивість 1): якщо робота *j* слідує за роботою *i* в множині *E*, то після їхньої перестановки (якщо вони залишаються випереджаючими) критерій збільшиться, якщо завдання *i* має менший пріоритет, ніж завдання *j*.

Роботи в множині *T* повинні бути впорядкованими за спаданням пріоритетів β_i / p_i. Це правило базується на аналогічній властивості задачі (властивість 2): якщо робота *j* слідує за роботою *i* в множині *T*, то після їхньої перестановки (якщо вони залишаються запізнюючими) критерій збільшиться, якщо робота *i* має більший пріоритет, ніж робота *j*.

Слід зазначити, що в процесі такого впорядкування можуть утворюватися нові блоки. В такому разі в кожному новому блоці потрібно впорядкувати роботи таким же чином.

Фактично проводиться сортування робіт за відношенням їх штрафів до тривалості виконання, а отже ефективність цього етапу евристичного алгоритму залежить від вибраного алгоритму сортування.

5.3 Проведення *ET*-перестановок

ET-перестановка - це перестановка завдання *i* на позицію після завдання *g*, якщо виконуються такі умови:

1. завдання *i* належить множині *E*, завдання *g* належить множині *T*, і вони обидва належать одному блоку (завдання *i* займає більш ранню позицію, ніж завдання *g*);
2. директивний строк завдання *i* пізніший за момент початку виконання завдання *g*: $d_i \geq t_g$, де $t_g = C_g - p_g$.

В результаті виконання ET-перестановок завдання *g* із множини *T* переміщається на більш ранні позиції і може перейти до складу множини *E*.

Для зменшення критерію мають сенс тільки перестановки всередині блоків. Перестановки між блоками можуть тільки збільшити критерій – перестановка запізненого завдання з множини *T* на позицію після випереджаючого завдання з множини *E* наступного блоку зробить запізнене завдання ще більш запізненим, а випереджаюче – ще більш випереджаючим, тобто значення критерію збільшиться.

Тому в алгоритмі послідовно розглядаються всі блоки завдань, і для кожного блоку здійснюється впорядкування.

6. План проведення експериментів

Метою проведення комп'ютерних експериментів є оцінка ефективності і точності розробленого евристичного алгоритму. Для цього згенеруємо тестові задачі різної розмірності (від 10 до 20 робіт). Для кожної роботи в задачі випадковим чином згенеруємо:

1. тривалість виконання;
2. директивний строк;
3. штраф за випередження;
4. штраф за запізнення.

Загалом згенеруємо по 50 екземплярів задачі однакової розмірності, щоб визначити середній час роботи алгоритму та середнє відхилення від оптимального значення.

7. Висновки

В цій роботі розглянута задача мінімізації сумарного зваженого відхилення виконання робіт від директивних строків на одному приладі. У зв'язку зі складністю задачі основними напрямками її вирішення являються евристичний та мета-евристичний підходи.

Запропоновано евристичний алгоритм, який заснований на розбитті розкладу на блоки та впорядкування робіт всередині кожного блоку. Оскільки цей алгоритм не може гарантувати знаходження оптимального рішення, пропонується оцінка його ефективності, визначена на основі експериментальних досліджень. Для отримання точних розв'язків задачі застосовується метод гілок та меж.

Наведено план подальших експериментів.

Список літератури

1. Baker K.R, Scudder G.D. Sequencing with earliness and tardiness penalties: a review. Operations Research, Vol. 38, No. 1, 22–36
2. Лазарев А.А., Гафаров Е.Р. Теория расписаний. Задачи и алгоритмы. М.: Физический факультет МГУ, 2011, с. 72-75
3. Abdul-Razaq T. and C. Potts. Dynamic Programming State-Space Relaxation for Single-Machine Scheduling. – 1988 – J. Opnl. Res. Soc. 39, pp. 141-152
4. Ghaith R., Mansooreh M, Anagnostopoulos G. A branch-and-bound algorithm for the early/tardy machine scheduling problem with a common due-date and sequence-dependent setup time. Computers & Operations Research 31, 2004, pp 1727–1751;
5. Ow P., T Morton. The Single Machine Early-Tardy Problem. – 1989 – Mgnt. Sci. 35, pp. 177-191
6. Павлов О.А., Місюра О.Б., Мельников О.В. Дослідження властивостей та розв‘язання задачі «Мінімізація сумарного штрафу як за випередження, так і за запізнення відносно директивних строків при виконанні незалежних завдань одним приладом» / Вісник НТУУ ―КПІ. Інформатика, управління та обчислювальна техніка, 2008. – №48.– С.3-6
7. Kim, Yeong-Dae; Yano, Candace A. Minimizing mean tardiness and earliness in single machine scheduling problems with unequal due-dates. Naval Research Logistics, Volume 41, 1994, 913-933;

УДК 004.65

ГУБАР Б.Д.

ГЕНЕРАТОР ТЕСТОВИХ ДАНИХ ДЛЯ РЕКОМЕНДАЦІЙНИХ СИСТЕМ ОФЛАЙН-ПРОДАЖІВ

Рекомендаційні системи – аналізують інтереси користувачів і намагаються передбачити, що саме найбільш зацікавить користувача в даний момент часу. Для коректної роботи рекомендаційних систем, необхідно мати великий набір даних, що будуть основою для генерації рекомендацій. Не маючи можливості збору реальних даних, необхідно мати інструмент, що дозволить генерувати тестові дані, що будуть моделювати оцінки реальних користувачів.

Recommendation systems - analyze the interests of users and try to predict what the user most interested in a given time. For correct work of recommendation systems, it's needed for a large set of data that will be the basis for recommendations. If there is no opportunity to collect real data, it is necessary to have a tool that will generate test data that will be modeling estimates of real users.

1. Вступ

Рекомендаційні системи – аналізують інтереси користувачів і намагаються передбачити, що саме найбільш зацікавить користувача в даний момент часу [1].

Основними сферами використання таких систем – є електронна комерція та онлайн медіа ресурси. Ці компанії – є великими фахівцями в сфері IT, тому кожна компанія створює свою власну рекомендаційну системи, що відповідає її потребам. Для коректної роботи рекомендаційних систем, необхідно мати великий набір даних, що будуть основою для генерації рекомендацій.

Такі рекомендаційні системи можуть проходити тестування на реальних системах, з реальними даними від справжніх користувачів, покращуючись від версії до версії.

2. Загальний опис генератора

При створенні рекомендаційної системи, не прив'язаної до конкретної системи, не буде можливості використання реальних даних. Тому для проведення тестування системи, необхідно створити генератор тестових даних.

В будь-якій рекомендаційній системі існує 3 види сутностей [2]:

- Продукт – те що, оцінює користувач, а система надає, в якості рекомендацій;
- Користувач – інформація про певного користувача в системі;
- Рекомендація – це продукт, що рекомендується користувачу системою;
- Оцінка – в системах колаборативної фільтрації - характеризує відношення конкретного користувача до певного продукту, може бути, як явною, у вигляді оцінки, за певною шкалою, так і неявною.

Звичайно кожна предметна область рекомендаційних систем має свою структуру. Тому генератор має створювати дані різної структури.

Перші два набори сутностей – не пов'язані один з одним. Також ці дані, не впливають на рекомендації. Тому при їх генерації створюються довільні дані, без параметрів. Ці поля заповнюються даними з відкритих джерел.

Основним завданням генератора – є коректна генерація саме оцінок користувачів. Від цього залежить коректність роботи усієї рекомендаційної системи.

3. Параметри генерації.

Структура генерованих даних, повинна відповідати до предметній області. Отже необхідно передавати структуру даних в генератор, у вигляді параметра. Структура даних описується в вигляді XML файлу, відповідно до певної схеми. Кожний тег відповідає певному стовпцю в таблиці бази даних. Атрибути тегів відповідають певним параметрам таблиць, таким як обов'язковість поля та чи є поле

первинним ключем в таблиці, інші атрибути містять інформацію для генератора, таку як тип генерованих даних для стовпця чи діапазон для генерації.

Під час генерації оцінок, частина даних не є наперед визначеними константами, і їх значення необхідно генерувати за певними законами розподілу.

Такими даними є:

- кількість продуктів в чеку;
- середня вартість чеку;
- частота покупок.

В самому простому випадку ці дані можна генерувати, як набір з рівномірним розподілом в певному діапазоні значень. Але на практиці це не так. Тому, в залежності від вирішуваних завдань, необхідно вказувати в генераторі закон розподілу ймовірності для певного параметру.

4. Приклад роботи генератора

Роботу генератора буде описано на прикладі даних про покупки користувачів в продуктовому магазині.

Структура таблиць користувачів і продуктів показана в таблицях 1 і 2.

Табл. 1. Структура таблиці Користувач

Назва поля	Опис поля	Тип даних
User_id	Унікальний ідентифікатор користувача	int
Full name	Прізвище ім’я та по-батькові користувача.	Text

Табл. 2. Структура таблиці Продукт

Назва поля	Опис поля	Тип даних
Product_id	Унікальний ідентифікатор продукту	int
Product name	Назва продукту	Text
Price	Ціна продукту	real

На відміну від електронної комерції, де продаються вже готові товари, продукти харчування можуть не використовуватись самі по собі, а складатися у певні страви. Тому при обробці покупок необхідно враховувати не лише певний окремий продукт, а й інші продукти в чеку. Страви, що можна приготувати з продуктів, також необхідно зберігати в базі. Для цього генерується 2 додаткові таблиці. В першій описується сам рецепт, а в другій продукти, що необхідні для цього рецепту. Структура цих таблиць наведена в таблицях 3 і 4.

Табл. 3. Структура таблиці Рецепт

Назва поля	Опис поля	Тип даних
Recipe_id	Унікальний ідентифікатор рецепту	int
Recipe name	Назва рецепту	Text

Табл. 4. Структура таблиці РецептПродукти

Назва поля	Опис поля	Тип даних
Recipe_id	Унікальний ідентифікатор рецепту	int
Product_id	Унікальний ідентифікатор користувача	int
quantity	Кількість продукту, в рецепті	int

В системах рекомендації продуктів харчування для офлайн-продаж, єдиним джерелом оцінок – є інформація про покупки користувачів.

Для збереження цих даних необхідно згенерувати 2 таблиці: Чек – в якій описується інформація про користувача, що здійснив покупку і дату цієї покупки; ЧекПродукти – містить інформацію, про продукти, що були куплені в чеку.

Табл. 5. Структура таблиці Чек

Назва поля	Опис поля	Тип даних
Receipt_id	Унікальний ідентифікатор чеку	int
User_id	Унікальний ідентифікатор користувача	int
Date	Дата проведення покупки	date

Табл. 6. Структура таблиці ЧекПродукти

Назва поля	Опис поля	Тип даних
Receipt_id	Унікальний ідентифікатор користувача	long
Product_id	Прізвище ім'я та по-батькові користувача.	Text
Quantity	Кількість продукту, що придбав користувач.	int
Price	Ціна продукту	int

Разом з тим в цих таблицях первинними ключами є не одне поле а пара полів. Recipe_id та Product_id для таблиці РецептПродукти і поля Receipt_id та Product_id для таблиці ЧекПродукти. Також ці поля є зовнішніми ключами для таблиць Продукти, Чек і Рецепти, тому значення цих полів, повинні обиратися з множини значень полів у відповідних таблицях.

5. Результати генерації оцінок

Дані в таблицях, що відповідають сутностям користувач і продукт беремо з певних відкритих джерел. Заповнені таблиці цих сутностей представлено в табл. 7 – 10.

Для прикладу змодельовано роботу генератора, при кількості продуктів 5 і кількості рецептів 3.

Табл. 7. Дані в таблиці Користувач

User_id	Full Name
1	Петренко Петро
2	Іваненко Іван
3	Тарасенко Тарас

Табл. 8. Дані в таблиці Продукти

Product_id	Product name
1	Картопля
2	Огірки
3	Помідори
4	Цибуля
5	Гриби

Табл. 9. Дані в таблиці Рецепти

Recipe_id	Recipe name
1	Овочевий салат
2	Картопля з грибами
3	Мариновані гриби

Табл. 10. Дані в таблиці РецептПродукти

Recipe_id	Product_id	Quantity
1	2	200
1	3	200
1	4	75
2	1	500
2	4	150
2	5	250
3	4	100
3	5	300

Дані про оцінки користувачів (в нашому прикладі – це дані про покупки) генеруються з наступними параметрами: кількість товарів в чеку – менше, дорівнює 2, кількість продукту в чеку – довільна величина від 100 до 800, округлена до 50.

Отже в ході роботи модуль згенерував наступні дані:

Табл. 11. Дані в таблиці Чек

Receipt_id	User_id	Date
1	1	10.03.2017
2	2	10.03.2017
3	3	11.03.2017
4	3	12.03.2017
5	2	14.03.2017

Табл. 12. Дані в таблиці ЧекПродукти

Recept_id	Product_id	Quantity	Price
1	1	600	12
1	5	300	15
2	5	300	15
3	2	500	8
3	3	500	25
4	1	400	8
4	5	250	12
5	2	300	5
5	4	100	3

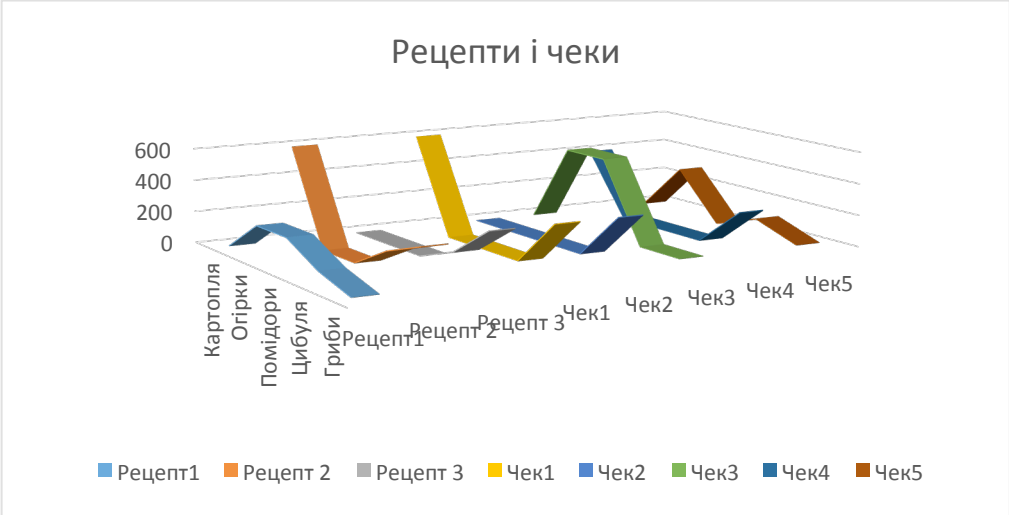


Рис. 1. Діаграма згенерованих рецептів і чеків

Після генерації тестових даних, необхідно безпосередньо проводити тестування рекомендаційної системи. Для надання рекомендацій продукту на основі певного чеку, необхідно знайти, які рецепти містять в собі продукти, що були куплені в чеку. При цьому необхідно також враховувати кількість придбаного продукту. Для надання рекомендації, необхідно знайти рецепт, для приготування якого не вистачає одного інгредієнту.

Отже для надання рекомендацій спочатку необхідно для кожного чеку j знайти рецепти i, для яких:

$$x_i - y_{ij} = 1 \quad (1)$$

, де x_i – кількість продуктів в рецепті i, а y_{ij} – кількість продуктів для рецепту i, в чеці j. Якщо, для певного j, не знайдено такого i, щоб задовольнити рівняння (1), то система не буде надавати рекомендації, для цього чеку. Якщо таких значень буде більше 1, то враховується ціна продукту.

Після того, як знайдено значення j та i, необхідно знайти, якого саме продукту не вистачає в чеці. Для кожної пари j та i, треба знайти єдиний p, такий що:

$$p \in Recipe_i \text{ та } p \notin Receipt_j \quad (2)$$

, де Recipe $_i$ – множина продуктів, в рецепті i, а Receipt $_j$ – множина продуктів в чеці j. Рекомендації, для кожного згенерованого чеку наведено в табл. 13.

Табл. 13. Рекомендації по чекам

Receipt_id	Продукти		Рекомендація	
	Product_id	Quantity	Product_id	Recipe_id
1	1	600	4	2
	5	300		
2	5	300	4	3
3	2	500	4	1
	3	500		
4	1	400		
	5	250		
5	2	300	3	1
	4	100		

Щоб переконатись, що система може рекомендувати різні дані і вони не залежать від генератора оцінок, змінимо дані, про продукти в рецептах (табл. 14.).

Табл. 14. Дані в таблиці РецептПродукти

Recipe_id	Product_id	Quantity
1	1	500
1	2	200
1	4	75
2	1	500
2	4	100
2	5	250
3	4	100
3	5	300

На рис. 2. – показано діаграму змінених даних.

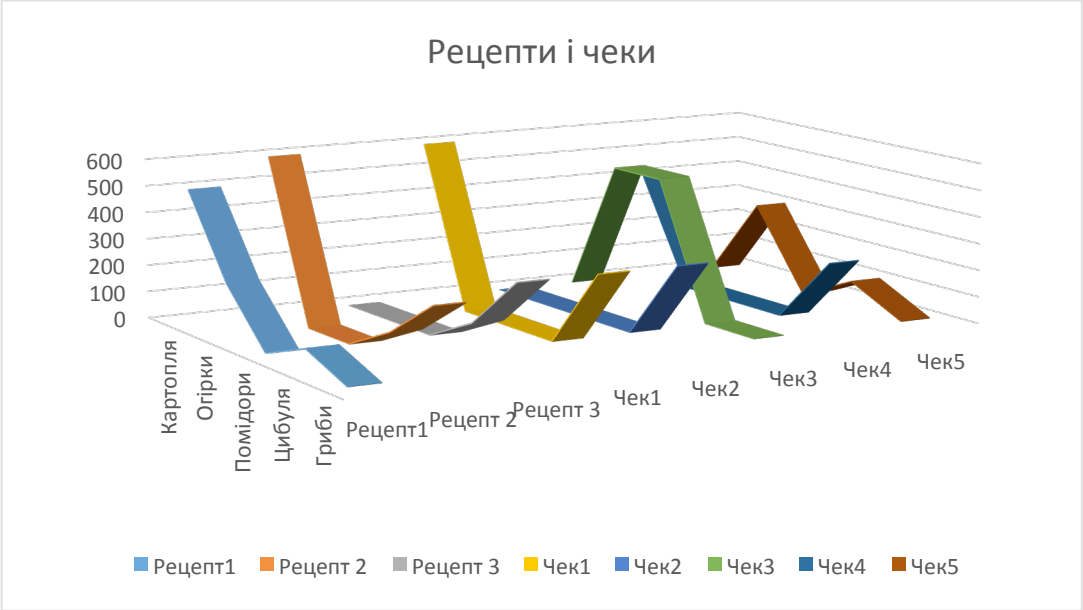


Рис. 2. Діаграма згенерованих рецептів і чеків

Після зміни даних, про рецепти, змінились також і рекомендації до чеків. Як видно з табл. 15. змінилися рекомендації до чеків з id 3 та 5. Для чеку 3, не знайдено рецептів, що задовольняють рівняння (1), а для чеку 5, змінився продукт рекомендації, що задовольняє умову (2).***Табл. 15. Рекомендації по чекам***

Receipt_id	**Продукти**		**Рекомендація**	
	Product_id	**Quantity**	**Product_id**	**Recipe_id**
1	1	600	4	2
	5	300		
2	5	300	4	3
3	2	500		
	3	500		
4	1	400		
	5	250		
5	2	300	1	1
	4	100		

6. Висновки

На прикладі рекомендаційної системи, для офлайн-продажів продуктів харчування, було продемонстровано роботу, модуля для генерації тестових даних для рекомендаційних систем. Цей модуль дозволяє проводити тестування рекомендаційних систем, не маючи даних від реальних користувачів.

Модуль генерує дані в залежності від способу розподілу ймовірностей, що задається, як параметр.

Для генерації даних різної структури, модуль приймає вид цієї структури у вигляді XML-файлу.

Отже цей модуль дозволяє розробляти і тестувати рекомендаційні системи, у відриві від предметної області її використання, що дозволить уніфікувати та пришвидшити процес розробки рекомендаційних систем, дозволить створювати рекомендаційні системи, аутсорсинговим компаніям, на пряму не пов’язаними з сферами застосування цих систем.

Список літератури

1. Королева Д.Е., Филиппов М.В. Анализ алгоритмов обучения коллаборативных рекомендательных систем. *Инженерный журнал: наука и инновации*, 2013, вып. 6.
2. А.Г. Гомзин, А.В. Коршунов. Системы рекомендаций: обзор современных подходов. *Труды Института системного программирования РАН, том 22, 2012, стр. 401-418.*

УДК 004.02

ТКАЧЕНКО Н.В.,
ЖДАНОВА О.Г.,
СПЕРКАЧ М.О.

ЗАСТОСУВАННЯ ПОШУКУ ІЗ ЗАБОРОНАМИ ДЛЯ РОЗВ'ЯЗАННЯ ЗАДАЧІ МІНІМІЗАЦІЇ СУМАРНОГО ЗАПІЗНЕННЯ РОЗКЛАДУ ВИКОНАННЯ ЗАВДАНЬ ЗІ СПІЛЬНИМ ДИРЕКТИВНИМ ТЕРМІНОМ ІДЕНТИЧНИМИ ПАРАЛЕЛЬНИМИ МАШИНАМИ

В різних областях людської діяльності виникає необхідність приймати рішення, пов'язані із розподіленням матеріальних та людських ресурсів. Такі рішення завжди спрямовані на досягнення певних цілей, що мають бути досягнуті в рамках певних обмежень. Непростою є задача обрання оптимального способу досягнення цілі серед інших, що потребують різних витрат ресурсів. Однією із найбільш розповсюджених задач у всіх сферах життя та виробництва є задача складання розкладів виконання завдань, або задача календарного планування. Переважна більшість таких здач відносяться до класу NP-складних. Тобто, не для кожної конкретної задачі можна розробити алгоритм, що вирішує її за прийнятний час. В цій роботі розглянуто метод, що дозволяє зменшити розмірність задачі мінімізації сумарного запізнення розкладу виконання завдань зі спільним директивним терміном ідентичними паралельними машинами. Також запропоновано алгоритм розв'язання, що базується на рандомізованому алгоритмі пошуку із заборонами.

Ключові слова: СКЛАДАННЯ РОЗКЛАДІВ, КАЛЕНДАРНЕ ПЛАНУВАННЯ, ТАБУ-ПОШУК, ПОШУК ІЗ ЗАБОРОНАМИ

In various areas of human activity it is necessary to make decisions related to material and human resources distribution. Such decisions are always aimed at achieving certain goals that should be achieved within certain limits. Is not easy task to chose the best goals achievement way from others required different resource costs. One of the most common problems in all areas of life and production is scheduling problem. The most of that problems are NP-complex. That is, no any of specific problems can be solved in a reasonable time. This paper describes the minimizing total tardiness for common due date identical parallel machine scheduling problem dimension reducing method. Also, there is an algorithm based on randomized tabu-search.

Keywords: SCHEDULING, SCHEDULING, TABU SEARCH, SEARCH FOR THE PROHIBITION

1. Вступ

Задачі теорії розкладів у наш час мають велике прикладе значення. Побудова розкладів є одним із ключових моментів функціонування підприємств, роботи сфери обслуговування, транспорту, освіти тощо. Тому актуальною є розробка алгоритмів складання календарних планів виконання завдань паралельними машинами, що допомагають зменшити сумарний штраф за порушення директивного терміну.

В цій роботі наводиться метод розв'язання поставленої задачі, заснований на алгоритмі табу-пошуку, або пошуку із заборонами. Табу-пошук є мета-евристичним алгоритмом, що здійснює локальний пошук і забороняє переміщення, що змушують повертатися до попередніх рішень.

2. Постановка задачі

Задано множину завдань J, число машин m, для кожного $j \in J$ відомо тривалість виконання t_j. Всі завдання мають спільний директивний термін D. Необхідно побудувати розклад σ виконання завдань $j \in J$ на m ідентичних машинах такий, щоб досягнути мінімуму функціоналу:

$$F(\sigma) = \sum_{j \in J} \max[0; C_j(\sigma) - D],$$

де $C_j(\sigma)$ – момент завершення виконання завдання j в послідовності σ.

Передбачається, що всі завдання множини J надходять на виконання одночасно і обслуговуються без переривань [1].

Наведена вище задача належить до класу NP-складних задач. Її можна вирішити за псевдополіноміальний час при $m = 2$.

Завдання множини $J = \{1,2,...,n\}$ перенумеруємо за неспаданням значень t_j та розіб'ємо на підмножини $J_1, J_2, ..., J_i, ..., J_m$, які попарно не перетинаються, де J_i – множина завдань, що обслуговуються i-ю машиною, $i = \overline{1,m}$.

Множину завдань J_i розбиваємо на підмножини $P_i(\sigma)$, $S_i(\sigma)$, $Q_i(\sigma)$ за наступними ознаками:

$P_i(\sigma)$ – множина завдань, що не запізнюються в розкладі машини i;

$S_i(\sigma)$ – множина завдань, що запізнюються в розкладі машини i, для яких виконуються умови: $S_i^H < D$, $C_j > D$, $\forall j \in S_i(\sigma)$, де S_i^H – момент початку виконання завдання j;

$Q_i(\sigma)$ – множина завдань, що запізнюються в розкладі машини i, для яких виконуються умови: $S_i^H > D$, $\forall j \in Q_i(\sigma)$,

$P = \bigcup_{i=\overline{1,m}} P_i$; $S = \bigcup_{i=\overline{1,m}} S_i$; $Q = \bigcup_{i=\overline{1,m}} Q_i$.

$R_i(\sigma) = D - \sum_{j \in P_i(\sigma)} t_j$ – резерв часу машини i в розкладі σ;

$\Delta_i(\sigma) = \sum_{j \in P_i(\sigma) \bigcup S_i(\sigma)} t_j - D$ – запізнення виконання завдання $j \in S_i(\sigma)$ відносно директивного терміну.

3. Достатні умови оптимальності

Введемо позначення $\mu = |P(\sigma) \cup S(\sigma)|$. Із теорем, що сформульовані та доведені в [1], [2], виведена достатня умова оптимальності розкладу: оптимальним є рівномірний розклад $\sigma \in \Psi_P \subseteq \Psi_{PS}$. До класу Ψ_{PS} належать такі розклади, у яких в $P(\sigma) \cup S(\sigma)$ входять перші μ найкоротших завдань; до класу $\Psi_P \subseteq \Psi_{PS}$ – розклади, у яких в $P(\sigma)$ входять найкоротші завдання. Детальний опис розкладів з класів Ψ_P, Ψ_{PS} наведено в [1].

Позначимо як $\overline{\mu}$ найбільше (і як $\underline{\mu}$ – найменше) значення μ, при якому $\sigma \in \Psi_{PS}$. Для оцінки $\underline{\mu}$ уявимо граничний випадок, коли для деякого розкладу σ при $\mu = \mu'$ виконується $\sum_{j=1}^{\mu'} t_j = m \cdot D$, тобто всі завдання із множини $P \cup S$ виконуються у директивний термін. Також очевидно, що при $\mu = \mu' + 1$ виконується $\sum_{j=1}^{\mu'+1} t_j > m \cdot D$, і при цьому $J_{\mu+1} \in Q$. Логічно припустити, що в такому випадку $\underline{\mu} = \mu'$. Розглянемо $\mu'' = \mu' - 1$: $\sum_{j=1}^{\mu''} t_j = \sum_{j=1}^{\mu'} t_j - t_\mu$;

$t_\mu > 0 \Rightarrow \sum_{j=1}^{\mu''} t_j < \sum_{j=1}^{\mu'} t_j \Rightarrow \sum_{j=1}^{\mu''} t_j < m \cdot D$. Отримана нерівність протирічить беззаперечній нерівності $\sum_{j=1}^{\mu} t_j \geq m \cdot D$. Отже, $\forall \mu$ виконується

$\sum_{j=1}^{\mu-1} t_j < m \cdot D \leq \sum_{j=1}^{\mu} t_j$.

Звідки отримуємо таку оцінку $\underline{\mu}$:

$$\underline{\mu} = \min\left\{\mu \mid \sum_{j=1}^{\mu-1} t_j < m \cdot D \leq \sum_{j=1}^{\mu} t_j\right\}.$$

Аналогічними міркуваннями знаходимо оцінку $\overline{\mu}$:

$$\overline{\mu} = \max\left\{\mu \mid \sum_{j=1}^{\mu-m} t_j < m \cdot D \leq \sum_{j=1}^{\mu} t_j\right\}.$$

Згідно з [2], оптимальним буде такий розклад $\sigma \in \Psi_P$, в якому завдання з множин $S(\sigma)$, $Q(\sigma)$ рівномірно розподілені між m машинами, тобто за умови кратності m кількості завдань, що належать множинам $S(\sigma)$

та $Q(\sigma)$. Позначимо $|S(\sigma)| = s$, $|P(\sigma)| = p$. Очевидно, що:
$|Q(\sigma)| = n - \mu; \mu = p + s; 0 \le s \le m$.

Розклад є рівномірним, якщо $(n - \mu + s) \bmod m = 0$. Під *ідеальним контуром* будемо розуміти такий частковий розклад σ^* виконання завдань з множини $P(\sigma)$, за якого можна гарантувати $\sigma \in \Psi_p$ та його рівномірність. Для кожного рівномірного розкладу справедливо: момент закінчення виконання завдань з множини $P(\sigma)$ на деяких $m - s$ пристроях буде збігатися із директивним терміном; для решти s пристроїв $|S_i(\sigma)| = 1$. Для часткового розкладу σ^* справедливо наступне: для $m - s$ пристроїв, у яких в повному розкладі $S_i(\sigma) = \varnothing$, в частковому розкладі $\Delta_i = R_i = 0$; для решти s пристроїв у частковому розкладі сумарний резерв $U = m \cdot D - \sum_{j=1}^{p} t_j$. Необхідно забезпечити виконання умови «найменша тривалість виконання роботи $j \in S(\sigma)$ є більшою ніж найбільший резерв машин» для того, щоб повний розклад σ належав класу Ψ_p. При виконанні цієї умови можна побудувати частковий розклад σ^*, що буде ідеальним контуром. В ньому i-та машина $(i = \overline{1,s})$ матиме резерв $1 \le R_i \le U - s + 1$, а також виконується $\sum_{i=1}^{s} R_i = U$. Підхід до побудови ідеального контуру детально наведено в [3]. Приклад ідеального контуру наведено на рисунку 1.

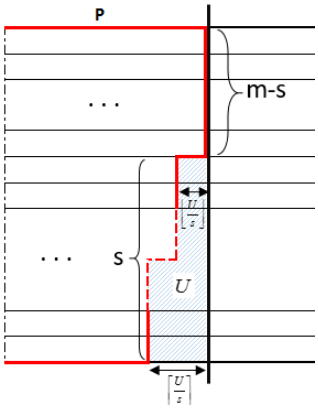


Рис. 1. Один із можливих варіантів ідеального контуру

Якщо під час розв'язання задачі буде знайдено такий частковий розклад σ^*, що співпадає з ідеальним контуром, то за побудовою повний розклад σ належатиме класу Ψ_p, буде рівномірним, а отже і оптимальним. Тобто, достатньою умовою оптимальності повного розкладу є існування такого часткового розкладу σ^*.

4. Застосування алгоритму пошуку із заборонами

Одним із методів, який можна застосувати при вирішенні задачі складання розладу виконання завдань ідентичними машинами, є табу-пошук, або пошук із заборонами. Табу-пошук є мета-евристичним алгоритмом, що здійснює локальний пошук, забороняючи такі переміщення, що змушують повертатися до попередніх рішень. Заборони вводяться задля запобігання попадання в пастку в передчасних локальних оптимумах.

Починається алгоритм табу-пошуку із вихідного рішення. В якості нового рішення обирається найкраще з околу рішень, що генерується під час кожної ітерації. Існує поняття «табу-список», в який в кінці кожної ітерації заносяться деякі атрибути попередніх рішень. До виконання будь-якого із критеріїв зупинки обирається нове найкраще допустиме рішення: з околу обирається таке допустиме краще рішення, щоб воно не мало жодного із заборонених атрибутів.

Однією із найбільш трудомістких частин алгоритму пошуку із заборонами є формування околу поточного розв'язку. До нього мають входити такі розв'язки (часткові розклади), які складають завдання з множини $P(\sigma) = \{1,2,...,|P(\sigma)|\}$), в які можна перейти з поточного за одну перестановку, а також такі, що не порушують умови для $\sigma \in \Psi_p$. Тому слід розглянути типи можливих перестановок:

- ***перестановками P–P*** назвемо зміни місцями таких двох робіт j_1 та j_2, що належать множинам $P_{i_1}(\sigma)$ та $P_{i_2}(\sigma)$ відповідно;
- ***перестановками P–S*** назвемо зміни місцями таких двох робіт j_1 та j_2, що належать множинам $P_{i_1}(\sigma)$ та $S_{i_2}(\sigma)$ відповідно;
- ***перестановками S–S*** назвемо зміни місцями таких двох робіт j_1 та j_2, що належать множинам $S_{i_1}(\sigma)$ та $S_{i_2}(\sigma)$ відповідно.

Для кожного з типів перестановок є умови допустимості отриманих нових розв'язків, які базуються на обмеженнях для класу розкладів $\sigma \in \Psi_P \subseteq \Psi_{PS}$.

Отже, до околу $N(x)$ декого поточного розв'язку x входять всі розв'язки (часткові розклади), отримані всіма можливими допустимими перестановками *P–P*, *P–S*, *S–S*. Для внесення випадковості при виборі напрямку спуску вводиться ненульова ймовірність q. Тобто, кожна із допустимих перестановок береться до розгляду з ймовірністю q.

Умови завершення

Сформулюємо умови, при досягненні хоча б однієї з яких завершується робота алгоритму:

- перевищення максимальної допустимої кількості ітерацій , що задається на початку роботи алгоритму $K_{\max}$;
- знайдено розв'язок (частковий розклад σ^* виконання завдань з множини $P(\sigma)$), який співпадає з ідеальним контуром – листом дерева ідеальних контурів [3].

Отже, на основі вищевикладеного можна побудувати адаптований алгоритм табу-пошуку для розв'язання задачі складання часткового розкладу (і наближення його до ідеального контуру).

Введемо наступні позначення:

σ_0 – початковий розклад;

σ^*_{opt} – найкращий відшуканий частковий розклад (рекордний розв'язок);

σ – поточний частковий розклад;

k – номер поточної ітерації;

$K_{\max}$ – максимальна кількість ітерацій;

q – ймовірність потрапляння нового розв'язку до околу;

m – кількість пристроїв;

n – кількість завдань;

J – множина всіх завдань;

t_j – тривалість виконання завдання $j \in J$.

Позначимо як $TabuList_l(\sigma)$ – *табу-список* – список розв'язків (розкладів), перехід до яких заборонений. Список має довжину l – максимальна кількість розв'язків, що зберігаються в один момент у списку.

В якості цільової функції оберемо мінімум функціоналу:

$$F(\sigma) = \sum_{j \in J} \max[0; C_j(\sigma) - D],$$

де $C_j(\sigma)$ – момент завершення виконання завдання j в розкладі σ.

Адаптований алгоритм табу-пошуку

КРОК 0 Задати значення $K_{\max}$ та l.

КРОК 1 Побудувати початковий розклад σ_0 за жадібним алгоритмом.

КРОК 2 Обчислити:

$$F(\sigma_0) = \sum_{j \in J} \max[0; C_j(\sigma_0) - D]$$

Визначити початкові дані:

$k = 0$, $F^* = F(\sigma_0)$, $\sigma^*_{opt} = \sigma_0$, $TabuList_l(\sigma_k) = \varnothing$

КРОК 3 Обчислити значення p, s, U

КРОК 4 *Перевірити виконання умов завершення*

4.1 ЯКЩО $k = K_{\max}$

ТО перейти на **КРОК 11** (σ^*_{opt} – кінцевий частковий розклад)

ІНАКШЕ перейти на **КРОК 5**

4.2 ЯКЩО виконується достатня умова оптимальності (σ_k збігається з одним із листів дерева ідеальних контурів)

ТО перейти на **КРОК 11** (σ^*_{opt} – кінцевий частковий розклад)

ІНАКШЕ перейти на **КРОК 5**

КРОК 5 Для поточного розкладу σ_k сформувати окіл $N_q(\sigma_k)$, що з ймовірністю q містить кожен з можливих розкладів, що можуть бути отримані перестановками типу P–P, P–S, S–S, та не належать $TabuList_l(\sigma_k)$.

КРОК 6 ЯКЩО $N(\sigma_k) = \varnothing$

ТО перейти на **КРОК 11** (σ^*_{opt} – кінцевий частковий розклад).

КРОК 7 Знайти σ_{k+1} таке, що $F(\sigma_{k+1}) = \min\limits_{\upsilon \in N_q(x)} \{F(\upsilon)\}$.

КРОК 8 ЯКЩО $F^* > F(\sigma_{k+1})$

ТО обновити найкращий поточний розв'язок: $F^* = F(\sigma_{k+1})$, $\sigma^*_{opt} = \sigma_{k+1}$.

КРОК 9 Додати σ_k до списку заборон: $TabuList_l(\sigma_{k+1}) = TabuList_l(\sigma_k) \cup \{\sigma_k\}$.

КРОК 10 Збільшити лічильник ітерацій $k := k + 1$.

Перейти на **КРОК 4.**

КРОК 11 На основі отриманого оптимального часткового розкладу (ідеального контуру) побудувати повний розклад.

Завдання з множини $S(\sigma)$ розташувати таким чином, щоб виконувалась умова $S^H_{j_k} \le S^H_{j_l}$, якщо $t_{j_k} \le t_{j_l}$, $\forall j_k,\ j_l \in S(\sigma)$.

Завдання з множини $Q(\sigma)$ розташувати таким чином, щоб виконувалась умова: Q_i містить ті і тільки ті елементи, котрі відрізняються від $|P \cup S| + i$ на величину, кратну m, $i = \overline{1,m}$.

КІНЕЦЬ АЛГОРИТМУ

Актуальним є питання роботи з табу-списком, в якому постійно здійснюється пошук з порівнянням і зберігається великий масив даних. Одним із варіантів оптимізації роботи є представлення списку у вигляді хеш-таблиці.

5. Висновки

В рамках цієї роботи був сформований метод розв'язання задачі мінімізації сумарного запізнення розкладів виконання завдань зі спільним директивним терміном ідентичними паралельними машинами. Сформульовані достатні умови оптимальності дозволяють зменшити розмірність задачі за рахунок оперування лише частиною завдань (складання часткового розкладу). Застосування модифікованого рандомізованого алгоритму пошуку із заборонами дозволяє отримувати більш точний розв'язок, а також дозволяє алгоритму поступово виходити з локальних оптимумів та наближатись до глобального або близького до нього. Одним із способів збільшення швидкодії алгоритму є оптимізація роботи з табу-списком, наприклад, представленням списку у вигляді хеш-таблиці.

Список літератури

1. Основы системного анализа и проектирования АСУ: Учебн. пособие. / А.А. Павлов, С.Н. Гриша, В.Н.Томашевский и др. Под общ. ред. А.А.Павлова, К.: Выща шк.,– 1991.– 367 с.

2. Танаев В. С. Введение в теорию расписаний / В. С. Танаев, В. В. Шкурба. – М.: Наука, 1975. – с. 256.

3. Ткаченко В.Ю., Ткаченко Н.В., Жданова О.Г., Сперкач М.О. Задача мінімізації сумарного запізнення розкладу виконання завдань зі спільним директивним терміном паралельними пристроями//Системний аналіз та інформаційні технології: матеріали 19-ї Міжнародної науково-технічної конференції SAIT 2017, Київ, 22-25 травня 2017 р. – К.: НТК "ІПСА" НТУУ "КПІ", 2017. – С. 126 — 127.

УДК 044.724.4(045)

КУЛАКОВ Ю.А.,
КОГАН А.В.,
ХРАПОВ В.В.

СПОСОБ КОНСТРУИРОВАНИЯ ТРАФИКА ПРИ ОРГАНИЗАЦИИ МНОГОПУТЕВОЙ МАРШРУТИЗАЦИИ

В работе предложен способ конструирования трафика на основе многопутевой маршрутизации. Основным преимуществом конструирования трафика на основе многопутевой маршрутизации является то, что не нужно каждый раз пересчитывать все пути, достаточно выбирать пути уже из множества сформированных, а осуществлять реконфигурацию только для сформированных путей.

Применение теории игр и принципа самоподобия позволило достичь необходимого уровня качества обслуживания QoS и равномерно загрузить все пути.

A method for constructing traffic based on multipath routing is proposed. The main advantage of designing traffic based on multipath routing is that you do not need to recalculate all the paths every time, it is enough to select paths from a lot of generated ones, and to perform reconfiguration only for the generated paths.

The application of game theory and the principle of self-similarity allowed to achieve the required level of QoS service quality and to evenly load all the paths.

1. Введение

Современной тенденцией развития компьютерных сетей является переход от специализированных сетей, каждая из которых предназначена для выполнения узкого круга услуг, к мультисервисным сетям. Неотъемлемой частью системы управления такой сети должна быть надежная система управления трафиком.

Появление новых технологий позволило Интернету переносить трафик, предлагаемый приложениями реального времени с более высокой пропускной способностью и минимальными требованиями к задержке, такими как потоковое видео и передача голоса по IP (VoIP). Проблема маршрутизации этого трафика в сети с целью нахождения оптимальной конфигурации сети, минимизации задержки, потери пакетов и оптимизации использования полосы пропускания может быть решена путем применения многопутевой маршрутизации за счет эффективной идентификации пути и распределения трафика.

Для организации безопасной системы конструирования трафика необходимо выполнить следующее условия:

1. Формирования пути с определенными желаемыми свойствами, такими как минимальная задержка, максимальная ширина полосы и т. д.

2. Распределение трафика оптимально разделяет поток между несколькими путями, чтобы достичь балансировки нагрузки.

Вопросы управления трафиком в современных мультисервисных сетях рассматриваются в работах [1, 2, 3,]. Преимущество распределенных алгоритмов в том, что они настраиваются в одном временном масштабе, и способны быстро реагировать на изменения трафика [4].

Несмотря на это, остается ряд нерешенных задач:

- фактически отсутствует строгая теоретическая база, которая пришла бы на смену классической теории массового обслуживания при проектировании современных систем распределения информации с учетом мультимедийного трафика;
- нет единой модели пульсирующего трафика;
- не существует достоверной и признанной методики расчета параметров и показателей качества систем распределения информации с учетом мультимедийного трафика;

- отсутствуют алгоритмы и механизмы, обслуживания в условиях

2. Анализ существующих способов. Постановка задачи

Одним из эффективным способов повышения уровня качества обслуживания трафика является формирования и выбор наиболее подходящего пути для передачи данных по сети. Существующие протоколы маршрутизации, такие как RIP, OSPF, IGRP/EIGRP и IS–IS не учитывают такие параметры качества обслуживания (QoS) как задержка, вероятность потери пакетов и надежность, а ориентируются лишь на пропускную способность каналов или количество промежуточных маршрутизаторов. Это приводит к появлению задержек доставки пакетов и даже их потерь.

В работе [5] предложен метод управления трафиком в QoS-ориентированном протоколе маршрутизации. Особенностью данного метода является относительная простота его реализации, а также небольшой объем передаваемой служебной информации и невысокая вычислительная сложность функционирования. Теоретическим недостатком, как следствием простоты, является меньшая эффективность, в сравнении с адаптивными динамическими алгоритмами балансировки нагрузки.

Процесс передачи данных в сети часто рассматривается в соответствии с Пуассоновским процессом. Огромные потоки данных трафика позволяют проследить долговременную зависимость, то есть процесс самоподобия. Сейчас можно утверждать, что самоподобие является существенным фактором сетевого моделирования [6]. Самоподобные свойства трафика позволяют с достаточной степенью достоверности прогнозировать появление на сегменте сети временных периодов с перегрузкой по производительности оборудования и линий связи, что в свою очередь, делает возможным построение обеспечивающие качество пульсирующего трафика [4]. системы с динамическим управлением возможной пропускной способности для отдельных видов трафика.

Еще один перспективный способ конструирования трафика базируется на использовании технологии MPLS, что позволяет распределить трафик по нескольким каналам одного пути, и сбалансировать нагрузку по всем путям для достижения оптимальной загрузки каналов и повышения производительности сети. Используя эти возможности сетей MPLS рассмотрим стратегию разделения трафика с помощью теории игр.

3. Решение поставленной задачи

Способ формирования множества непересекающихся путей предложен в работе [7], позволяет сформировать все множество непересекающихся маршрутов между двумя узлами с требуемым уровнем QoS. Для конструирования трафика, используя сформированные пути, предлагается использовать теорию игр [8, 9, 10].

Данный способ эффективно применяется для решения задач системного анализа и матричных игр. Использования теории игр для управления трафиком, даст возможность определить и оценить каждое из возможных решений с различных точек зрения, и принять решения с учетом минимизации риска и необходимого QoS.

Рассмотрим граф сети G(A, E), где A – количество вершин, E – количество связей между вершинами рис.1. Для организации безопасной маршрутизации в сети, осуществлять передачу информации необходимо по множеству непересекающихся путей. Сформировать все множество путей между двумя вершинами [11] даст возможность выбрать лучший вариант для передачи определенного типа трафика.

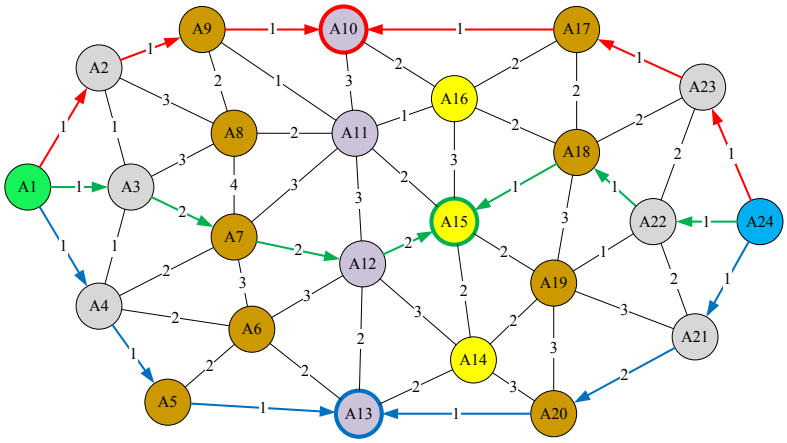


Рис. 1. Пример графа сети G(A, E)

1) Найденные пути:

P_1 = {A1-A2-A9-A10-A17-A23-A24}, суммарный вес – 6;

P_2 = {A1-A3-A7-A12-A15-A18-A22-A24}, суммарный вес – 10;

P_3 = {A1-A4-A5-A13-A20-A21-A24}, суммарный вес – 7.

2) Надёжность узлов в найденных маршрутах:

A1=0.99	A9 = 0.90	A18=0.89
A2=0.98	A10=0.93	A20=0.89
A3=0.96	A12=0.92	A21=0.97
A4=0.98	A13=0.94	A22=0.99
A5=0.95	A15=0.95	A23=0.98
A7=0.92	A17=0.91	A24=0.99

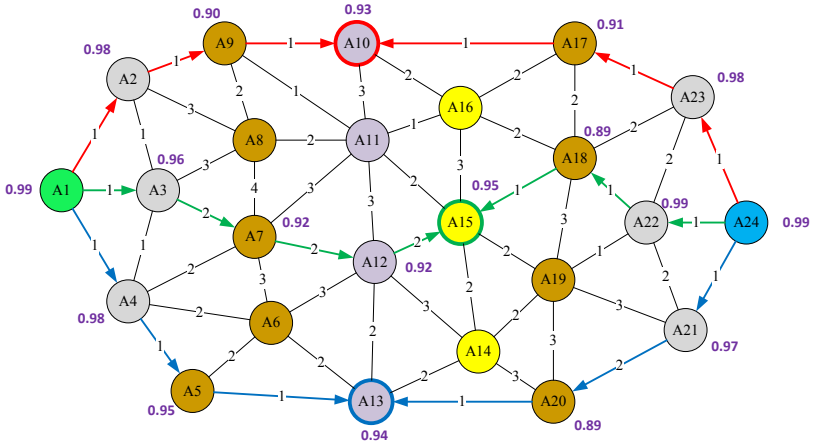


Рис. 2. Пример графа сети G(A, E) с надёжностью узлов

Пусть для каждого *i*-го узла дана надежность узла p_i. Для вычисления вероятности потери пакета $P(A_l)$ для путей с множеством $P_l = (P_1, P_2, ..., P_l)$ будем применять формулу произведения вероятностей:

$$P(A_l) = 1 - \prod_{i=1}^{n} P(p_i), i = \overline{1,n}, \qquad (1)$$

где A_l – вероятность потери пакета при передаче по *l*-му маршруту, p_i - надежность *i*-го узла.

3) Расчёт вероятности потери пакета для каждого найденного маршрута:

$p_1 = 1 – (0.99 • 0.98 • 0.90 • 0.93 • 0.91 • 0.98 • 0.99) = 1 – 0.717 = 0.283$

$p_2 = 1 – (0.99 • 0.96 • 0.92 • 0.92 • 0.95 • 0.89 • 0.99 • 0.99) = 1 – 0.666 = 0.333$

$p_3 = 1 – (0.99 • 0.98 • 0.95 • 0.94 • 0.89 • 0.97 • 0.99) = 1 – 0.74 = 0.26$

4) Итоговая вероятность потери пакета при передаче по маршрутам:

$p_1 = 28\%$

$p_2 = 33\%$

$p_3 = 26\%$

Каждый из непересекающихся путей, который сформирован, имеет ряд характеристик, а именно: длинна пути, вероятность выхода из строя узла, пропускная способность и вероятность потери информации при передаче. Зная эти особенности, игроки смогут выбрать наилучший вариант решения задачи.

Представим игру в виде

$\Gamma = < N, X_1, X_2 \dots X_n, K_1(X_1 \dots X_n), \dots, K_n(X_1 \dots X_n) >,$

где N – множество игроков а именно данные которые передаются в сети, N={1, 2, 3…n}.
X – количество стратегий – количество путей с определенным QoS,
X_1 – количество стратегий 1-го игрока,
X_n - количество стратегий n-го игрока.
$K_1(X_1 \dots X_n)$ – функция выиграша *i*-го иргока.

Игроки независимо один от другого выбирают стратегию $x_i \epsilon X_i$ и соответственно каждый получает наилучший результат решения. $X=(X_1 \dots X_n)$ – ситуация игры. Учитывая эти параметры пути, можем определить правила игры. В зависимости от входного трафика определяется путь который соответствует заданным требованиям. Для этого предлагается проводить “маркировку” входного трафика и в зависимости от приоритета выбирается стратегия игры каждого игрока.

4. Равномерная загрузка путей

Основным преимуществом конструирования трафика на основе многопутевой маршрутизации заключается в том, что не нужно каждый раз пересчитывать все пути, достаточно выбирать пути уже из множества сформированных, а осуществлять реконфигурацию только для сформированных путей.

Распределение нагрузки по множеству сформированных путей передачи данных между двумя узлами является одной из важных задач. Во многих ситуациях при передачи данных по пути, суммарный трафик который передается, может в несколько раз превышать максимально допустимую пропускную способность пути. В этом случае, единственным возможным решением может быть деление сообщения на части [12] и передачу по нескольким путям. При высокой скорости движения информации по сети, пакеты поступают на узел не по отдельности а целиком. Трафик в таких сетях имеет явно выраженный всплесковый характер, что повышает вероятность перегрузок в узлах сети.

Однако, в компьютерных сетях, число и характер событий на определенном временном интервале зависит от прежних, весьма отдаленных событий. Это означает, что при больших масштабах сети трафик обладает свойством самоподобия [13], т.е. выглядит практически одинаково при любых достаточно больших масштабах временной оси.

Рассмотрим необходимые меры предотвращения возникновения перегрузки на пути.

1. Необходимо изменить маршрут передачи данных для обхода проблемного узла. Например, схема живучести [14] может уже существовать и быть реализована за счет альтернативного механизма в виде запасного пути для передачи [15], который гарантирует непрерывность обслуживания в случае сбоя.
2. Перенаправить весь поток на другой менее безопасный путь. В случае перегрузки основного пути, можно весь трафик перенаправить на альтернативный менее безопасный маршрут, но безопасность информации также сохраняется. Если такового пути нет, тогда маршрут не изменяется.
3. Распределить трафик с основного пути на все не загруженные пути независимо от уровня их безопасности.

Предложена процедура восстановления, должна быть запущена для пути, при возникновении отказа или перегруза рис. 3.

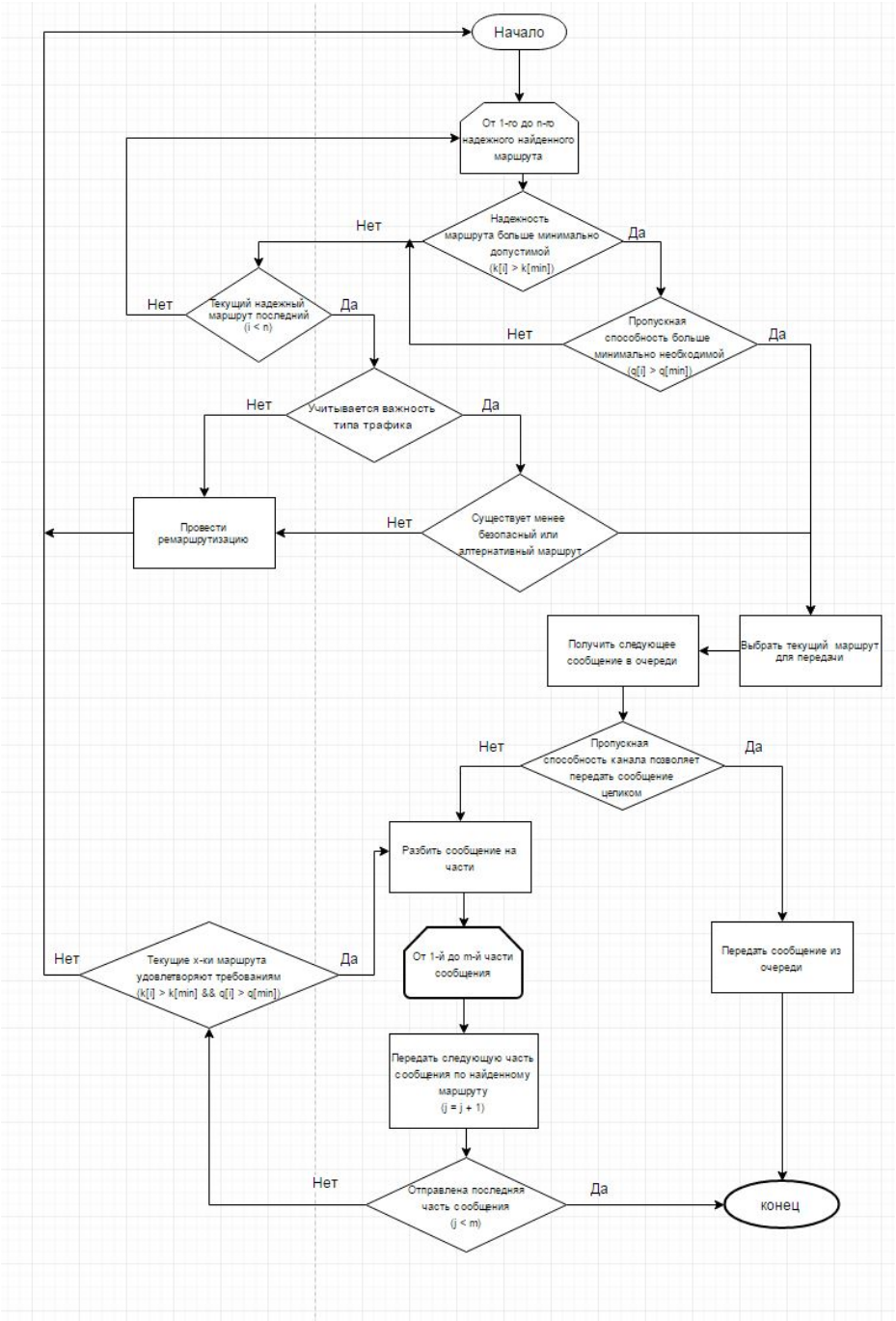


Рис. 3. Алгоритм конструирования трафика при организации многопутевой маршрутизации

Выводы

Многопутевая маршрутизация может быть использована в восстановлении IP для ускорения маршрутизации путем переопределения потоков и перенаправления трафика по сокращенному набору путей. Это поможет избежать дополнительного формирования путей и резервирования ресурсов, обычно применяемых при восстановлении MPLS, для достижения более быстрого восстановления. Достижение быстрого изменения маршрута может стать более важным в условиях ремаршрутизации, когда “поврежденный” путь - тот, у которого самый высокий потенциал для передачи, и процесс восстановления может потребовать, чтобы сумма запасных мощностей оставшихся путей была строго больше, чем ширина полосы, где существует “поврежденный” путь. Предложенная схема конструирования трафика с несколькими путями могут обеспечить решение этой проблемы.

Список литературы

1. Dahai Xu, Mung Chiang, Jennifer Rexford, "Link- State Routing With Hop-by-Hop Forwarding Can Achieve Optimal Traffic Engineering", IEEE/ACM TRANSACTIONS ON NETWORKING, Vol. 19, no. 6, December 2011.
2. Ditixa Vyas, Ritesh Patel, Amit Ganatra. Survey of Distributed Multipath Routing Protocols for Traffic Management / Ditixa Vyas, Ritesh Patel, Amit Ganatra // International Journal of Computer Applications (0975 – 8887) Volume 63– No.17, February 2013. – P. 42-48.
3. R. Banner and A. Orda. Multipath Routing Algorithms for Congestion minimization. In Proc. IFIP Networking, 2005.
4. Ke Xu, Hongying Liu, Jiangchuan Liu, Jixiu Zhang, " LBMP: A Logarithm- Barrier-Based Multipath Protocol for Internet Traffic Management", IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 22, NO. 3, MARCH 2011.
5. Симаков Д. В. Управление трафиком в сети с высокой динамикой метрик сетевых маршрутов / Симаков Д. В. // Интернет-журнал «НАУКОВЕДЕНИЕ». – 2016. – Том 8, No1. – С. 1-13. Реж. дост. http://naukovedenie.ru
6. Тарадаев С.А. АНАЛИЗ СВОЙСТВ САМОПОДОБИЯ ТРАФИКА В СЕТИ ASTERISK / С.А. Тарадаев, К.А. Бохан // Системи обробки інформації. – 2012. – Випуск 2 (100). – С. 222-227.
7. Кулаков Ю. А. Формирование множества непересекающихся путей в компьютерных сетях с применением алгоритма «обратной волны» / Кулаков Ю. А., Коган А. В., Диброва М. А., Чхаидзе Д. М. // Проблеми інформатизації та управління: збірник наукових праць. – К.:НАУ, 2015. – Вип. 4 (52). – С.68-73.
8. Федорова М. Л. ОБ ИССЛЕДОВАНИИ СВОЙСТВА САМОПОДОБИЯ ТРАФИКА МУЛЬТИСЕРВИСНОЙ СЕТИ / М. Л. Федорова, Т. М. Леденева // ВЕСТНИК ВГУ, СЕРИЯ: СИСТЕМНЫЙ АНАЛИЗ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ, 2010. – No 1. – С. 46-54.
9. Додонов А.Г. Живучесть информационных / А. Г. Додонов, Д. В. Ландэ // – К.: Наук. думка, 2011. – 255 с.
10. Кулаков Ю.А. Способ и средства конструирования трафика на основе безопасной многопутевой маршрутизации в мобильных сетях. / Ю.А. Кулаков, В.В. Лукашенко, А.В. Коган // Transactions of Azerbaijan national academy of sciences. Informatics and control problems. – 2014. – №3, Vol. XXXIV. – Pp.62-68.
11. Кулаков Ю. А. Формирование множества непересекающихся путей между граничными маршрутизаторами сети MPLS / Кулаков Ю. А., Диброва М. А., Коган А. В. // Electronics and Communications. Електроніка та зв'язок. Електроніка та зв'язок. – 2016. – Том 21. – №1(90), – С.50-55.
12. Кулаков Ю.О. Алгоритм поділу і збірки секретного повідомлення для багатошляхової маршрутизації в бездротових мережах / Кулаков Ю.О., Коган А.В., Пирогов А.А. // Вісник НТУУ «КПІ». Інформатика, управління та Обчислювальна техніка: збірник наукових праць. – К.: Століття +, 2012. – №57. – С.46-50.
13. Mohammad Naserian, Kemal Tepe. Game theoretic approach in routing protocol for wireless ad hoc networks / Mohammad Naserian, Kemal Tepe // Ad Hoc Networks. – 2009. - № 7. – pp. 569–578
14. M. Afergan, “Using repeated games to design incentive-based routing systems,” in Proc. of the IEEE INFOCOM, 2006.
15. Кулаков Ю.О. Спосіб мінімізації часу обходу скомпрометованих вузлів в мобільних мережах / Кулаков Ю.О., Коган А.В. // Electronics and Communications. Електроніка та зв'язок. Електроніка та зв'язок. – 2014. – Том 19. – №2(79), – С.116-122.

УДК 004.492.3

ГЕГА А.С.
СЕЛІН Ю.М.

СПОСОБИ МОНІТОРИНГУ АКТИВНОСТІ ПРОЦЕСІВ В СИСТЕМІ LINUX

В даній статті розглянуто важливість створення ПЗ для моніторингу активності процесів в Linux системах. Надано визначення поняття процесу. Розглянуто основні способи моніторингу активності процесів, приведений їх опис, розглянуті переваги та недоліку кожного методу.

In this article described the importance of creating software to monitor activity processes in Linux systems. Provided the definition process. Described main methods of monitoring processes activity, given their description, considered the advantages and disadvantages of each method.

1.Вступ

Вважається, що Unix-подібні операційні системи краще захищені від комп'ютерних вірусів ніж системи Windows. До сьогоднішнього дня немає жодного широко розповсюдженого віруса для Linux, в той час як в оточення Windows кишить різноманітними шкідливими додатками.

Проте, не можна вважати перехід на *nix платформу гарантією захисту від шкідливого ПЗ. Платформи *nix набирають популярність, велика кількість шрафічних оболонок нічим не поступаються Windows, а наявність вбудованого пакетного менеджера роблять поріг входу навіть нижчим ніж у Windows. Відсутність системного реєстру дозволяє системі через тривалий час працювати так само швидко як і в перший день після втановлення. І не варто забувати що більшість рішень – open source, тобто безкоштовні.

Разом з популярністю *nix систем і зниженням порогу входу росте загроза вірусного цунамі, що захлисне недосвідчених користувачів. Уже зараз безпека *nix систем на грані визнання її міфом. В більшті популярних дистрибутивів програми поставляються у вигляді бінарних файлів, що дає простір для проведення атак. Звісно, скачаний бінарний файл за замовчуванням не є виконуваним, більше того, для роботи певних функцій можуть бути потрібні права адміністратора або вище, проте отримати їх не складно, а недосвідчений користувач не завжди спроможний побачити загрозу, тому може легко надати потрібні права шкідливому ПЗ.

Першим кроком на шляху вирішення даної проблеми є моніторинг активності процесів. Статистика запусків додатків допоможе виявити шкідливе ПЗ та його патерни поведінки, а при наявності даної інформації запобігти його запуску чи подільшій робті.

2. Поняття процесу

Процеси - це одна з найстаріших і найбільш важливих абстракцій, притаманних операційній системі. Вони підтримують можливість здійснення (псевдо) папаралельних операцій навіть при наявності всього одного центрального процесора.

Процес - це просто екземпляр виконуваної програми, включаючи поточні значення лічильника команд, регістрів і змінних. Концептуально у кожного процесу є свій, віртуальний, центральний процесор. Зрозуміло, насправді справжній центральний процесор постійно перемикається між процесами, але щоб зрозуміти систему, куди простіше думати про набір процесів, запущенних в (псевдо) паралельному режимі, ніж намагатися відслідковувати, як центральний процесор перемикається між програмами. Це постійне перемикання між процесами називається мультипрограмуванням або багатозадачним режимом роботи.

Одразу варто пояснити разницю між процесом і програмою. Програма це послідовний набір дій виконавши який буде

певний результат. Процес — це уже дія. В нього є програма, вхідні та вихідні дані, набір станів.

Варто відзначити, що якщо програма запущена двічі, то вважається, що нею зайняті два процеси. Наприклад, часто можна двічі запустити текстовий процесор або одночасно роздрукувати два файли, якщо одночасно доступні два принтера. Той факт, що два працюючих процеси запущені від однієї і тієї ж програми, до уваги не береться, оскільки це два різних процеси. Операційна система може дозволити їм використовувати загальний код, тому в пам'яті буде присутній тільки одна копія цього коду, але це чисто технічна деталь, що не міняє концептуальну ситуацію, що стосується двох працюючих процесів.

3.Файлова система /proc

Перший спосіб спосіб моніторингу процесів – моніторинг файлвої системи /proc. Файлова система /proc - це спеціальна файлова система, присутня в багатьох сучасних UNIX-системах і містить масу корисної інформації в текстовому вигляді, хоча і не завжди зрозумілою пересічним користувачам. Важливо пам'ятати, що ця ФС не є фізичною і файли, розташовані на ній, не зовсім файли в традиційному розумінні.

Система /proc контролюється ядром. Через те, що вона надає інформацію, що контролюється ядром, логічно, що вона розташовується в пам'яті, що також контролюється ядром. Команда "ls -l" покаже, що більшість файлів в цій системі мають нульову довжину, але подивившись будь-який файл Ви отримаєте достатньо інформації. Це працює тому що файлова система /proc як будь-яка інша файлова система реєструється на рівні VFS (Virtual File System layer). Тому при запиті файлів/каталогів, файлова система /proc створює ці файли/каталоги на підставі інформації, що міститься в ядрі.

Деякі важливі файли:

- /proc/cpuinfo — інформація про процесор
- /proc/meminfo — інформація про RAM, swap і т.д.
- /proc/mounts — список примонтованих файлових систем
- /proc/devices — список пристроїв
- /proc/filesystems — список підтримуваних файлових систем
- /proc/modules — список завантажуваних модулів
- /proc/version — версія ядра
- /proc/cmdline — список параметрів ядра переданих при запуску

Файлова система /proc служить джерелом інформації про виконуються процесах. Всередині каталогу /proc знаходяться каталоги, назви яких складаються з цифр - це і є інформація про процеси - назва каталогу відображає ідентифікатор процесу (PID). Коден такий каталог містить файли і інші підкаталоги з різноманітної інформацією про процес — від часу та команди запуску до кількості дочірніх процесів та пріоритетів планування.

Сама файлова система /proc не здійснює моніторинг процесів. Проте її можна використати для цього. Як приклад, можна створитидодаток, що при запуску буде зчитувати з /proc інформацію про процеси і оновлювати її через деякий час. Відсутність pid в списку буде означати завершення роботи відповідного процесу, поява нового pid — створення нового процесу.

Плюсами даного методу є його простота. Робота з валами /proc не відрізняється від роботи з іншими файлами, тому задача додатку зводиться до зчитування потрібного розділу через певний проміжок часу та не складна обробка отриманих даних.

Мінусами даного методу є його невисока точність та неефективність. Щоразу потрібно зчитувати всю інформацію для аналізу, незалежно від того відбулись якісь зміни. Неточність полягає у тому, що у випадку, коли час життя процесу коротший від інтервалу оновлення даних, інформація про цей процес не буде записана.

4. Реєстрація подій на рівні ядра

Іншим способом моніторингу активності процесів є відслідковування основних етапів роботи процесу на рівні ядра.

В операційній системі Unix створення процесів відбувається унікальним чином. У більшості операційних систем використовується механізм породження

процесів (Spawn mechanism), який створює для процесу новий адресний простір, завантажує в нього код для виконання і передає йому управління. В системі Linux застосовується досить незвичайний підхід, при якому зазначені вище операції розділені на дві самостійні функції: fork() і exec(). Перша функція - fork() - створює породжений процес, який є копією поточного. Він відрізняється від батьківського процесу тільки значенням ідентифікатора PID (який повинен бути унікальним в системі), значенням параметра PPID (ідентифікатор PID батьківського процесу, для якого встановлюється значення PID початкового процесу), деякими ресурсами, такими як сигнали, які очікують обробки (що не успадковуються), а також статистикою використання ресурсів. Друга функція - exec() - завантажує виконуваний файл в адресний простір процесу і передає йому управління. Комбінація функцій fork () і exec () аналогічна тій одній функції створення пропроцесу, яка реалізована в більшості інших операційних систем.

В операційній системі Linux функція fork () реалізована через виклик системної функції clone(). Їй передається в якості аргументів набір параметрів, що визначають, які ресурси повинні бути загальними (якщо потрібно) у батьківського і дочірнього процесів. У всіх бібліотечних функціях fork(), vfork() і __clone() викликається системна функція clone(), якій передається набір відповідних параметрів. У самій же функції clone() викликається функція do_fork().

Як правило, знищення процесу ініціюється самим процесом. Це відбувається коли в самому процесі викликається системна функція exit(). Причому це може статися як явно, коли вся робота програми зроблена і потрібно завершити її роботу, так і неявно, при виконанні повернення з основної процедури будь-якої програми з ім'ям main(). Іншими словами, компілятор мови С поміщає виклик функції exit() в код, який виконується після повернення з процедури main(). Процес також може бути завершений ненавмисно. Так відбувається, коли процес отримує сигнал або виникає виняткова ситуація, яку той не може обробити або проігнорувати. Незалежно від того, яким чином завершується процес, основну масу роботи виконує функція do_exit().

Отже, незалежно від типу процесу, під час його створення буде викликано функцію do_fork(), при завершенні ж — do_exit(). Відстедення виклику цих 2 функцій, а також їх аргументів та повернених знацень, дасть можливість точно встановити час запуску і смерті процеса, а також отримати усю інформацію, поьтрібну для створення, наприклад ім'я і шлях до виконуваного бінарного файлу, аргументи запуску, налаштування оточення і т.д.

Пепевагами даного методу є його ефективність і точність. Інформація буде записана лише раз під час певної події. Тобто, навіть якщо буде якась затримка в відправленні інформації про події, сама ця інформація міститиме коректний час.

Недоліками ж методу є складність реалізації і можливе сповільнення роботи системи. Функції виконуються на рівні ядра, вся інформація знаходиться в ядрі. Тому для реалізації подібного моніторингу потрібно модифікувати код ядра або писати модуль ядра. Сама робота в ядрі відрізняється від написання програм рівня користувача. Також проблемою є передача інформація між додатком рівня користувача та модулем ядра. Функції do_fork і do_exit є ключовими функціями системи, дістати інформацію про виконнання яких можна лише доданням власних інструкцій з потрібними нам діями, що само по собі може спричинити проблеми. Та навіть у випадку коректного додання існує можливість сповільнення роботи системи.

5.Висновки

Операційна система Linux набирає популярність, разом з цим збільшуєтсья увага до неї зі сторони розробників шкідливого ПЗ. Хоча Linux захищена краще Windows систем, цей захист далеко не абсолютний і це потрібно усвідомити. Уже тривалий час Linux дистрибутиви і супутні програми розповсюджуються у вигляді бінарних файлів. З однієї сторони, це спрощує адаптацію нових користувачів. З іншої — відкриває додаткові можливості для проведення атак на систему. На даний

момент не існує ПО яке б могло б захистити систему від подібного по або хоча б

Файлова система /proc надає повний спектр інформації рпо процес, уніфікований та зручний достув до цієї інформації. На жаль, через описані вище проблеми, /proc не може забезпечити потрібну точність моніторингу. З іншого боку, вона ідеально підходить для отримання інформації про систему в поточний момент часу.

Моніторинг виконання функцій ядра є сенс використовувати при потребі точно зафіксувати статистику для проведення подільшого аналізу.
відобразити статистику запусків процесів, використання цього методу надасть інформацію про 100% процесів. Проте, користуватися цим механізмом потрібно акуратно. В важливі системні функції буде додано власні команди що будуть виконані під час кожного запуску відповідної функції. Тому важливо, щоб додатковий код був максимально швидкий і короткий

Список літератури

1. Robert Love. Linux Kernel Development (3rd edition), Pearson Education Inc., 2010. — 467c.
2. Andrew S. Tanenbaum. Modern Operationg Systems (4th edition), Pearson Education International, 2015 — 552 c.

.

УДК 519.68

КОМАРИЦЯ Р. В.

ЗАСТОСУВАННЯ РОЗПОДІЛЕНОГО БАЄСІВСЬКОГО КЛАСИФІКАТОРА ДЛЯ РОЗВ'ЯЗКУ ЗАДАЧІ РОЗПІЗНАВАННЯ ЕКЗОНІВ В ДНК

Розглядається метод розв'язку задачі розпізнавання кодуючих ділянок генів (екзонів) в ДНК з використанням наївного баєсівського класифікатора реалізованого на базі технології Apache Spark.

The subject of this article is a method for solving a recognition problem of exons in DNA using distributed naïve Bayes classifier implemented on the top of Apache Spark framework.

1. Вступ

Сучасна біоінформатика виникла в кінці 70-х років XX ст. з появою ефективних методів розшифрування послідовності ДНК. Важливою віхою у становленні та розвитку біоінформатики став проект «Геном людини», який був завершений на початку XXI ст.

Саме з цього часу біоінформатика перестала бути тільки допоміжним інструментом. При переході до аналізу повних геномів, роль комп'ютерних методів інтелектуального аналізу даних стала настільки важливою, що ці дослідження оформились у самостійний науковий напрямок.

Однією з ключових задач біоінформатики є задача розпізнавання білок-кодуючих областей ДНК (екзонів) які в свою чергу розділені сегментами некодуючої ДНК (інтронами).

З математичної точки зору поставлена проблема відноситься до задач розпізнавання, алгоритмів вирішення яких є доволі багато, більшість з яких базуються на методах машинного навчання. Та основна проблема полягає в тому, що існуючі об'єми даних неможливо обробити класичними методами машинного навчання, особливо враховуючи обмеження обчислювальних потужностей. Незважаючи на те, що розподілені обчислення відкрили нові шляхи для вирішення задач, які потребують великих обчислювальних потужностей, а стрімкий ріст даних обумовив практично повсюдне використання розподілених обчислень, їх використання в зв'язці з методами машинного навчання досі залишається відносно новою і мало дослідженою задачею. Розпаралелювання обчислень та використання обчислювальних кластерів для вирішення подібного роду задач дозволить значно підвищити швидкість обробки даних.

В статті подається метод вирішення поставленої задачі за допомогою наївного баєсівського класифікатора реалізованого на базі Apache Spark, що забезпечує можливість розгортання на кластері та горизонтального масштабування. Також проведено порівняльний аналіз швидкості роботи при використанні різних обчислювальних потужностей.

2. Постановка задачі

В задачі розпізнавання інтронів та екзонів на заданій ділянці гену задана послідовність символів S окремі елементи якої $S_i, i = 1,..n$ належать скінченному алфавіту R_s (рис.1) [1]. В випадку розпізнавання фрагментів гену, в ролі символів виступать нуклеотиди чотирьох типів: A, C, G, T.

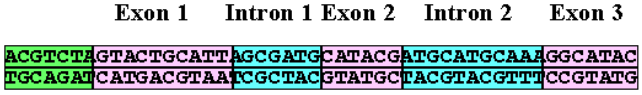


Рис. 1. Приклад послідовності ДНК

Звідси випливає визначення R_s:

$$R_s = \{A, C, G, T\}. \quad (1)$$

Кожному спостережуваному символу $S_i \in R_s$ ставиться у відповідність один прихований стан H_i з скінченної множини R_h. Для задачі розпізнавання фрагментів генів:

$$R_h = \{E, I\}, \quad (2)$$

де E – нуклеотиди, що входять в склад екзона, а I– нуклеотиди, що відповідають

інтронам. На основі введених позначень можна сформулювати наступну задачу.

Задача

Знайти алгоритм $A: R_S^* \to R_h^*$, який довільній послідовності спостережуваних символів $S \in R_s^n$ ставить у відповідність ланцюг прихованих станів такої ж довжини $H \in R_h^n$ згідно з визначеним критерієм якості L

$$A(S) = \arg max_H L(S, H). \quad (3)$$

3. Модель наївного баєсівського класифікатору

Нехай змінна може приймати значення з деякої множини значень V. Змінна x описується змінними $(a_1, a_2, \dots, a_n)$ – атрибутами. Необхідно знайти найбільш ймовірне значення x, тобто:

$$v_0 = \arg max_{v \in V} P(x = v | a_1, a_2, \dots, a_n). \quad (4)$$

За теоремою Баєса:

$$v_0 = \arg max_{v \in V} \frac{P(a_1, a_2, \dots, a_n | x = v) P(x = v)}{P(a_1, a_2, \dots, a_n)}. \quad (5)$$

Оцінити $P(x = v)$ легко (при наявності навіть невеликого числа тестових даних можна оцінити частоту появи кожного іх них). Але оцінити різні $P(a_1, a_2, \dots, a_n | x = v)$ неможливо, оскільки їх кількість є занадто великою. Для того, щоб отримати кожну із цих ймовірностей, потрібно кожну комбінацію атрибутів спостерігати декілька разів. Це не можливо на практиці. Тому наївний баєсівський класифікатор передбачає умовну незалежність атрибутів при умові даного значеня цільової функції:

$$P(a_1, a_2, \dots, a_n | x = v) = P(a_1 | x = v) \times$$
$$\times P(a_2 | x = v) \times \dots \times P(a_n | x = v) \quad (6)$$

Не дискретні атрибути повинні бути спочатку дискретизовані. Якщо даний клас і значення атрибута ніколи не зустрічались разом в наборі навчальних даних, тоді оцінка, яка базується на ймовірностях буде рівна нулю

Не зважаючи на простоту та наївність баєсівського класифікатора, неодноразово було показано його ефективність навіть при роботі з складними даними [2].

4. Алгоритм розпізнавання екзонів

Spark API надає розробнику можливість створювати та керувати конвеєром машинного навчання, при цьому немає необхідності зосереджувати свою увагу на інфраструктурі проекту та розподілених обчисленнях. Всю роботу, що пов'язана з розподіленими обчисленнями на себе бере RDD (Resilient Distributed Dataset), що в свою чергу являє собою розподілену таблицю. RDD підтримує широкий набір вбудованих функцій які дозволяють трансформувати дані (наприклад: селекція, фільтрація, об'єднання, перетин, трансформаціяб і т.д.). Крім цього важливою особливістю є можливість об'єднувати виклики функції і будувати ланцюг. Результатом виклику RDD є нова RDD.

Процесс машинного навчання часто представляє собою послідовність певних кроків (наприклад обробка даних, навчання системи, конвертація типів і тд) [3]. Конвеєр машинного навчання, що представлений в Spark API, також являє собою набір кроків. Кожен з кроків може бути представленим однією з двох сущностей – Transformer або Estimator.

Transformer - абстракція що включає в себе трансформатор ознак та моделі для навчання. З технічної точки зору Transformer реалізовуває метод transform() який відповідає за певне перетворення поточних даних (поточні дані мають бути представлені у форматі DataFrame - що по суті являє собою динамічну таблицю з колонками). Перетворення даних може включати в себе будь-які операції над даними включаючи агрегацію, фільтрацію, сортування, тощо. Результат перетворень зазвичай зберігається в вигляді колонок, що додаються до вже існуючих в поточному DataFrame.

Estimator абстрагує концепт алгоритму навчання чи будь-якого іншого алгоритму, що пристовоується або тренується на вхідних даних. З технічної точки зору Estimator реалізовуває метод fit(), який приймає DataFrame, та створює модель, що в свою чергу являється Transformer.

Етапи створення моделі наївного баєсівського класифікатора, необхідного для вирішення задачі розпізнавання, показано на рисунку 2.

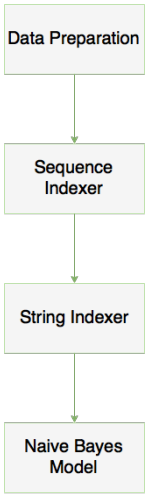


Рис. 2. Процес створення моделі наївного баєсівськго класифікатора

Data Preparation (Transformer) – зчитування та обробка початкових даних (зчитування файлу з екзонами і генами та їх агрегація). Зазвичай даний крок являється першим при побудові конвеєру.

String Indexer (Transformer) – кодує колонку ярликів (labels), що задані у вигляді текстових рядків в колонку індексів даних ярликів [4]. Дані індекси сортуються за частотою використання ярлика (так наприклад найбільш вживаному ярлику буде присвоєно індекс 0). Якщо ярлики буду представлені у вигляді числових значень, то кожен ярлик буде преведено до текстового типу і після цього проіндексовано.

Sequence Indexer (Transformer) – виконує задачу аналогічну до String Indexer, але з однією відмінністю, а саме: значення вхідної колонки задається у вигляді масиву текстових рядків, а не окремого текстового рядку. Всі строкові значення мають належати одному спільному алфавіту.

Naive Bayes Model (Estimator) – наївний баєсівський класифікатор, який приймає на вхід колонку ярликів та колонку індексованих ознак. Результатом роботи класифікатора є нові колонки.

На рисунку 3 зображено існуючі колонки на кожному етапі конвеєру. Оранжевим кольором показані вхідні дані для наступного етапу.

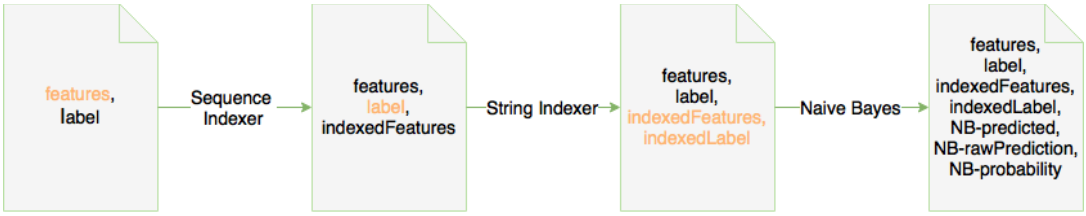


Рис. 3. Колонки DataFrame на етапах конвеєра

На малюнку 4 зображена архітектура високого рівня, що відображає процес роботи створеного програмного додатку, що складається з наступних кроків: створення RDD на базі існуючих даних, трансформації даних та створення вектора ознак, тренування моделі на базі отриманого вектора ознак, та розпізнавання екзонів на отриманій моделі.

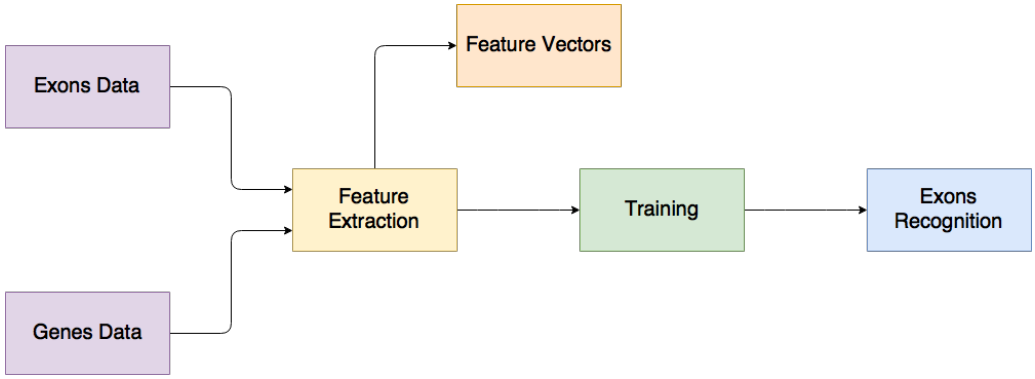


Рис. 4. Архітектура програмного додатку

5. Методологія випробувань

Для оцінки швидкості роботи алгоритму та побудови графіка залежності часу роботи відносно обчислювальної потужності було проведено серію випробувань з використанням різної кількості ядер процесора задіяних під час обчислень.

Для експерименту було використано реальні дані з Ensembl genome database project (проект, що забезпечує інтегрований доступ до баз даних геномів). Було використано базу Ensembl Genes 88 та збірку Human Genes (GRCh38.p10).

Вхідні дані представляють собою 2 текстові файли: набір екзонів та набір генів, що включають в себе дані екзони. Через фізичні обмеження тестової машини оброблені дані було відсемпловано та використано лише 5%. Розбиття тестових даних проводилось випадковим чином у наступному співвідношенні: 80% було відведено для навчання, 20% для тестування.

Програмна частина розроблена на мові Scala. Параметри процесора: 2.6 GHz Intel Core i7, оперативна пам'ять – 16 GB

6. Результати

Результати роботи приведені в таблиці 1.

Табл. 1. Час роботи алгоритму

	Кількість ядер процесора		
№	**1**	**2**	**4**
1	450	259	143
2	464	242	147
3	433	244	143
4	441	253	145
5	436	243	149
Avg.	444.8	248.2	145.4

З отриманих результатів, залежність часу виконання відносно кількості ядер процесору задіяних при обчисленні змінюється поліноміально (рис. 5).

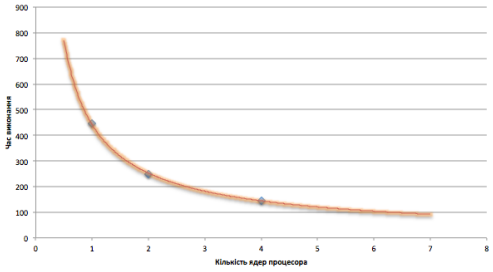


Рис. 5 Апроксимована залежність часу виконання від кількості ядер процесора

Дана залежність говорить про те, що мають місце розподілені обчислення. Проте в той же час ми бачимо що починаючи з певного моменту різниця в часі виконання майже не змінюється. Однією з причин подібної поведінки може бути складність данних та час затрачений на їх зчитування та підготовку для подальшої обробки алгоритмом класифікатора.

7. Висновок

Було розроблено метод розпізнавання екзонів в ДНК з використанням наївного баєсівського класифікатора. Розробка велась на базі технології Apache Spark. Дана технологія, завдяки своїй архітектурі, та концепту конвеєра, забезпечує розподіленість обробки даних та надає можливість надзвичайно простого горизонтального масштабування системи в майбутньому, без необхідності у внесенні будь-яких змін. Було показано залежність часу обробки даних відносно обчислювальної потужності.

Список літератури

1. Гупал А.М., Сергиенко И.В. Симметрия в ДНК. Методы распознавания дискркетных последовательностей. – Киев: Наукова думка, 2016. – 257 с.
2. Гупал А. М., Сергиенко И. В. Байесовская процедура - оптимальная процедура распознавания и преобразования // Пробл. упр. и информатики. - 2001. - № 3. - С. 5-15.
3. ML Pipelines: A New High-Level API for MLlib [Електронний ресурс] // Режим доступу: https://databricks.com/blog/2015/01/07/ml-pipelines-a-new-high-level-api-for-mllib.html
4. Extracting, transforming and selecting features [Електронний ресурс] // Режим доступу: https://spark.apache.org/docs/latest/ml-features.html

УДК 004.4

ФЕДОРОВ О.О.
СТІРЕНКО С.Г.

РОЗРОБКА ПЛАГІНІВ ДЛЯ ДОСЛІДЖЕННЯ ЗНІМКІВ ОПЕРАТИВНОЇ ПАМ'ЯТІ У VOLATILITY FRAMEWORK

Ця стаття зосереджена на поясненні внутрішньої роботи інструменту для аналізу оперативної пам'яті комп'ютера Volatility Framework. В ній описано створення та використання власних плагінів для аналізу знімку оперативної пам'яті комп'ютера. Стаття буде корисна для новачків в криміналістиці оперативної пам'яті комп'ютера, так як вона пояснює деякий недокументований код та функції Volatility Framework.

This article will be focused on the insides of computer memory forensics tool Volatility Framework. It describes creating and using own plugins for memory dump analysis. It will be useful for beginners in memory forensics, as it will explain some undocumented code and features of Volatility Framework.

1. Вступ

Криміналістичний аналіз комп'ютерної пам'яті це аналіз знімку оперативної пам'яті комп'ютера для встановлення всіх операцій що проводились під час знімку. Він зазвичай використовується для аналізу прихованих атак на комп'ютери, коли інших артефактів не залишилося на жорсткому диску, або якщо потрібно відновити інші цінні артефакти для розслідування, такі як мережеву активність, активності всіх процесів у системі, відкриті файли, ключі шифрування (які існують тільки в енергозалежній пам'яті), і т.д.

Знімок оперативної пам'яті комп'ютера може бути отриманий з працюючої системи з використанням різних методів, які мають власні переваги і недоліки. Деякими з них є:

- знімок з гіпервізора;
- файл сплячого режиму;
- файл аварійного знімку;
- знімок зсередини операційної системи;
- знімок спеціалізованим обладнанням.

Volatility Framework це фреймворк для аналізу та проведення криміналістичної експертизи оперативної пам'яті комп'ютера, реагування на інциденти безпеки та аналізу шкідливого програмного забезпечення. Volatility Framework кросплатформний, з відкритим кодом та реалізований на Python. Він здатний аналізувати дампи пам'яті з різних версій Windows, Linux, MacOS.

Існує вже готова база функцій, призначена для виконання різноманітного аналізу знімку пам'яті комп'ютера, яка включає лістинг інформації щодо запущених процесів, доступу до файлової системи, мережеву активність і т.д. Але іноді, для додання інших функцій, або налаштуванні криміналістичного процесу, нам може знадобитися інша функціональність, якої нема в готовому пакеті функцій.

Volatility Framework це модульний і дуже добре написаний фреймворк. Тому додання нового функціоналу, або зміну поведінки стандартних функцій проводити дуже легко. Для цієї мети

можна використати систему власних плагінів.

Існують різноманітні інструкції по використанню Volatility Framework, проте там не так багато інформації про те, як писати власні плагіни, а код фреймворку загалом не документований.

Тому пояснення роботи з власними плагінами Voltility Framework є актуальною темою.

2. Структура плагіну

При запуску Volatility Framework буде сканувати каталоги плагінів, специфічні для ОС знімку пам'яті що аналізується.

Для Windows ОС:
/volatility/plugins

Для Linux ОС:
/volatility/plugins/linux

Для macOS:
/volatility/plugins/mac

Якщо сумісний плагін буде знайдено, він буде завантажений.

Назва плагіну може бути будь-якою, але за згодою для зрозумілого іменування плагінів, плагіни для знімків Linux і Mac OS йдуть з префіксами linux_ * і mac_ * відповідно. Плагіни для знімків ОС Windows йдуть без префіксів.

Для того щоб Volatility Framework зміг впізнати та завантажити плагін, він необхідно повинен:

- мати наступні імпорти:
  import volatility.obj as obj
  import volatility.plugins.linux.common
  (приклад для знімку ОС Linux)
- визначити підкласс з назвою плагіну, та наслідуватися від linux_common.AbstractLinuxCommand (приклад для знімку ОС Linux)
- реалізовувати наступні функції:

```
def __init__(self, config, *args, **kwargs)
def render_text(self, outfd, data)
def calculate(self)
```

Крім того, можна (та рекомендується) наслідувати існуючі плагіни, якщо новий плагін має схожу функціональність, для того щоб уникати дублювання коду.

Основну структуру плагіну складають функції:

```
def __init__(self, config, *args, **kwargs):
def render_text(self, outfd, data):
def calculate(self):
```

def __init__(self, config, *args, **kwargs):

У цій функції проходить ініціалізація та ідентифікація плагіну фреймворком, визначення його параметрів командної строки та виконання необхідної підготовки даних, які можливо, будуть потрібні під час аналізу.

Параметри командної строки можливо задати наступним кодом:

```
config.add_option('OUR_PARAMETER',
        short_option = 'z',
        default = None,
        help = 'Displayed description of this parameter when help is called',
        action = 'store',
        type = 'str')
```

Тоді буде можливо використати його при запуску фреймворку, як наприклад:

```
python vol.py -f /STORAGE/ram.elf --profile LinuxUbuntu1510x64 linux_newplugin --our_parameter value
```

таким чином значення параметру буде збережено у змінній

self._config.OUR_PARAMETER

Важливо не перевизначити або перезаписати існуючі параметри командної строки, так як це може привести до непередбачуваної поведінки. Щоб їх знайти, зверніться до останньої інструкції по використанню фреймворку, або подивіться на клас плагінів, який наслідується. Потрібно бути особливо обережним з параметром «short_option». Проте параметр «short_option» в функції config.add_option можна взагалі пропустити, оскільки він є опціональним, та використовувати лише довгу версію.

def render_text(self, outfd, data):

У цій функції буде проводитися форматування, візуалізація і виведення результатів розрахунків користувацького плагіну.

Перше, що доцільно було б зробити, це надрукувати деякий заголовок таблиці вихідних даних, які будуть обчислюватися. Volatility Framework забезпечує корисну внутрішню функцію для форматування таблиць. Достатньо просто визначити список кортежів, які будуть містити ім'я стовпця та його довжину. Можливо використовувати ключове слово *[addrpad]*, щоб автоматично налаштувати необхідну ширину стовпця для типу покажчика, відповідного до архітектури знімку ОС що аналізується.

Наприклад:

```
self.table_header(outfd,
    [("Offset", "[addrpad]"),
    ("Name", "20"),
    ("Pid", "15")])
```

Далі, кожен елемент з результатів розрахунку буде доступний один за одним (від функції-генератора) в змінній «data», отже потрібно буде вивести його. Для того, щоб заповнити описану вище таблицю, ми можемо просто ітерувати через змінну «data» і додавати рядки, як наприклад:

```
for entry in data:
    self.table_row(outfd, entry[0], entry[1], entry[2])
```

def calculate(self):

Це основна функція (функція-генератор) плагіну, де міститься вся логіка і розрахунки. Фреймворк використовує можливості генераторів Python і створює об'єкти один за іншим, так, що не потрібно чекати розрахунку всіх результатів щоб передати їх до функції-виводу. Кожен об'єкт, який буде повернений з цієї функції-генератора буде елементом змінної «data» у функції «render_text».

Наведемо приклад найпростішої функції-генератора:

```
for x in range(0, 3):
    yield (x*10 + 1, x*10 + 2, x*10 + 3)
```

з такою функцією “calculate” плагін буде працювати наступним чином:

1. запуск функції *__init__*
2. плагін отримає параметри командної строки
3. запуск функції *render_text*
3.1 функція *render_text* виведе заголовок
3.2.1 функція *render_text* намагається отримати доступ до першого елементу змінної “data”
3.2.2 функція *calculate* поверне перший елемент (1, 2, 3)
3.2.3 функція *render_text* виведе перший рядок результатів (1, 2, 3)

3.3.1 функція *render_text* намагається отримати доступ до другого елементу змінної “data”

3.3.2 функція *calculate* поверне другий елемент (11, 12, 13)

3.3.3 функція *render_text* виведе другий рядок результатів (11, 12, 13)

3.4.1 функція *render_text* намагається отримати доступ до третього елементу змінної “data”

3.4.2 функція *calculate* поверне третій елемент (21, 22, 23)

3.4.3 функція *render_text* виведе третій рядок результатів (21, 22, 23)

Таким чином описано роботу елементарного плагіну Volatility Framework який буде працювати однаково на всіх платформах.

Наступне завдання полягає в тому, щоб власне виконати певний аналіз і розрахунки на знімку пам'яті. З цього моменту робота плагіну буде залежати від платформи, від того як реалізована ОС знімок якої аналізується, і на скільки вона покрита Volatility Framework. На даний момент, найкращий спосіб дізнатися, як використовувати всі реалізовані функції - це прочитати код вже реалізованих плагінів. Те, що описано у даній статті, а саме структура та основи того, як працює плагін, допоможе краще зрозуміти їх.

Список літератури

1. VOLATILITY FOUNDATION - Project Web Page [Електронний ресурс] : [Веб-сайт] – Режим доступу: http://www.volatilityfoundation.org
2. Volatility - An advanced memory forensics framework – GitHub Repository [Електронний ресурс] : [Веб-сайт] – Режим доступу: http://www.volatilityfoundation.org.

УДК 004.942

О.А.МОРОЗОВА

ЗАСТОСУВАННЯ ЛІНГВІСТИЧНОГО МОДЕЛЮВАННЯ ДЛЯ ПРОГНОЗУВАННЯ ЧАСОВИХ РЯДІВ

Розглядається алгоритм прогнозування часового ряду за допомогою лінгвістизації та з використанням прихованих Марковських моделей та формальних граматик. Описано процедури інтервалізаціїї часового ряду, формалізації граматики, використання моделей Маркова

A time series prediction algorithm that uses linguistization, hidden Markov models and formal grammars has been reviewed. Time series intervalization, grammar formalization and the use of Markov models have been described.

1. Вступ

Лінгвістичне моделювання понині широко використовується і досліджується. Досліджуються нові методи лінгвістичного моделювання на основі різних математичних підходів – кластеризації, семантичних мереж, нейролінгвістичного моделювання.

У дослідженні запропоновано метод прогнозування часових рядів за допомогою лінгвістизації та інтервалізації ряду, визначення формальної граматики з використанням прихованих марківських моделей. На даний час цей метод не є достатньо вивченим, проте має перспективи для подальшого дослідження.

2. Лінгвістизація часових рядів

Завданням лінгвістичного моделювання є перетворення часових рядів, експериментальних даних до лінгвістичних послідовностей та відновлення за ними формальної граматики мови. Це дозволяє вирішувати такі проблеми, як: прогнозування часових рядів, розпізнання образів.

Лінгвістичне моделювання передбачає перетворення чисельних образів до символьного вигляду.

Лінгвістичне моделювання – комплекс методів, методик та алгоритмів, які використовують процес перетворення числових масивів інформації до лінгвістичних послідовностей, на основі яких відновлюється формальна граматика. [1]

Лінгвістичне моделювання базується на трьох основних підходах: структурний підхід та математична лінгвістика, інтервальні обчислення та робастні методи, сучасні методи ймовірнісного моделювання. [2]

Одним з методів прогнозу є структурний (синтаксичний) підхід. Він базується на деяких принципах розпізнавання образів, яка складається з трьох основних частин – блока попередньої обробки, блока опису об'єкта, блока синтаксичного аналізу. На базі структурного підходу розроблено наступний метод створення лінгвістичної послідовності на базі числового ряду. Слід виконати наступні перетворення:

- Підрахувати різницю $\Delta y(i) = y(i) - y(i+1), i = \overline{1,N}$ між сусідніми значеннями ряду;
- Виключити дублікати з отриманого ряду;
- Відсортувати ряд за зростанням (чи за спаданням);
- Виокремити від'ємну $(a(k))$ та додатню $(b(k))$ послідовність зі значень $\Delta y(i)$.

Для розбиття множини чисельних даних ряду може використовуватись інтервальний підхід. Згідно [3] дамо основні визначення інтервалізації.

Нехай X та Y – частково впорядковані множини. Кожну з цих множин вважатимемо умовно повними структурами S(X), S(Y).

Якщо $a, b \in S(X)$ та $a \leq b$, то множину $I(a,b) = [a,b] = \{x \in X, a \leq x \leq b\}$ будемо називати інтервалом на S(X).

Шириною інтервалу $I[a_i, b_i]$ будемо називати величину $\omega(a_i, b_i) = b_i - a_i$

Можливе використання наступних типів інтервалів:
- рівнозначні інтервали;
- логарифмічні інтервали;
- рівноімовірнісні інтервали;
-інтервали за певним розподілом (Пуасона, нормальним, бета- і т.п); [4]

При рівнозначній інтервалізації N-го рівня множини X маємо:

$$\omega(a_1, b_1) = \omega(a_2, b_2) = \cdots = \omega(a_n, b_n)$$

При рівноймовірній (або рівночастотній) інтервалізації маємо

$$\dim\{I[a_1, b_1]\} = \dim\{I[a_2, b_2]\} = \cdots = \dim\{I[a_i, b_i]\} = \cdots = \dim\{I[a_N, b_N]\}$$

В результаті інтервалізації отримаємо наступні множини інтервалів:
$I_{0,1} = [a_0, a_1], I_{1,2} = [a_1, a_2], \dots, I_{N-1,N} = [a_{N-1}, a_N]$, де $a_0 = \min(a(k))$, $a_n = 0$;
$J_{0,1} = [b_0, b_1], J_{1,2} = [b_1, b_2], \dots, J_{N-1,N} = [b_{N-1}, b_N]$, де $b_0 = 0, b_n = \max(b(k))$;

Після вибору алфавіту (наприклад латинського) поставимо кожному члену послідовностей $(a(k))$ та $(b(k))$ символи алфавіту a^i та b^i $i = \overline{1, K}, j = \overline{1, L}$.

Зауважимо, що потужність алфавіту повинна бути не набагато меншою, ніж послідовність $a(k) \oplus b(k)$. У результаті даних перетворень була отримана лінгвістична послідовність c_i.

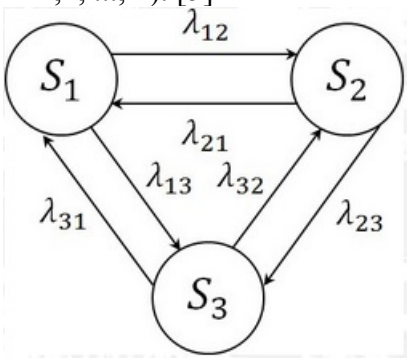
3. Приховані марковські моделі

Для побудови формальної граматики використовуються приховані марківські моделі. Теорія прихованих марківських моделей не нова. Її основи опублікував Баум на початку 60-х и 70-х років. Перші ПММ були використані в розпізнаванні мови і телекомунікаційних системах.

Нещодавно модель отримала широке застосування.

Марковський випадковий процес з дискретними станами і дискретним часом називають ланцюгом Маркова. Для такого процесу моменти коли система S може змінювати свій стан, розглядають як послідовні кроки процесу, а в якості аргументу, від якого залежить процес, виступає не час t, а номер кроку 1,2,...,k,.. Випадковий процес в цьому випадку характеризується послідовністю станів S (0), S (1), S (2), ..., S (k), ..., де S (0) - початковий стан системи (перед першим кроком); S (1) - стан системи після першого кроку; S (k) - стан системи після k-го кроку.

Подія S (k) = Si, являє собою те, що відразу після k-го кроку система знаходиться в стані Si, та є випадковою подією. Послідовність станів S (0), S (1), S (2), ..., S (k), ... можна розглядати як послідовність випадкових подій. Така випадкова послідовність подій називається Марківського ланцюгом, якщо для кожного кроку ймовірність переходу з будь-якого стану Si в будь-якому Sj не залежить від того, коли і як система прийшла в стан Si. Можливими станами ланцюга Маркова називаються ймовірності того, що після k-го кроку (і до (k + 1)-го) система S буде знаходитися в стані Si (i = 1,2, ..., n). [5]



Мал. 1.1. Ланцюг Маркова з трьома станами

На малюнку S1, ..., X3 - стану процесу; λ12 - ймовірність переходу зі стану S1 в стан S2, λ23 - ймовірність переходу зі стану S2 в стан S3 і т.д.

Приховані марківські моделі використовуються в тому числі для моделювання і прогнозування фінансових часових рядів. Цей метод базується на теорії марковських ланцюгів, але за умови, що ми не знаємо фактичний стан системи, а тільки спостерігаємо вихідні значення. Іншими словами, даний алгоритм дозволяє досліджувати приховані зв'язки всередині даних, які не можна спостерігати безпосередньо.

Для ілюстрації розглянемо приклад курсу валюти USD-UAH. Стан системи si приймає наступні значення: ціна зросте,

ціна знизиться або ціна не зміниться, а ймовірності переходу від одного стану до іншого – aij. Наприклад, ймовірність підвищення ціни завтра за умови, що сьогодні ціна не змінилася (стосовно вчорашньої ціни) a_{13}=0,2.

Природно ввести умову рівності одиниці всіх імовірнісних переходів з кожного стану, тобто $\sum_{j=1}^{N} a_{ij} = 1, i = \overline{1,N}$ де N – кількість станів системи.

Властивість ланцюгів маркова – залежність поточного стану системи тільки від минулого значення, тобто:

$$P(q_t|q_{t-1}, \dots, q_1) = P(q_t|q_{t-1})$$

Де $P(q_t|q_{t-1})$ – умовна вірогідність події q_t при умові, що попередня подія була q_{t-1}.

Марковський ланцюг, в якому між будь-якими двома станами ймовірність переходу не дорівнює нулю, називається повним, або ергодичним. Зміну цін активів часто можна вважати марковським процесом. Це означає, що все, що нам необхідно для прогнозу, - це поточне значення активу, а вся попередня історія вважається незначущою. Марківська властивість цін активів відображає так звану слабку форму ефективного ринку, згідно з яким поточна ціна відображає всю минулу інформацію, що стосується даного активу.

Марковські ланцюги корисні, коли ми хочемо підрахувати ймовірність конкретної послідовності подій. Однак події, що цікавлять часто не можна спостерігати безпосередньо. В такому випадку використовують приховані моделі Маркова (ПММ).

Скінченний дискретний ланцюжок визначається:

- множиною станів, подією є перехід з одного стану в інший в результаті випадкового випробування;
- вектором початкових ймовірностей (початковим розподілом);
- матрицею перехідних ймовірностей, що характеризує ймовірність переходу процесу з поточним станом у наступний, та сума ймовірностей переходів з одного стану дорівнює 1. [1]

Для c_i підрахуємо частоту існування пар символів $(c_i c_{i+1})j = \overline{1, N-1}$ і побудуємо таблицю ймовірностей виникнення символу $c_{i+1} P_{j+1}(c_{i+1} \mid c_i)$.

В послідовності c_i підрахуємо частоту існування трійок $(c_{i-1} c_i c_{i+1})j = \overline{1, N-2}$ та будуємо таблицю ймовірностей $P_{j+1}(c_{i+1} \mid c_i c_{i-1})$. Аналізуємо частоту існування послідовностей $P_{j+1}(c_{i+1} \mid c_{i-k} \dots c_i)$. (ймовірність появи символу c_{i+1} за умови, що відомі послідовності попередніх символів). [1]

4. Відновлення граматики на основі лінгвістичних ланцюжків

Формальна граматика G – це четвірка (V_T, V_N, P, S), де:

V_T – алфавіт термінальних символів (терміналів),

V_N – алфавіт нетермінальних символів (нетерміналів), що не перетинаються з V_T,

P – кінцева підмножина множини $(V_T \cup V_N)^+ \times (V_T \cup V_N)^*$; елемент (α, β) множини P називається правилом виводу і записується у вигляді $\alpha \to \beta$,

S – початковий символ (мета) граматики, $S \in V_N$.

Мовою, що породжується граматикою $G = \langle V_T, V_N, P, S \rangle$, називається множина $L(G) = \{\alpha \in T^* | S \to \alpha\}$

Іншими словами, L (G) - це всі ланцюжки в алфавіті V_T, які виводяться з S за допомогою правил P.

Граматики G_1 і G_2 називаються еквівалентними, якщо L (G_1) = L (G_2)

Граматики G_1 і G_2 майже еквівалентні, якщо $L(G_1) \cup \{\varepsilon\} = L(G_2) \cup \{\varepsilon\}$, тобто граматики майже еквівалентні, якщо мови, ними породжувані відрізняються не більше, ніж на ε.

Ланцюжок в алфавіті належить мові, породжуваною граматикою $G = \langle V_T, V_N, P, S \rangle$, тільки в тому випадку, якщо існує її вивід з початкового символу S цієї граматики.

По розширеній матриці передування побудуємо правила ймовірнісної граматики. Для кожної ненульової клітинки будується правило граматики.

5. Висновки

Було розглянуто загальний алгоритм реалізації методу на базі інтервального підходу. По розширеній матриці передування, побудованій за допомогою прихованих марківських моделей досліджується побудова формальної граматики та побудова прогнозу. Метод має широкий спектр застосування, але у роботі ми зупинилися лише на прогнозуванні часового ряду. Описаний метод дозволяє вирішувати такі проблеми, як, наприклад, прогнозування часових рядів, розпізнання образів.

Список літератури

1. Fraser A. M. Hidden Markov models and dynamic systems / Fraser Andrew M. – Philadelphia: Society for Industrial and Applied Mathematics. 2008.
2. Баклан И.В. Лингвистическое моделирование: основы, методы, некоторые прикладные аспекты // Системные технологии. Региональный межвузовский сборник научных работ. – Выпуск 3 (74). –Днепропетровск, 2011. – с.10 – 19.
3. Канторович Л.В. О неоторых новых подходах к вычислительным методам и обработке наблюдений // Сибирский Математический Журнал. – 1962. – Т.3, No.5. – С. 701-709.
4. Баклан І. В. Інтервальний підхід до побудови лінгвістичної моделі / І. В. Баклан // Систем. технології. - 2013. - № 3. - С. 3-8. - Бібліогр.: 4 назв. - укр.
5. Вентцель Е.С., Овчаров.Л.А. Завдання і вправи з теорії ймовірностей. 5-е изд., испр. - М.: Академия, 2003.— 448 с
6. Кельберт М. Я., Сухов Ю. М. Вероятность и статистика в примерах и задачах. Т. II: Марковские цепи как отправная точка теории случайных процессов и их приложения. М.: МЦНМО, 2009. — 295 с.: ил.
7. Степанкова Г. А. Приховані марковські моделі: класифікація / Г. А. Степанкова, І. В. Баклан // Систем. технології. - 2012. - № 4. - С. 18-27. - Бібліогр.: 14 назв. - укр.
8. Черкай А.Д. Бухгалтерский учет и его универсальный семантический код: новый метод быстрого обучения бухгалтерскому учету [Текст] / А.Д. Черкай. – М.: 2010

УДК 004.931

ТРИГУБИШИН І.Л.
СЕЛІН І.М.

ЛІНГВІСТИЧНА МОДЕЛЬ ДЛЯ АНАЛІЗУ ЦИФРОВИХ АУДІО ВІДБИТКІВ

В статті розглянуто питання побудови лінгвістичної моделі аудіо сигналу. Запропоновано підхід представлення цифрових аудіо відбитків у вигляді лінгвістичної послідовності. Приведено приклад використання лінгвістичної моделі для ідентифікації аудіо сигналів.

The article deals with the construction of a linguistic model of an audio signal. The approach of presentation of digital audio prints in the form of a linguistic sequence is proposed. An example of using a linguistic model for identifying audio signals is given.

1. Отримання відбитків аудіо сигналів

Аналіз аудіо сигналів, в першу чергу, пов’язаний з отриманням і ідентифікацією дескрипторів для відображення характеристик музичної системи, індексацією музики на основі цих дескрипторів і розробкою пошукових систем музичних творів.

У ядрі багатьох додатків, таких як ідентифікація аудіо, системи рекомендацій музики активною темою дослідження є подібність музики. Основна проблема дослідження в галузі музичної подібності полягає в тому, щоб визначити відповідну відстань або міру подібності. Ми повинні обрати музичні аспекти і описові дескриптори, рівень абстракції (при надто конкретному рівні будуть відкинуті деякі правильні варіанти, а при надто абстрактному рівні можливі помилкові спрацьовування), а також бажаний рівень деталізації або часові рамки. Окрім того, визначення подібності залежить від того, в якій сфері вона буде застосовуватися і може бути суб'єктивною якістю, для якої необхідне моделювання досліджень сприйняття на основі певної вибірки людей.

Для задачі класифікації використовуються дескриптори невеликого розміру, які витягуються безпосередньо з акустичного сигналу, і у яких зберігаються тільки важливі характеристики аудіо сигналу. У MPEG-7, стандарті опису аудіовізуального контенту запропоновані інструменти звукового опису, що описують контент. Витяг дескрипторів низького рівня є нормативним. За допомогою низькорівневих дескрипторів можна здійснювати пошук і фільтрацію аудіоконтенту щодо, наприклад, спектру, гармонії, тембру і мелодії. Дескриптори можна поділити на 2 основні групи: глобальні і тимчасові. Глобальні описують аудіо сигнал в цілому, а тимчасові лише для конкретного часового фрейму, який представляє собою частину аудіосигналу. Часових і спектральних параметри, які існують на даний момент, описують форму хвилі і спектральні характеристики лог-частот (наприклад, спектральний центр, спектральний розкид, спектральна площину), параметри сигналу: основна частоту і гармонію сигналів, спеціалізовані спектральні характеристики в лінійному частотному просторі.

Одним із специфічних типів аудіо дескрипторів є цифровий аудіо відбиток. Цифровий аудіо відбиток є компактною сигнатурою, що узагальнює вміст аудіо запису. Технологія цифрових аудіо відбитків є технологією ідентифікації аудіоконтенту на основі вилучення відповідних акустичних характеристик. Цифрові аудіо відбитки дозволяють здійснити моніторинг звуку незалежно від його формату без урахування мета-даних або впровадження водяних знаків [5].

В технології аудіо відбитків відбиток створюється, виходячи з перцептивно найбільш важливих компонентів звуку. В порівнянні з водяними знаками, такий метод менш уразливий для атак і спотворення, так як зміна відбитку означає зміну якості звуку. Крім того,

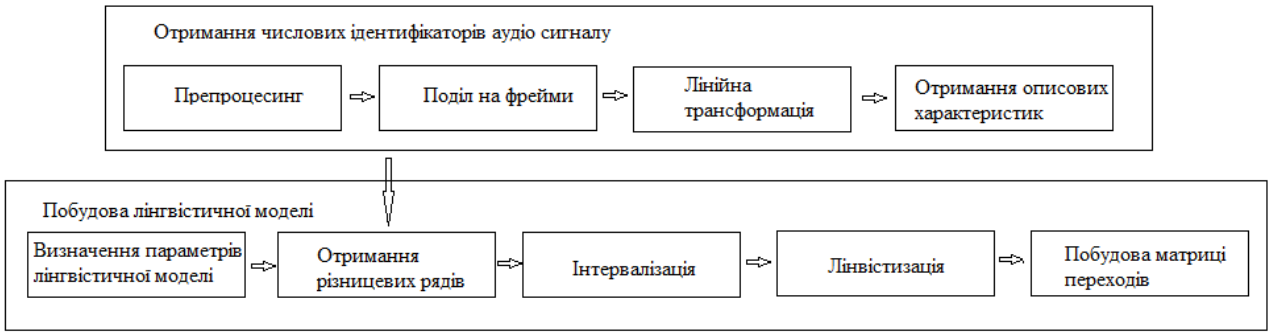


Рис 1. Етапи побудови лінгвістичної моделі

він не вимагає модифікації аудіоконтенту. Як недолік, обчислювальна складність технології аудіо відбитків, як правило,вище, технології водяних знаків, окрім того є необхідність підключення до сховища відбитків пальців. На відміну від водяних знаків, відбиток не є незалежним від змісту. Тому, наприклад, не представляється можливим розрізняти перцептивно ідентичні копій запису. Так само, як з технології водяних знаків, сфера застосування відбитків є більшою ніж просто ідентифікація. Зокрема, вона може також використовуватися для перевірки цілісності вмісту.

При проектуванні цифрових аудіо відбитків важливим завданням є вибір відповідного способу їх отримання для точної репрезентації фрагменту аудіосигналу. На практиці, мультимедійний вміст має схильність до деградації через стиснення, поліпшення, додавання шуму і аналого-цифрове перетворення.

Таким чином, система звукової ідентифікації повинна мати можливість допускати деяку модифікацію звуку, розрізняючи один звуковий запис від іншого. У загальному випадку функція цифрових аудіо відбитків повинна мати такі властивості:

- надійність (інваріативність за перцептуальною схожістю). Цифрові аудіо відбитки спротвореного аудіосигналу повинні бути схожі на відбитки вихідного звуку
- паралельна незалежність. Два звукозаписи, які сприймаються по-різному, повинні мати різні відбитки пальців.
- репрезентативність. Аудіо відбиток повинен бути побудований на основі перцептивного сприйняття аудіо сигналу[2]

Процес отримання цифрових аудіо відбитків з допомогою лінгвістичного моделювання складається з наступних етапів: препроцесинг, нормалізація лінійна трансформація, поділ на фрейми, отримання описових характеристик, побудова лінгвістичної моделі, побудова матриці переходів на основі лінгвістичної моделі. Як зображено на рисунку 1, в результаті обчислення описових характеристик ми отримаємо короткий числовий ідентифікатор аудіо сигналу. Лінгвістичні методи дозволяють перетворити це цифрове представлення в символьну форму.

На етапі препроцесингу звук оцифровується і перетворюється в заданий формат: здійснюється усереднення лівого і правого каналів і приведення до певної частоти дискретизації(44,1 КГц).

Для стискання даних, видалення шуму і приведення їх до одного порядку здійснюються лінійна трансформація і нормалізація. Звукові дані нормуються в діапазоні (-1,1). Як лінійну трансформацію можна застосувати швидке перетворення фур'є, дискретне перетворення Фур'є, каскади Хаара, трансформацію Уолша-Адамса.

Ключовим припущенням при вимірюванні характеристик є те, що сигнал можна розділити на стаціонарні інтервали в декількох мілісекунд. Таким чином, сигнал ділиться на кадри розміром, порівнянних зі тривалістю основної акустичні події. Кількість кадрів в секунду називається частотою кадрів. Конічна функція вікна застосовується до кожного блоку, щоб мінімізувати розриви на початку і в кінці. Перекриття повинні бути застосовані для забезпечення стійкості до зміщення, щоб покрити випадок, коли вхідні дані не ідеально вирівняні.

Після розбиття на фрейми і нормалізації, ми визначаємо описову характеристику кожного фрейму, яка буде основу для подальшої лінгвістизації. Для отримання цифрових аудіо відбитків, що задовольняють властивостям визначеним вище, використаємо алгоритм на основі нормалізованого моменту спектрального піддіапазону.

Нехай P[n, k] - короткочасний спектр потужності аудіосигналу в k-ому частотному буфері кадру номер n. Тоді v-ий момент m-ого піддіапазону визначається як

$$\zeta_v[n,m] = \sum_{k=C[m]+1}^{C[m+1]} k^v P[n,k]$$

де *C[m]* визначає частотні границі m-ого піддіапазону. Тоді нормалізовані моменти першого і другого порядку можна визначити як:

$$\eta_1[n,m] = \frac{\zeta_1[n,m]}{\zeta_0[n,m]} \quad (2)$$

$$\eta_2[n,m] = [\frac{\zeta_2[n,m]}{\zeta_0[n,m]} - (\eta_1[n,m])^2]^{\frac{1}{2}}$$

Нормовані моменти спектрального діапазону стійкі до місцевих спектральних спотворень, таких як вирівнювання, тощо[1]. Значення цієї описової характеристики визначаються для кожного фрейму з вхідних даних. Таким чином ми перетворюємо вхідний аудіо сигнал в цифровий вигляд, який відповідає перцептивному сприйняттю музики. Всі ці дані утворюють масив, який подається на вхід лінгвістичної моделі.

2. Побудова лінгвістичної моделі на основі цифрових аудіо відбитків

Головним завданням лінгвістичного моделювання є перетворення чисельних рядів, експериментальних даних, багатомірних даних до лінгвістичних послідовностей та відновлення за ними формальної граматики мови відповідного сигналу. В нашому випадку ми перетворюємо звукові в символьні.

Лінгвістична модель – побудована на основі лінгвістичного моделювання сукупність символьних послідовностей за обраними параметрами лінгвістизації та відновлена на її основі формальна граматика. Основна ідея переходу від множини чисельних значень до символьного алфавіту є розбиття цієї множини на чисельні інтервали. Можна визначити наступні основні етапи побудови лінгвістичної моделі: отримання різницевих рядів, інтервалізація, лінгвістизація, побудова матриці переходів. Розглянемо детальніше ці етапи.

– Отримання різницевих рядів.

На цьому етапі з вхідних рядів ми отримуємо ряди, які характеризують динаміку зміни цифрових аудіо відбитків: швидкість, прискорення. Таким чином різницеві ряди являються похідними від вихідного[2].

На вхід даного етапу подається потужність вхідного вектора $\bar{A}$ k. Потужність вихідного вектора значень різницевих рядів $\overline{D}$ n = k-1. Вихідний вектор отримуємо наступним чином:

$$\forall d_i \in \overline{D}\ d_i = a_{i+1} - a_i$$

– Інтервалізація

На етапі інтервалізації здійснюється розбиття відсортованого різницевого ряду на множину інтервалів. Перед початком цього етапу необхідно визначити алфавіт користувача, кожній букві якого буде відповідати певний інтервал. Таким чином кожен елемент інтервалу буде мати у відповідність літеру алфавіту.

Вхідними даними цього етапу є отриманий з минулого етапу вектор значень $\underline{D}$, потужністю n, а також потужність алфавіту k, при цьому потужність алфавіту повинна бути меншою ніж потужність вектора значень. В результаті інтервалізації ми отримуємо k векторів, в кожному з яких буде значення початку і кінця інтервалу.

Окрім потужності алфавіту, для задачі інтервалізації необхідно визначити власне тип інтервалізації. Можна визначити наступні типи інтервалізації:

- інтервали різнозначні;
- логарифмічні інтервали;
- рівноймовірні інтервали;
- інтервали з певним розподілом ймовірностей[3].

Тип інтервалізації визначається, виходячи з задачі і характеру вхідних даних. При рівнозначній інтервалізації N-ого рівня множини X маємо

$$w(a_1, b_1) = w(a_2, b_2) = \ldots = w(a_N, b_N)$$

де $w(a_i, b_i) = b_i - a_i$

При рівноймовірній інтервалізації маємо

$$D[I_1] = D[I_2] = \ldots = D[I_N]$$

де $D[I_i]$ – кількість елементів, що потрапляють до певного інтервалу.

Якщо кількість інтервалів k< n, то

$$\sum_{i=o}^{k} D[I_i] = n$$

Частотність інтервалу $I[a_i, a_{i+1}]$ на часовому рядку потужності N

$$v_{i,i+1} = v[a_i, a_{i+1}] = \frac{D\{I[a_i, b_i]\}}{N}$$

Для інтервалів з певним розподілом ймовірностей межі інтервалів визначаються на основі параметрів розподілу. Таким чином, необхідно визначити не лиш типу інтервалізації, а й параметри, необхідні для даного типу.

– Лінгвістизація

Призначенням даної підзадачі є отримання лінгвістичного ланцюжку шляхом знаходження відповідної літери алфавіту для кожного значення різницевого ряду, тобто перетворення вхідних даних в символьний формат. Літера алфавіту однозначно відповідає певному інтервалу з множини інтервалів, отриманих в результаті розв'язання попередньої задачі.

Для виконання цієї підзадачі необхідно мати вхідний вектор чисел, що відповідає вхідній послідовності, а також вектор пар значень меж інтервалів. В результаті ми отримуємо вектор цілих чисел, потужністю рівною потужності вхідного вектора, кожне значення якого буде відповідати номеру інтервалу відповідного елементу вхідного вектора.

3. Побудова матриці переходів

Призначенням даного етапу є побудова матриці переходів між двома літерами алфавіту в реченні. Алфавіт та його літери визначені у підзадачі інтервалізації, а речення – у задачі лінгвістизації.

На вхід подаються вектор значень номерів інтервалів і потужність множини інтервалів. Для кожного символу з речення ми визначаємо ймовірність переходу у кожен інтервал з множини інтервалі. На основі векторів для кожного значення отримуємо квадратну матрицю переходів розмірністю n на n.

Отриману послідовність аналізують на наявність граматичних конструкцій. На виході отримуємо список граматичних конструкцій з імовірностями їх наявності в процесі, а також матрицю імовірностей переходу з символу в символ[2].

4. Ідентифікація аудіо відбитків на основі лінгвістичної моделі

Як було зазначено вище, проблема ідентифікації сигналу пов'язана з визначенням подібності між різними треками. На основі значень подібності між сигналами вирішується задачу сегментації і ідентифікації аудіо сигналів.

Коли у нас є велика база даних відбитків, нам потрібно визначити, чи представлений вхідний сигнал в базі даних, а також визначити цифрові аудіо відбитки, на які цей сигнал найбільш схожий. Для цього ми визначаємо рівень подібності ланцюжка з представленими в репозиторії відбитками.

Для цього до вхідного сигналу застосовуються перші етапи утворення цифрових відбитків сигналу. Тобто спочатку ми отримуємо числові ідентифікатори сигналу на основі описових дескрипторів. Після цього на основі вже визначених раніше параметрів лінгвістичної моделі здійснюємо лінвістизацію. Таким чином, ми отримуємо речення і нам необхідно з'ясувати ймовірність того, що речення було породжено відповідною граматикою.

Використовуючи матриці переходів ми отримуємо оцінку подібності для кожного відбитка в базі з вхідним сигналом. Щоб вирішити, що є правильна ідентифікація, оцінка повинна бути вище певного порогового значення. Нелегко вибрати поріг, оскільки він залежить від кількості елементів в базі, рівню зашумленості вхідних даних, способу отримання відбитків. Чим більше відбитків у базі даних, тим вище ймовірність помилкового збігу. А якщо задати завеликий поріг, може вийти так, що правильна відповідь не попаде в цей поріг.

5. Висновки

Були розглянуті методи отримання лінгвістичної моделі аудіосигналу. Ми отримали метод витягнення описових дескрипторів аудіосигналу для використання їх в якості цифрових аудіо відбитків на основі нормалізованого моменту спектрального піддіапазону.

Представлено спосіб лінгвістизації звукового сигналу за допомогою цифрових аудіовідбитків. Таким чином здійснено перетворення звукового сигналу в цифрову

форму, а після цього з цифрової форми в в форму слів.

З використанням лінгвістичної моделі запропоновано спосіб ідентифікації аудіосигналів на основі порівняння цифрових аудіовідбитків з матрицею переходів. Необхідні подальші дослідження на базі музичних творів для визначення ефективності і швидкості подібного методу.

Список літератури

1. Seo J S, Jin M, Lee S Audio Fingerprinting Based on Normalized Spectral Subband Moments. IEEE SIGNAL PROCESSING LETTERS, VOL. 13, NO. 4, APRIL 2006
2. P. Cano. A Review of Algorithms for Audio Fingerprinting. International Workshop on Multimedia Signal Processing, US Virgin Islands, December 2002
3. Баклан І. В. Інтервальний підхід до побудови лінгвістичної моделі / І. В. Баклан // Системні технології. - 2013
4. Баклан І. В. Лінгвістичне моделювання: основи, методи, деякі прикладні аспекти / І. В. Баклан// Систем. технології. - 2011. - № 3. - С. 10-19
5. M. Schedl , E. Gómez and J. Urbano. Music Information Retrieval: Recent Developments and Applications. Foundations and Trends in Information Retrieval Vol. 8, No. 2-3 (2014) 127–261

УДК 519.6

БЕРЕЖНЯК М.О.

ПРОГНОЗУВАННЯ УСПІШНОСТІ СТУДЕНТІВ ОНЛАЙН КУРСІВ

У статті описано підхід до прогнозування успішності студентів на основі даних, які генеруються під час навчального поцесу. Задачу формалізовано як задачу класифікації. Проведений проівняльний аналіз застосування найпоширеніших алгоритмів класифікації на заданій предметній області. На основі отриманих теоретичних результатів приведені можливості практичного застосування, а також обчислювальні задачі, які при цьому виникають.

This article describes the approach for the analysis of student progress basing on the data that had been generated during studying process. The task was formalized as a classification problem. The article provides comparative analysis of using most popular classification algorithms in the target domain. With outlined theoretical results, there is a description of possible usages of the solution and the problems and challenges that follow after that

1. Вступ

У 2011 році Стенфордський університет запропонував всім охочим безкоштовно пройти три своїх курси через мережу Інтернет. Лише в одному з цих курсів тоді взяло участь близько 160 тисяч студентів зі 190 країн світу. Саме так широка громадськість дізналася про новітній формат масових відкритих онлайн курсів (МВОК) – комбінації безкоштовних відеолекцій кращих викладачів, інтерактивних завдань та форумів для обговорення навчальних матеріалі.

На червень 2014 року в світі існувало вже близько 2600 МВОК – 327% зростання порівняно з 2013. Масові онлайн курси створюють десятки провідних університетів по всьому світу, а такі країни як Франція, Китай та Йорданія заснували національні платформи МВОК.

Prometheus[1] - це сервіс масових відкритих онлайн курсів, побудований на платформі Open edX, що розроблена Масачусетським інститутом технологій спеціально для створення МВОК. У вересні 2014 року на платформі Prometheus з'явилися 2 перших масових відкритих онлайн-курси українською мовою: «Фінансовий менеджмент»[2] та «Історія України: від Другої світової війни до сучасності»[3].

Мета будь-якої навчальної онлайн-платформи – збільшувати освіченість населення. Світова практика показує, що серед усіх зареєстрованих на онлайн-курс користувачів, в середньому його завершують лише 10% студентів. Виникає природня потреба у заохоченні студентів, підвищення їхньої мотивації під час проходження курсу. Проблема є комплексною, оскільки спочатку треба ідентифіковувати неуспішних користувачів, а потім вживати заходів, щоб їх мотивувати на завершення курсу. Ці заходи можуть варіюватися від звичайної розсилки по електронній пошті нагадувань до персоналізованих змін у інтерфейсі онлайн-платформи, структурі курсу, його змісті тощо[4]. Тут буде розглянуто підходи до прогнозування успішності користувачів.

Prometheus має типічну для онлайн-курсів структуру, тобто принцип її роботи подібний до Coursera та edX. Будемо проводити аналіз даних, що відображають навчальний процес на платформі Prometheus. Зокрема використаймо інформацію про зазначені вище курси історії та фінансового менеджменту. Всі дані добровільно керівництвом Prometheus.

2. Постановка задачі

Користувачі платформи Prometheus навчаються за вільним графіком: під час старту курсу навчальні матеріали надходять щотижнево, доки не досягнуть кінця курсу, потім вони лишаються відкритими для всіх зареєстровиних на курс користувачів.

Обмежень часових по проходженню курсу немає.

У системі Prometheus користувачі навчаються, і для них характерна певна поведінка і результати успішності. Всього система має певну кількість користувачів користувачів. Результати успішності у правильності відповідей на запитання тестів виражаються у оцінці за кожний тест. Якщо користувач не зробив спроби проходження тесту – оцінка рівна 0. За проходження курсу студенти отримають сертифікат. Відповідно всі студенти курсу поділяються на тих, хто має сертифікат, і тих, хто не має. Для успішних і неуспішних студентів характерні певні типи поведінки і результативність. Надійною характеристикою системи можна вважати оцінки проходження тестів. Варто зазначити, що у кожного користувача може бути різна кількість оцінок, бо студенти не забов'язані проходити курс синхронно.

Таким чином проблема полягає у прогнозі успішності студента, на основі наявних оцінок у певний момент часу. Маємо задачу бінарної класифікації[7]: треба визначити завершить студент курс чи ні. Проведемо формалізацію задачі. Дано:

- кількість користувачів – s;
- кількість тестів для курсу – m;
- наявність сертифікату у *i*-го користувача - $c^i, i = \overline{1,s}$;
- оцінка проходження *j*-го тесту *i*-м користувачем - $g_j^i, j = \overline{1,m}, i = \overline{1,s}$.

Вихідні дані: булева(1 – так, 0 - ні) оцінка проходження курсу користувачем - $\hat{c}$. Об'єктом у даній задачі є користувач. X – простір характеристик об'єкта.

$$X = \{g_j^i, j = \overline{1,m}, i = \overline{1,s}\} \quad (1)$$

Y = {1,0} – множина класів(пройшов/не пройшов курс).

$$y^* : X \to Y \quad (2)$$

(2) – цільова залежність, що виражається через $c^i, i = \overline{1,s}$. Задача: знайти залежність (3), де x∈X, y∈Y; знайти $f(x) = \hat{c}$.

$$f : X \to Y \quad (3)$$

3. Обгрунтування розв'язання

Поставлена задача класифікації може бути вирішена за допомогою статистичних методів класифікації[5]: логістичної регресії, методу k найближчих сусідів, методу опорних векторів, наївного баєсівського класифікатору, дерева рішень. Поставлена задача передбачає бінарну класифікацію.

Основним принципом методу k найближчих сусідів є те, що об'єкт присвоюється того класу, який є найбільш поширеним серед сусідів даного елемента. Сусіди беруться виходячи з безлічі об'єктів, класи яких вже відомі, і, виходячи з ключового для даного методу значення k вираховується, який клас найбільш численний серед них[6].

Метод опорних векторів належить до групи граничних методів. Опорними векторами вважаються об'єкти множини, що лежать на цих межах. Основна ідея методу опорних векторів – перевід вихідних векторів у простір більш високої розмірності та пошук роздільної гіперплощини з максимальним проміжком у цьому просторі. Дві паралельні гіперплощини будуються по обидва боки гіперплощини, що розділяє наші класи. Роздільною гіперплощиною буде та, що максимізує відстань до двох паралельних гіперплощин. Алгоритм працює у припущенні, що чим більша різниця або відстань між цими паралельними гіперплощинами, тим меншою буде середня помилка класифікатора[6].

Наївний баєсівський класифікатор — класифікатор, що використовує теорему Баєса для визначення ймовірності приналежності спостереження (елемента вибірки) до одного з класів за умови того, що залежні змінні приймають задані значення. Перевагою цього підходу є те, що вимоги до розміру вибірки скорочуються від експотенційних до лінійних. Недолік — те, що модель точна лише у випадку, коли виконується припущення про незалежність[6].

Класифікація деревом рішень використовує дерево рішень як модель для прогнозу, що відображає наявні спостереження на цільову змінну. Деревовидні моделі у даному випадку називаються деревами класифікації. В цих структурах листки відповідають класам, а гілки – кортежам значень атрибутів вибірки[6].

Логістична регресія передбачає створення лінійної моделі шляхом оцінки її параметрів методом найбільшої правдоподібності. Інші

вищезгадані методи застосовуються для багатокласової класифікації і призначені для описання більш складних залежностей. Ще одна перевага логістичної регресії у меншому ризику перенавчання моделі[6].

4. Проведення експериментів

Для вибору алгоритму класифікації проведемо експеримент: запустимо кожен алгоритм на вибірці даних курсу «Фінансовий менеджмент» і оцінимо точність. Як мову програмування оберемо Python, оскільки для цієї мови розроблені зручні бібліотеки для наукових розрахунків та машинного навчання[7]. Експерименти будемо проводити у середовищі IPython Notebook. Також скористаймося бібліотекою scikit-learn, що надає однаковий інтерфейс до реалізації необхідних для експерименту алгоритмів. Для оцінки точності використаємо кросс-валідацію. Кросс-валідація – це метод оцінки аналітичної моделі і її поведінки на незалежних даних.

Проведемо дослідження кращого алгоритму на вибірках даних з курсів «Фінансовий менеджмент» та «Історія України: від Другої світової війни до сучасності». За змінні для прогнозування оберемо результати перших чотирьох тестів. Враховуючи те, що в загальносвітовій практиці онлайн курс завершуються в середньому 10% студентів, будемо порівнювати побудовані моделі з контрольною, яка завжди каже, що користувач курс не пройде. Результати експерименту наведено у таблиці 1:

Таблиця 1. Дослідження точності роботи алгоритмів

Алгоритм	Курс історії. Точність моделі	Курс фінансового менеджменту. Точність моделі
Логістична регресія	0.954	0.956
Наївний баєсівський класифікатор	0.956	0.954
К найближчих сусідів	0.944	0.952
Дерева рішень	0.949	0.951
Метод опорних векторів	0.955	0.955
Контрольна модель	0.924	0.894

У випадку курсу фінансового менеджменту найкращий результат показала логістична регресія, у випадку історії – наївний баєсівський класифікатор. Контрольна модель показує точність 0.89 для фінансового менеджменту та 0.92 для історії. Це означає, що курс пройшли 11% і 8% користувачів відповідно. Всі алгоритми показали точність порядку 0.95 для обох курсів. Проте різниця серед алгоритмів виявляється лише на тисячний долях точності, а це означає їхню практичну еквівалентність при розв’язанні поставленої задачі. Для визначеності обремо логістичну регресію як основний алгоритм. Для наочності побудуємо діаграми на основі отриманих результатів(рисунок 1, рисунок 2):

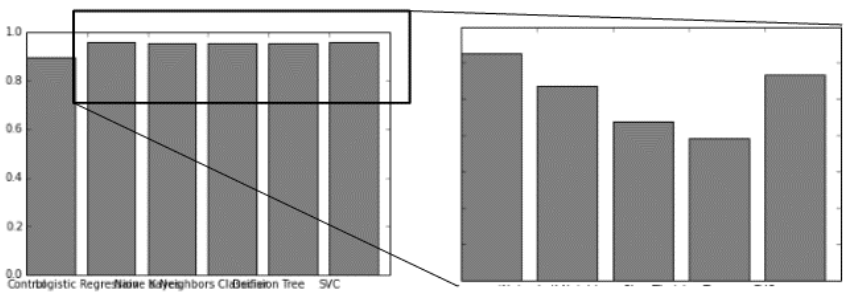


Рисунок 1. Курс історії. Порівняння алгоритмів

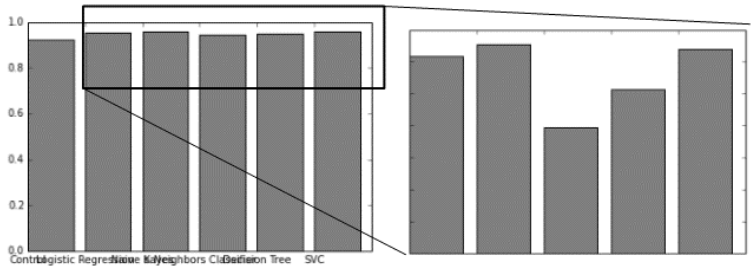


Рисунок 2. Курс фінансового менеджменту. Порівняння алгоритмів

Для попереднього дослідження ми використовували результати перших чотирьох пройдених тестів. Проте на практиці структура курсу може відрізнятися: тестів може бути більше або менше. Проведемо дослідження залежності точності моделі від кількості ознак, на якій вона побудована. Контрольну модель використаємо ту саму. Повні результати експерименту наведені у таблиці 2. Для наочності побудуємо діаграми – рисунок 3 і рисунок 4:

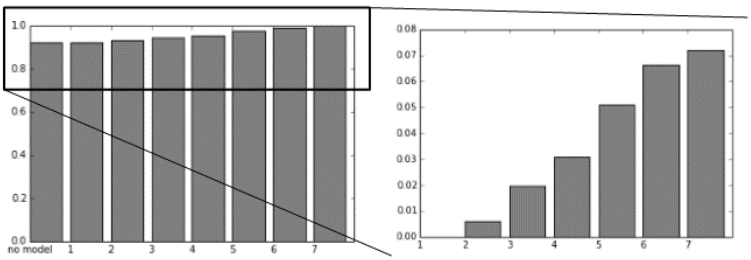


Рисунок 3. Курс історії. Якість прогнозу

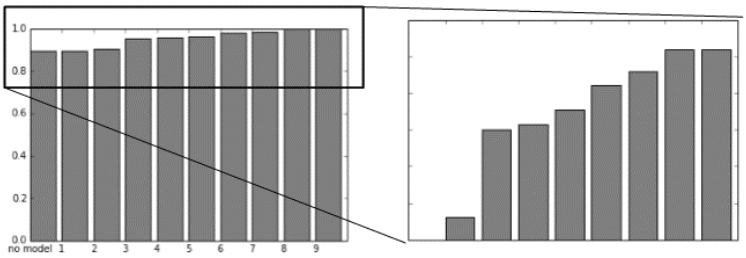


Рисунок 4. Курс фінансового менеджменту. Якість прогнозу

На рисунку 3 та рисунку 4 зображено точність прогнозу в залежності від кількості атрибутів моделі. Очевидно, що чим більше атрибутів, тим більше точність. Вона змінюється від несуттєвої(один атрибут) до абсолютної(максимальна кількість атрибутів). Для курсу історії точність росте монотонно, а для курсу фінансового менеджменту стрибкоподібно(за допомогою результатів перших трьох тестів можна досягти хорошої точності прогнозу). Отже, точність в залежності від кількості атрибутів може рости по-різному для різних курсів. Результати експерименту наведено у таблиці 2. При вирішенні поставленої задачі необхідно обирати ту кількість атрибутів для навчання моделі, скільки тестів користувач зміг вже пройти. Це забезпечить найкращий результат.

Таблиця 2. Дослідження точності роботи алгоритму для різної кількості атрибутів:

Кількість атрибутів	Курс історії. Точність моделі	Курс фінансового менеджменту. Точність моделі
Контрольна модель	0.924	0.893
1	0.924	0.893
2	0.930	0.905
3	0.943	0.953
4	0.954	0.956
5	0.974	0.964
6	0.990	0.978
7	0.996	0.985
8	-	0.997
9	-	0.997

5. Практичне застосування

Запропонований підхід цілком вирішує поставлену задачу, проте на практиці його важко інтегрувати у програмну систему та постійно використовувати. Була розроблена програмна реалізація, яка включала у себе вивантаження даних з системи Prometheus, алгоритм класифікації та презенацію результатів. Дослідним шляхом було виявлено, програма працює дуже повільно: опрацювання даних для одного курсу не менше 5 хвилин. Заміри витрат часу на операції показали, що слабкими місцями системи є: послідовна обробка даних у алгоритмі класифікації, а також фаза вивантаження даних з Prometheus. Відповідно було прийняте рішення про модифікацію алгоритму згідно рекомендацій вказаних у [8]. Внаслідок чого швидкість відгуку програми зменшилась до 1хв. Даний варіан є прийнятним у реалізації.

6. Висновки

Отже, маємо задачу прогнозування успішності студентів онлайн-курсів в залежності від кількості оцінок, які є наявні. Їх може бути довільна кількість. Поставлена задача зводиться до задачі бінарної класифікації, що може бути розв’язана багатьма методами, наприклад, логістичною регресією, яка стабільно добре працює при різних наборах даних. Чим більше оцінок студента відомо, тим більша точність прогнозу успішності. Варто відмітити, що прогнозування також має сенс при невеликій кількості відомих оцінок. Це доводить випадок курсу фінансового менеджменту, де вже при відомих результатах трьох тестів(із дев’яти можливих) можна з точністю 95% говорити, пройде студент курс чи ні. Такі результати залежать від конкретного набору даних конкретного курсу, і їх важко передбачити.

Список літератури

1. Прес-реліз проекту Prometheus [Електронний ресурс]. – 2014. – Режим доступу до ресурсу: http://prometheus.org.ua/prometheus-start/.

2. Курс фінансового менеджменту [Електронний ресурс] – Режим доступу до ресурсу: http://courses.prometheus.org.ua/courses/NAUKMA/101/2014_T2/about.

3. Курс історії [Електронний ресурс] – Режим доступу до ресурсу: http://courses.prometheus.org.ua/courses/KNU/101/2014_T2/about.

4. Clustering and Sequential Pattern Mining of Online Collaborative Learning Data. IEEE Transactions on Knowledge and Data Engineering / Perera, Kay, Koprinska та ін.], 2009. – (21). – С. 759–772.

5. Statistical classification [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Statistical_classification.

6. Schutt R. Doing Data Science / R. Schutt, C. O'Neil., 2014.

7. Andrew N. Machine Learning [Електронний ресурс] / N. Andrew – Режим доступу до ресурсу: https://www.coursera.org/learn/machine-learning.

8. Параллельные алгоритмы решения задач вычислительной математики. / Химич А.Н., Молчанов И.Н., Попов А.В., Чистякова Т.В., Яковлев М.Ф. – Киев: Наук. думка. – 2008, 247 с.

УДК 004.773.2

ТОМАШЕВСЬКИЙ В.М.,
БОБКО С.А.

ЗАСТОСУВАННЯ АРХІТЕКТУРИ PEER-TO-PEER МЕРЕЖ ДЛЯ ІДЕНТИФІКАЦІЇ СЛУХАЧІВ ЦІЛЬОВОЇ АУДИТОРІЇ

Досліджена та проаналізована сутність слухачів як невід'ємної частини аудиторії. Запропоновано підхід до реалізації інформаційної системи представлення цільової аудиторії та внутрішньої взаємодії її складових.

The investigated and analyzed the nature audience as an integral part of the audience. The approach to implementation of information systems to presenting the target audience and internal interactions of its components.

Вступ

Однією із актуальних проблем сучасного інформаційного суспільства наразі є ідентифікація слухачів та їх взаємодія між собою в межах однієї аудиторії з врахуванням наявної територіальної приналежності.

Аудиторія у загальному розумінні – це деяка група читачів, глядачів і слухачів, користувачів, а більш у вузькому розумінні – це лише певна кількість людей, яка об’єднана деяким комунікаційним інтересом. Можна цілком стверджувати що без врахування задоволення цих інтересів неможливо оцінити ефективність засобів масової комунікації, висновки та перспективи подальших досліджень в даному напрямку.

2. Сутність аудиторії

Аудиторія є однією із різновидностей маси − кількості людей, які певний час знаходяться у взаємозв'язку, наприклад чи то демонстрація, пікет, тощо. Достатньо двох сукупностей щоб описати сутність аудиторії: перша лише характеризує та не залежить від типу та засобу комунікації: наприклад соціально-демографічні характеристики (вікова, гендерна, соціальна, рівень освіти тощо) та психологічна. Другою групою ознак аудиторії є специфіка ставлення до системи загалом так і до її конкретних частин. Сукупність характеристик даної аудиторії реалізується через мотивації поведінки щодо джерел інформації, вимог, очікувань, пов’язаних зі змістом наданої інформації.

Під час вивчення масових інформаційних процесів аудиторії як об'єкту комунікаційного впливу необхідно не виключати з розгляду ряд соціальних та психологічних особливостей аудиторій. В цих процесах взаємодії ламаються межі декількох видів: фізичні – простір і час; соціальні – що визначаються ролями індивідів, які складають аудиторії; гносеологічні – які з’являються через відсутність чи неповноту абстрактного мислення; ідеологічні – результат засвоєння світогляду і стереотипів, що суперечать ідеологічним основам здійснюваного впливу; психологічні – у вигляді вже існуючих у свідомості аудиторії соціальних настанов, сформульованих думок, пересторог, пасивності уваги тощо. Сукупність цих факторів визначає висоту бар'єрів, які стоять між джерелом інформації та аудиторією [3].

Сутність аудиторії полягає не в тому, що вона складається з окремих людей, а в тому, що вона відображає їх взаємозв'язки між собою, із суспільством у цілому в процесі масового спілкування. Саме тому аудиторію також можна визначати як стійку сукупність людей, яка виникає на підставі спільності їх інформаційних потреб що зображено на рисунку 1. Чим глибше і краще будь−який орган масової комунікації відповідає на запити людей, тим ширше стала його аудиторія [1].

Рис. 1. Взаємодія аудиторії

Зазвичай виділяють реальну і потенційну аудиторію. Число людей, що мають фізичну можливість сприймати інформацію визначають потенційну аудиторію, а число читачів що практично реалізують цю можливість − саме реальну.

Цільова аудиторія визначається як сукупність людей, об'єднаних спільними соціальними, демографічними, культурними та економічними характеристиками, що є метою впливу матеріального або нематеріального продукту.

Множина індивідів, які отримують в різний час або одночасно певну, що має значимість, інформацію; особистості, включені в мережу реальних суспільних відносин і зв'язків становить масова аудиторія. Для неї здебільшого характерні ознаки неоднорідності та розосередженості.

Сукупність користувачів соціальних мереж з метою взаємодії, комунікації між собою (спілкування), пошуку інформації складають аудиторію соціальних мереж.

3. Постановка задачі у загальному вигляді

В умовах інформаційної глобалізації електронні соціальні мережі об’єднують людей довкола спільних інтересів та світогляду, утворюючи певну соціальну групу, − множину користувачів із задоволенням їхніх потреб. Незалежно від класифікації мережі чи то соціалізуючі, навчальні, специфічні або ділові та політичні. Швидке розповсюдження електронних соціальних мереж, часу користувачів, який вони проводять у мережах, доступності електронних пристроїв (комп’ютера, планшета, ноутбука, мобільного телефону) з підключенням до глобальної мережі Інтернет – все це зумовлює привабливість електронних соціальних мереж для розширення використання у більшості галузях діяльності людини.

Вже сьогодні соціальні мережі визначають важливу частину життя соціальної людини. У ході різних досліджень виявилося, що 95% постійних користувачів Інтернет користуються соціальними мережами, 77% тих, хто періодично відвідує Інтернет, користуються соціальними мережами, 590 тисяч користувачів заходять хоча б раз на місяць в 5 соціальних мереж. Не менш ніж 3.6 мільйонів користувачів заходять відразу в 4 мережі. Не менш 11 мільйонів заходять в 3 з них щомісяця. Аудиторія соціальних мереж зростає з величезною швидкістю, ще в 2012 році світова аудиторія соціальних мереж перевищила 1 млрд користувачів і не зупиняється у розвитку. Це говорить про те, що соціальні мережі стали місцем об'єднання людей за схожими інтересами, перетнувши кордону простого інструменту для обміну інформацією [2].

Середовище соціальних мереж представляє чималий інтерес для всіх

фахівців, що професійно вивчають людину і різноманітні види людської активності. Діяльність людини у віртуальному середовищі стала предметом вивчення соціологів, психологів, маркетологів, політологів та інших фахівців.

Дана стаття пропонує підхід до реалізації системи, який дозволить інформаційно описати (частково або навіть повністю) множину цільової аудиторію, а також внутрішні взаємозв'язки слухачів. Також основною вимогою є те що слухачі територіально пов'язані з деяким місцем: торговий центр, будинок, громадський транспорт тощо.

Метою роботи є скорочення часу необхідного для проведення ідентифікації особистостей в групах людей за рахунок впровадження мобільного застосунку.

Об’єкт дослідження є процес ідентифікації слухачів.

4. Розв’язання задачі

Запропонований підхід базується на використанні архітектури peer-to-peer (P2P або пірингова) мережі. Peer-to-peer – це мережа, яка ґрунтується на принципі рівності учасників, тобто в базовій peer−to−peer мережі відсутні поняття клієнтів та серверів, а лише рівнозначні вузли, які одночасно функціонують як клієнти та сервери по відношенню один до одного. Модель мережевої взаємодії відрізняється від клієнт−серверної архітектури, в якій зв'язок відбувається лише між клієнтами та центральним сервером (схематично зображено на рисунку 2). Така організація дозволяє зберігати працездатність мережі при будь−якій конфігурації доступних їй вузлів. Проте існують P2P мережі, які мають в архітектурі сервери, але їх роль полягає вже не у наданні сервісів, а у підтримці інформації з приводу сервісів, що надаються клієнтами мережі [4], [5].

Природа технології P2P робить її придатною для забезпечення співпраці між користувачами. Це може бути обмін повідомленнями, онлайн ігри, сумісна робота над документами тощо.

Пірингові мережі розрізняють за ступенем централізації:

- чисті пірингові мережі - вузли є рівними. Кожен вузол виконує роль як сервера, так і клієнта. Не існує центрального сервера, що керує мережею.

- гібридні пірингові мережі мають центральний сервер, що зберігає інформацію про вузли та відповідає на запити відносно цієї інформації. Вузли забезпечують мережу ресурсами (центральний сервер їх не має), повідомляють сервер про наявність цих ресурсів, надають ресурси іншим вузлам. Існує велика кількість таких систем, наприклад такі як Ares, FileSpree тощо.

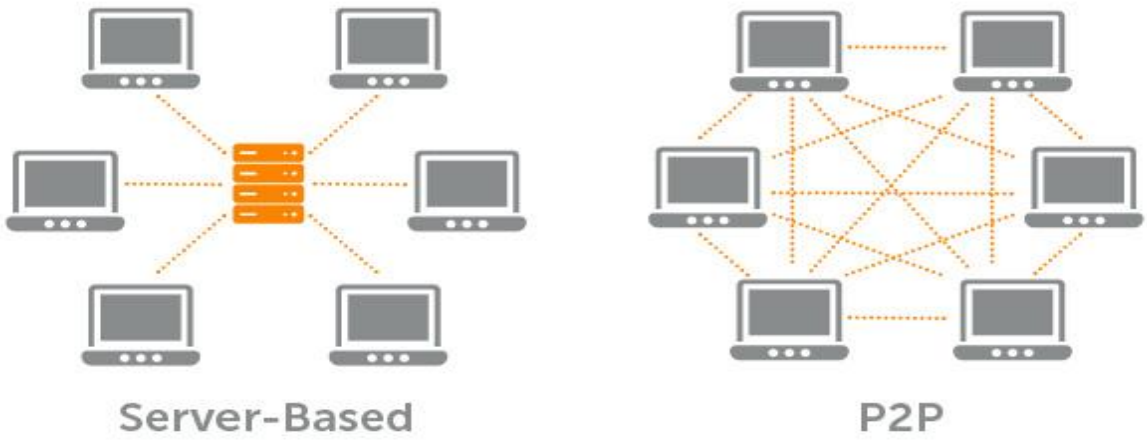


Рис. 2. Відмінність архітектур Server-Based та P2P

Коли вузол розміщує у мережі деякий ресурс, він обробляє його зміст та обчислює значення hash−функції, яка буде ідентифікувати ресурс у мережі.

Hash−функція обирається таким чином, щоб унікальні номери учасників та обчислені ключі ресурсів набували значень з однієї множини. Обчисливши значення hash−функції, вузол шукає інший модуль, ID якого близький до знайденого ключа. Знайшовши такий вузол, розміщувач ресурсу передає свою IP−адресу та ключ, які знайдений модуль зберігає у себе. Таким чином, клієнт мережі, який потім хоче завантажити ресурс, знаючи з деяких джерел його ключ, намагається отримати інформацію про знаходження цього ресурсу в тих учасників мережі, унікальний номер яких близький до ключа ресурсу, який необхідно знайти.

Проаналізовані алгоритми пошуку файлів дозволяють користувачу, знаючи їх назву, частину назви або опис, швидко знайти потрібну інформацію, не перевантажуючи мережу численним «сліпими» пересиланнями. Необхідно зазначити, що, на відміну від мереж з клієнт−серверною архітектурою, задача ефективного пошуку в піринговых мережах − відкрита дослідницька проблема. У подальшому можлива розробка структури піринговой мережі із врахуванням географічного положення вузлів. Це дозволить побудувати систему пріоритетів, що, перш за все, призведе до швидшого завантаження потрібної інформації, а для користувачів Інтернету з розподіленим трафіком й до економії коштів.

У залежності від обраного методу представлення слухача і базується підхід до розв’язку поставленої задачі. Для інформаційного представлення аудиторії доцільно використати архітектуру peer-to-peer мережі у поєднанні з мобільними телефонами для представлення конкретного слухача аудиторії [6], адже він вже став невід'ємною складовою і неодмінним супутником нашого життя. Мобільний телефон дозволить вирішити наступні речі:

1. Побудова ідентифікаційного токену, що дасть змогу ідентифікувати та автентифікувати слухача.
2. Поєднання слухача з іншими членами аудиторії.
3. Інформаційне представлення та реалізація соціальної аудиторії.
4. Задача локальності при використанням безпровідних технологій близького зв'язку (Wi−Fi, Bluetooth тощо).

Використання мобільних телефонів із спеціалізованим програмним забезпеченням дозволить без додаткових затрат на апаратні засоби організовувати інформаційні аудиторії та відображати їх діяльність.

Приведемо більш наочний приклад на вже існуючому проекті соціальної мережі Badoo. Badoo – це соціальна мережа знайомств що дозволяє знаходити нових друзів і спілкуватися з ними. Мережа одна з перших реалізувала цікавий сервіс – від дозволяє побачити присутніх поруч з вами користувачів мережі і в залежності від сценарію та взаємної згоди сторін – зустрітись.

Повернемось до розв'язку поставленої задачі. Згідно з P2P−архітектурою, навіть вже один слухач A зможе представляти обрану ним самим цільову аудиторію X, яка на початку буде складатись лише з одного елемента $X = \{A\}$. Якщо в деякі проміжки часу з'являться слухач B та політолог C з такою ж цільовою аудиторією X то в результаті отримаємо цільову аудиторію {A, B, C}. І знову з P2P−архітектури випливає, що кожен учасник є самостійною одиницею і зможе отримати інформацію про мережу (аудиторію), її членів та взаємозв'язок з кожним. Особливо інформація щодо мережі може бути цікава політологу на чому і акцентувався початок статті.

Система, яка розроблена на даному підході включатиме переваги: масштабованості, децентралізованості, самодостатності та мобільності.

До недоліків системи можна віднести: складність в управлінні та надлишковий обмін інформацією.

5. Висновки

Запропонована мережа дозволить одночасно представити мережу слухачів, які географічно прив'язані до деякої місцевості, але знання координат кожного слухача при цьому не є необхідністю.

Реалізація подібної архітектури дозволить розроблювати гнучкі системи, які допоможуть інформаційно описувати множини аудиторій для отримання аналітичних даних про слухачів.

6. Список літератури

1. Литвин Ю. Структуризація аудиторії мас−медіа в системі політичної комунікації [Електронний ресурс] / Ю. Литвин // Соціально−економічні проблеми і держава. − 2011. − Вип. 2 (5)

2. Рябічев В. Л. Аудиторія українського сегмента всесвітньої мережі (друга половина 2013 р.) [Текст] / В. Л. Рябічев, В. В. Литвиненко // Наукові записки Інституту журналістики, 2014. т.Т. 56.-С.140-145

3. Олесюк Н., Лебеденко Л. Використання електронних соціальних мереж у соціально−педагогічній роботі зі школярами

4. Орлова М., Сапсай Т., Спринсян Я. - Алгоритми пошуку даних в peer−to−peer мережах. // IX конференція молодих вчених ПМК-2017

5. Verstrynge J. Practical JXTA II. Cracking the P2P puzzle. Netherlands, DawningStreams Publ., 2010. 271 p.

6. Шитько А., Пацей Н. Использование протокола peer-to-peer для защищенного обмена данными // Труды БГТУ. 2015 №6. Физико-математические науки и информатика. С. 162-165

УДК

БОРИСОВ М.О.
ПРОСКУРА С.Л.

СОЦІАЛЬНО-ІНФОРМАЦІЙНИЙ СЕРВІС "SLOVAMI"

У даній статті розглядається програмний продукт - соціально-інформаційний сервіс "Slovami". Він надає змогу зареєстрованому користувачу не тільки публікувати свої авторські тексти , наукові праці або дослідження , а й отримувати постійний пасивний фінансовий дохід, використовуючи свої матеріали, як майданчик для розміщення контекстної реклами Google AdSense. Розміщеною та структурованою інформацією по категоріям, може скористатись гість або читач - особа, яка пройшла реєстрацію і обрала відповідний тип профілю, має доступ до особистої сторінки з можливістю її заповнення та редагування, може коментувати статі, а також має можливість відмічати цікаві статі зі зберіганням їх в колекцію. Для оптимізації під пошукові системи використовуються сучасні SEO-технології. Під час розробки сервісу «Slovami» аналогічних зарубіжних та вітчизняних проектів з подібною концепцією , ідеєю та функціоналом не виявлено.

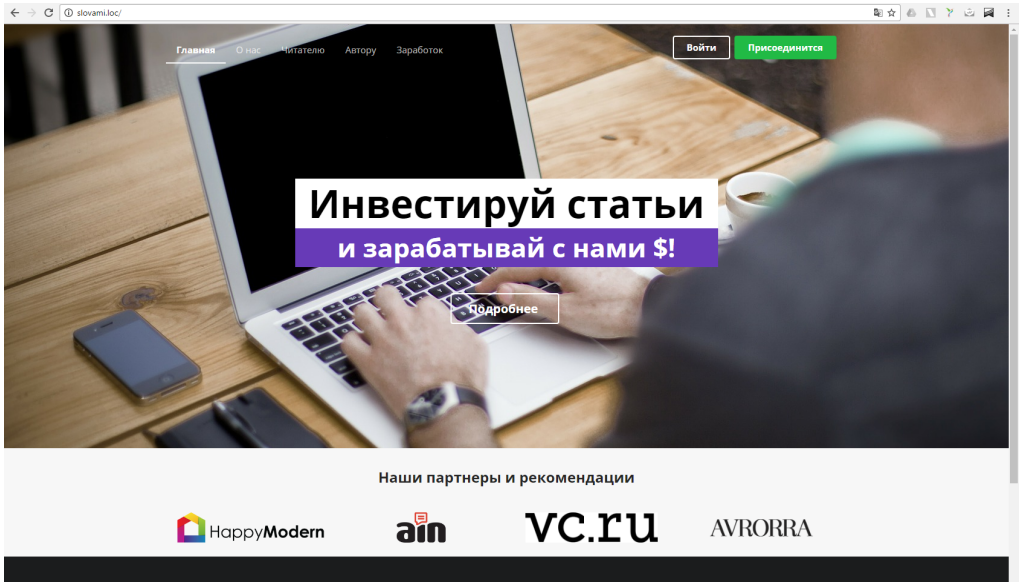
This article discusses the software - social and information services "Slovami". It gives possibility to the authorized user not only to publish the authorial texts, scientific works or researches, but also to get a permanent passive financial profit, using the materials as ground for placing of context advertisement of Google AdSense. By the placed and structured information for to the categories, a guest or reader can avail - a person that passed registration and chose the corresponding type of profile has an access to the personal page with possibility of her filling and editing, can comment post, and also has the opportunity to mark interesting post with storage of them in collection. For search engine optimization using modern SEO technology. During developing the service «Slovami» similar foreign and domestic projects with a similar concept, idea and functionality not found.

1. Вступ

На даний момент створюються та існують десятки тисяч різних тематичних інтернет ресурсів та інформаційних проектів Проаналізувавши найрізноманітніші тематики та направлення таких інформаційних інтернет проектів, можна виявити, що основна частина з них використовує та розміщує неунікальний, скопійований текстовий та інший контент, в кращому випадку зробивши рерайт - перефразування існуючого авторського тексту. Відповідно дуже мала кількість інформаційних інтернет ресурсів в своїх тематичних нішах можуть дозволити собі придбання або створення дійсно якісних, орієнтованих на свою аудиторію унікальних статей. Все це можна пояснити багатьма факторами: високою собівартістю якісних унікальних текстів, нестачею авторів та спеціалістів в тематичних рубриках та їх незацікавленістю в створенню та розповсюдженні своїх авторських текстів широкому загалу. Як правило професійні автори та копірайтери отримують винагороду за свою інтелектуальну роботу – одноразово, передаючи всі права на текст замовнику та інтернет-проекту в цілому.

У даній статті розглядається програмний продукт - соціально-інформаційний сервіс "Slovami". Він має на меті і надає змогу не тільки публікувати свої авторські тексти , наукові праці, а й отримувати постійний пасивний фінансовий дохід автору, використовуючи свої матеріали як майданчик для розміщення контекстної реклами. Автором може стати будь-який зареєстрований користувач сервісу. Сам автор безпосередньо зацікавлений в якості та правильності оформлення його контенту, чим якісніший текст по темі і чим більша відвідуваність в його публікації тим більший дохід в цілому він отримає.

Сервіс «Slovami» надає авторам та новачкам ряд матеріалів та комплекс рекомендацій по оформленню текстів автора для підвищення пошукових позицій та позитивного ставлення читачів до самих публікацій. Головна сторінка соціально-інформаційного сервісу «Slovami» показана на рис.1.



Ри.1 - Головна сторінка соціально-інформаційного сервісу «Slovami»

2. Призначення та цілі розробки

В цілому основним призначенням розробки є підтримка та публікація якісних та унікальних авторських статей широкому колу читачів. Даний проект надає змогу зареєстрованому користувачу не тільки опублікувати свої авторські тексти , наукові праці або дослідження , а й отримувати постійний пасивний фінансовий дохід, використовуючи свої матеріали, як майданчик для розміщення контекстної реклами Google AdSense. Розміщеною та структурованою інформацією по категоріям, може скористатись гість або читач - особа, яка пройшла реєстрацію і обрала відповідний тип профілю, має доступ до особистої сторінки з можливістю її заповнення та редагування, може коментувати статі, а також має можливість відмічати цікаві статі зі зберіганням їх в колекцію.

Основними цілями розробки є полегшення процесу реєстрації та авторизації через соціальні мережі, публікації авторського контенту, просування авторських статей широкому загалу, оптимізація та полегшення процесу редагування і модерації авторських статей, збільшення якісного та унікального авторського контенту по тематичним рубрикам в мережі, отримання пасивного фінансового доходу авторами від публікації власного контенту, використання комплексної пошукової SEO-оптимізації контенту сервісу.

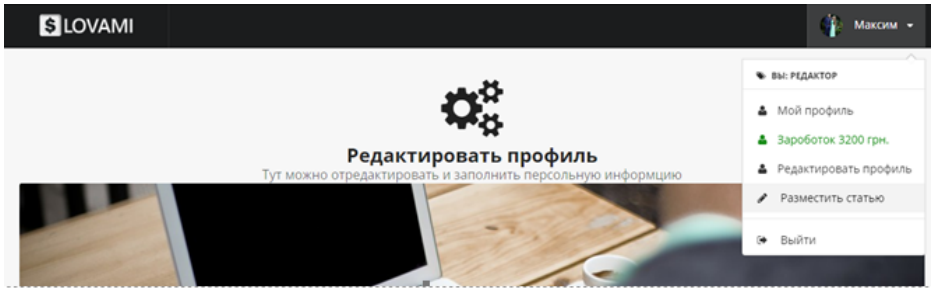


Рис.2 -Основні можливості автора в соціально-інформаційному сервісі «Slovami»

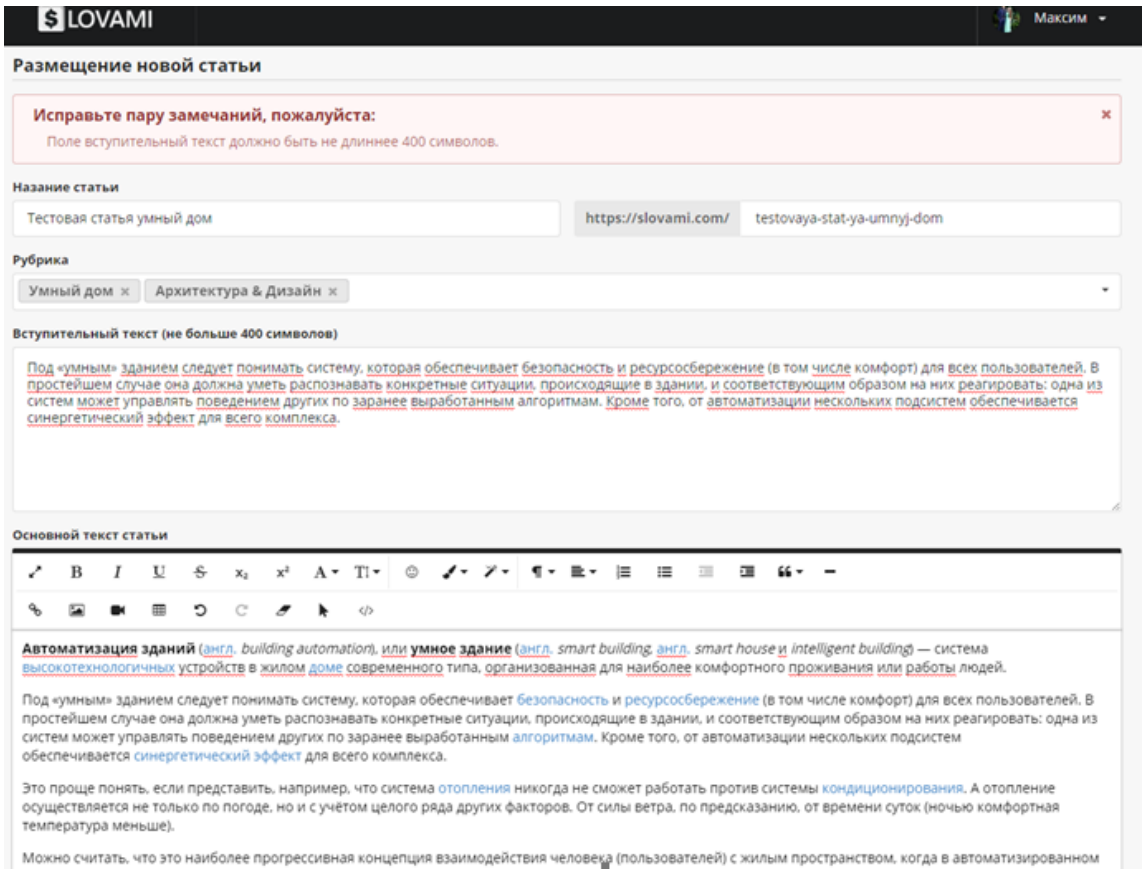


Рис.3. - Сторінка та функціонал створення та публікації власного авторського тексту

3. Змістовна постановка задачі

Відповідно до ідеї соціально-інформаційного сервісу автори публікацій отримують пасивну грошову винагороду за розміщення контекстної реклами у власних статях. Загальна сума прибуток від контекстної реклами ділиться порівну 50/50, з яких половину отримує сервіс, а інша половина розділяється між всіма авторами. Отже, сума грошових винагород автора розраховується відсотком від ½ загального прибутку сервісу, який прямопропорційно залежить від кількості переглядів публікації (статі) автора. Дані будуть синхронізуватись з системою контекстної реклами Google AdSense.

Також передбачено, що у сервісі будуть вестись статистка переглядів та грошових винагород для кожної публікації. Це створюється для того, щоб автор публікацій, міг побачити кількість переглядів його статті, суму винагороди за певний період та оцінити ефективність та якість тексту.

4. Математична постановка задачі

Маємо дві задачі:

– задача розрахунку суми винагороди автора публікації;

– задача ведення статистики кількості переглядів статей.

Для розрахунку суми винагороди автора використовуємо наступні параметри:

– M – загальна сума винагороди інформаційного сервісу від контекстної реклами;

– U – загальна кількість переглядів усіх публікацій за певний період;

– A – кількість переглядів статей автора

Формула (1) розрахунку суми винагороду, яку отримує автор за розміщення своїх статей :

$$S = \frac{\left(\frac{M}{2}\right) \times \left(\frac{A}{U}\right)}{100} \quad (1)$$

5. Обґрунтування методу розв’язання

Задача ведення статистики участі у акційній пропозиції є задачею пошуку наближеного розв'язку, яку можна розв’язати за допомогою методу найменших квадратів [7].

Метод найменших квадратів – метод знаходження наближеного розв'язку надлишково-визначеної системи. Часто застосовується в регресійному аналізі. На

практиці найчастіше використовується лінійний метод найменших квадратів, що використовується у випадку системи лінійних рівнянь. Зокрема важливим застосуванням у цьому випадку є оцінка параметрів у лінійній регресії, що широко застосовується у математичній статистиці і економетриці.

6. Опис методів розв'язання

Основною математичною задачею, яку розв'язує дана система, є задача ведення статистики відвідування публікації автора. Розглянемо довільну ситуацію. З публікацією автора Z за час публікації t ознайомились читачі. У той самий час з публікацією X опублікована стаття з таким самим часом існування, що й у публікації Z. Потрібно прорахувати тенденцію ведення статей та порівняти результати відвідування публікацій (статей) двох авторів. Спрогнозуємо тенденцію росту популярності статі через початок опублікування матеріалу. Приведемо графік для наглядності на рис.4

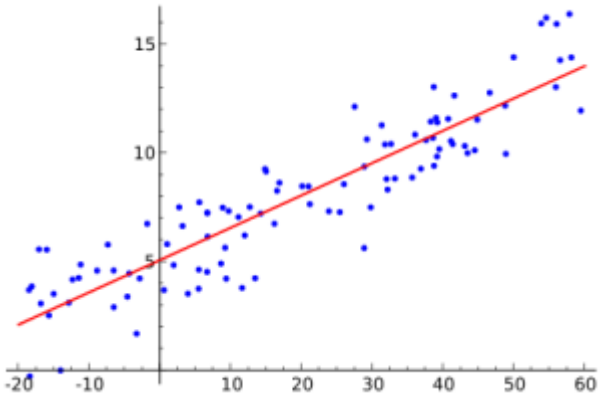


Рис. 4 – Графік відображення потенційної тенденції росту інтересу на відвідування публікації (статі).

Кроки алгоритму методу найменших квадратів:

Крок 0. Перевіряється умова ідентифікованості для кожного значення.

Крок 1. Здійснюється перехід від структурної форми моделі до зведеної.

Крок 2. Дається оцінка параметрів кожного рівняння зведеної форми моделі МНК.

Крок 3. Розраховуються оцінки параметрів рівнянь.

7. Засоби розробки

Для розробки серверної частини веб-сервісу була обрана мова PHP версії 7.0, PHP-фреймворк Laravel, а в якості бази даних була використана MySQL версії 5.5. Проект складається с ряду модульних рішень які відповідають окремим задачам сервісу «Slovami». Основні модулі проекту: авторизація та реєстрація, авторизація за допомогою соціальних мереж (Facebook, Google+, Twitter та ін.), користувачі та їх ролі, рубрики, публікація статей, інтеграція с Google AdSense та Google Аналітикою, адміністративна частина та інші менш важливі модульні рішення.

8. Огляд наявних аналогів

На сьогодні, головним конкурентом сервісу являється російський проект Upages (https://upages.io) від розробників популярної в свій час платформи uCoz, який перебуває на стадії закриття. Сервіс Upages також дозволяє кожному спробувати себе в ролі автора статей, оглядів, перекладів та інтерв'ю на тему лайфстайлу та інших рубрик і використовувати власні опубліковані матеріали, щоб отримувати пасивний дохід від розміщення контекстної реклами, але відмінність в підході до оптимізації під пошукові системи текстів та в алгоритмі нарахування винагороди яка є не досконалою і не відповідає . Основні відмінності сервісу від проекту «Slovami»:

- автори сервісу Upages отримують 80% від рекламних доходів власних публікацій, яка розраховується від кількості переглядів конкретної публікації автора на тисячу показів;
- публікації майже не оптимізовані під пошукові запити, не використовуються сучасні способи і методи SEO-оптимізації;

- сервіс на даний момент має непрозорий та недосконалий алгоритм нарахування суми винагороди авторам;
- неорієнтований на навчання нових авторів та підвищення ефективності існуючих авторів до написання якісних і правильно оптимізованих текстів під пошукові системи;
- сервіс не оптимізований на взаємодію з користувачем та для мобільними пристроями.

Під час розробки сервісу «Slovami» аналогічних зарубіжних та вітчизняних проектів з подібною концепцією та ідеєю і функціоналом не виявлено.

9. Висновки

Основною метою соціально-інформаційного сервісу «Slovami» є збільшення унікального тематичного контенту та мотивація авторів моделлю пасивної фінансової винагороди за свою інтелектуальну роботу. Всі права на текст публікації залишається за автором, що також є важливою складовою та юридичним аспектом проекту.

Модель пасивного доходу від власних текстів привертає увагу найрізноманітніших авторів, тематичних спеціалістів, професійних копірайтерів-фрілансерів та всіх тих інтернет користувачів, які можуть поділитися цікавими публікаціями, що в свою чергу збільшить кількість якісного контенту в мережі.

Якісний контент та комплекс оптимізації забезпечує веб-ресурсу вигідні позиції в пошуковій видачі та впливає на позитивне ставлення читачів до проекту, а велика читацька аудиторія привертає увагу великих рекламодавців.

Список літератури

1. Копірайтинг [Электронний ресурс] / Режим доступу: https://uk.wikipedia.org/wiki/Копірайтинг
2. Автор [Электронний ресурс] / Режим доступу: https://uk.wikipedia.org/wiki/Автор
3. Соціальна мережа [Электронний ресурс] / Режим доступу: https://uk.wikipedia.org/wiki/Соціальна_мережа
4. Контекстна реклама [Электронний ресурс] / Режим доступу: https://uk.wikipedia.org/wiki/Контекстна реклама
5. Developer Google AdSense [Электронний ресурс] / Режим доступу:https://developers.google.com/adsense/?hl=ru
6. PHP документация [Электронний ресурс] / Режим доступу: http://php.net/manual/ru/
7. Laravel документация [Электронний ресурс] / Режим доступ: https://laravel.ru/docs/v5

УДК 004.432+004.9

ВАСИЛЕНКО В.Г.,
ШИРІЙ В.В.,
БАКЛАН І.В.

СУЧАСНІ ПАРАДИГМИ ПРОГРАМУВАННЯ: ЙМОВІРНІСНЕ ПРОГРАМУВАННЯ

В статті представлено новітню програмну парадигму – ймовірнісне програмування. Ця парадигма дозволяє використовувати все складніші ймовірнісні моделі для забезпечення все новіших інструментів для розвитку та створення штучного інтелекту. Ми виділили окреме місце для ймовірнісного програмування серед інших програмних парадигм та охарактеризували особливості цієї нової парадигми.
Ключові слова: ЙМОВІРНІСНЕ ПРОГРАМУВАННЯ, ПАРАДИГМА ПРОГРАМУВАННЯ, ЙМОВІРНІСНЕ МОДЕЛЮВАННЯ

The paper presents the newest software paradigm - probabilistic programming. This paradigm allows all complex probabilistic models to ensure all newer tools for the development and creation of artificial intelligence. We have allocated a separate space for probabilistic programming includes programming paradigms and describe some realization of this new paradigm.
Keywords: PROBABILISTIC PROGRAMMING, PROGRAMMING PARADIGMS, PROBABILISTIC MODELING

1. Вступ

На сьогоднішній день стає поширеним застосування ймовірнісних моделей, які використовуються для створення сучасного штучного інтелекту, у прикладній статистиці чи в когнітивній науці. Це пояснюється тим, що вони пов’язані з роботою над ймовірностями та їх ймовірнісними висновками [1,2]. Однак, ймовірнісні моделі мають тенденцію до збільшення їхньої складності. Тому потрібно створювати нові інструменти для забезпечення нового комплексного підходу до ймовірнісного представлення моделей. І саме ймовірнісні мови програмування це забезпечують. Мови дозволяють створювати засоби для опису складних ймовірнісних розподілів та реалізують виконання ефективного ймовірнісного висновку для довільної комп’ютерної програми.

Ймовірнісні мови програмування, в їх прості формі, розширюють добре відомі детерміновані мови програмування з примітивними конструкціями для випадкового вибору (random choice) [2]. Проте з часом, відбулося створення нових інструментів для ймовірнісного виводу та зародження нових складніших ймовірнісних моделюючих програм. Наявність великої кількості ймовірнісних мов програмування змусило прийти до думки, що існує певна парадигма програмування, так зване, ймовірнісне програмування.

2. Парадигма ймовірнісного програмування

На Рис.1 зображено «Систематику парадигм програмування», яка дає змогу виділити всі основні програмні парадигми [3]. Оскільки, програмна парадигма являє собою підхід до програмування на основі математичної теорії або послідовного набору принципів, кожна з парадигм підтримує ряд понять, яка робить її кращою для вирішення певного виду проблем. Наприклад, об’єктно-орієнтоване програмування (ООП) є найкращим для вирішення проблем з великою кількістю пов’язаних з абстракцією даних, що організовані в ієрархії, а логічне програмування – для перетворення або управління над складними символічними

структурами відповідно до логічних правил.

В порівнянні з оригінальною класифікацією з Рис. 1, на Рис. 2 ми виділили ймовірнісне програмування серед інших як новітню парадигму програмування. Цю парадигму програмування можливо розділити на п’ять іншим парадигм, які розширюють основне ймовірнісне програмування. А саме: ймовірнісне логічне, функціональне ймовірнісне, раціональне ймовірнісне, імперативне ймовірнісне та об’єктно-орієнтоване ймовірнісне. Блоки ймовірнісного програмування зображено у вигляді двох типів розподілу: лівосторонню та правосторонню. Так, лівосторонні блоки відносяться до більш функціональних парадигм, а правосторонні – до імперативних. Особливості кожної з цих парадигм відображаються у відповідній наведеній програмній мові.

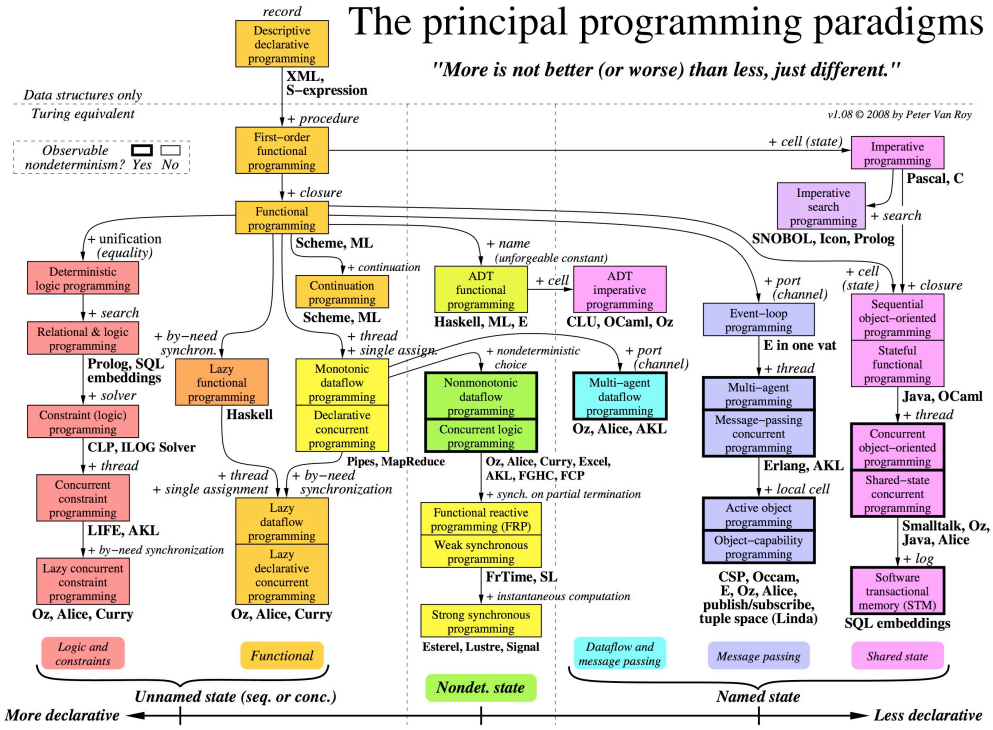


Рис.1 – Класифікація парадигм програмування [4]

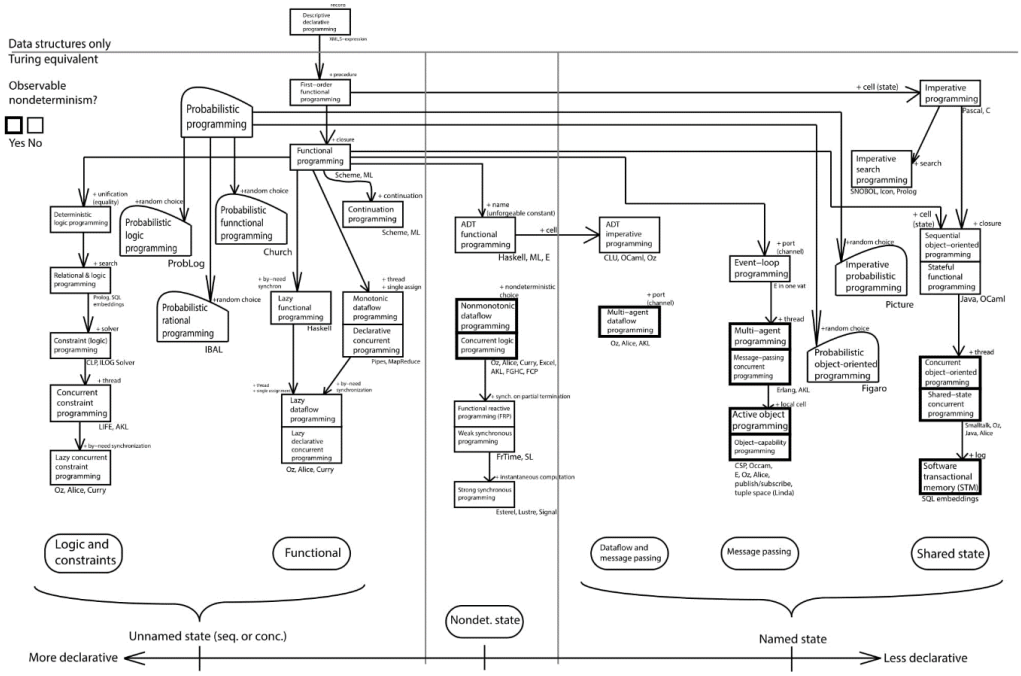


Рис. 2 – Місце ймовірнісного програмування серед інших парадигм програмування [2]

В ймовірнісних мовах програмування, наприклад в ProbLog, PRISM, ICL, використовується семантика розподілу для нескінченого набору випадкових змінних та загальний клас розподілів [3]. Однак, існує інші дві популярні семантики, засновані на основі ряду незалежних випадкових змінних (кінцева множина булевих випадкових змінних), що перетворюються в ймовірнісні факти (probabilistic facts), та незалежних ймовірнісних виборах (probabilistic choices).

Скоріше за все, найчастішим прикладом використання семантики розподілу є використання кінцевої множини булевих випадкових змінних, які, як передбачається, будуть незалежними. Тоді, ймовірнісними факт відповідає булевій випадковій змінній, яка має значення «true» з деякою ймовірністю *p* і «false» з ймовірністю *1-p*.

Для простоти моделювання та забезпечення безкінечної обчислювальної множини ймовірнісних фактів, в ICL та ProbLog використовуються небазисні ймовірнісні факти (non-ground probabilistic facts) для визначення множини випадкових величин. Всі базисні випадки такого факту є незалежними один від одного та розділяють ту ж саму ймовірнісне значення.

Ймовірнісні факти (також зустрічається назва «бінарні перемикачі») досить виразні, щоб представляти широкий спектр моделей, в тому числі байесових мереж, марковських мереж та прихованих марковських моделей. Проте, для простоти моделювання, часто буває зручніше використовувати багатозначні випадкові величини замість двійкових одиниць. Тому існують різні приклади використання ймовірнісного вибору. А саме: ймовірнісні альтернативи самостійного вибору логіки (ICL), мульти-арні випадкові вимикачі (PRISM), ймовірнісні положення стохастичних логічних програм (SLP) та ін..

Ймовірнісні мови логічного програмування і їх реалізації, як правило, не залишають вибір користувачу. У ICL, ProbLog, LPAD кожний небазисний ймовірнісний факт безпосередньо відповідає випадковій змінній в межах можливого світу, тобто кожна поява такого факту має таке ж значення істинності, тому факт можна зберегти чи мемоізувати (memoization). Окрім того, така ймовірність факту береться і обчислюється лише один раз, незалежно від того, скільки разів вона зустрічається в програмі. В AILog2, ProbLog1, ProbLog2 використовується стохастична мемоізація, яку, при необхідності, можливо

відключити для автоматичного додавання унікального ідентифікатора для кожного входження однієї і тієї ж випадкової змінної.

В ймовірнісних раціональних програмних мовах, програма визначає ситуацію, з якою зустрічається агент; програмна оцінка обчислюється за таким принципом: що робив або чому повірив би раціональний агент в певній заданій ситуації. Раціональне програмування поєднує в собі переваги декларативних уявлень з особливостями мов програмування, таких як модульність, композиційність і систему типів.

IBAL (потрібно читати як "eyeball") визначається розробниками мовою інтегрованого баєсового агента (Integrated Bayesian Agent Language). Як випливає з назви, вона інтегрує різні очікувані ймовірнісності на основі раціональної поведінки, в тому числі ймовірнісного міркування і оцінки байєсівських параметрів.

На відміну від IBAL, в Figaro вирази комбінуються в складніші вирази. І оскільки, моделі є об’єктами, вони можуть мати властивості. Ці властивості включають умови, що дозволяють визначити дані спостережень модульним способом. Властивості також полегшують висловити ситуації, в яких об’єкти пов’язані між собою різними способами, що складно в функціональних мовах. Ці властивості також включають умови, які дозволяють Figaro можливість добре виражати ненаправлені та породжуючі моделі. Оскільки, одним з ключових особливостей об’єктно-орієнтованого програмування є наслідування, Figaro зберіг цю можливість – від базових класів можливо отримати більш специфічно-необхідніші на даний момент класи.

3. Основні результати та висновки.

Ймовірнісне програмування є швидко розвиваючою областю досліджень, про що свідчить створення багатьох ймовірнісних мов програмування та примітивів, які були введені протягом останніх декількох років. В цій статті було спробовано визначити місце і роль ймовірнісного програмування серед інших сучасних парадигм програмування. Були виявлені головні особливості парадигми ймовірнісного програмування та наведені програмні мови, що представляють їх.

Звісно, з плином часу та впровадженням ймовірнісного програмування, потрібно буде розвивати та розширювати область поширення даної парадигми програмування. В майбутніх публікаціях планується зробити довести до кінця порівняльний аналіз існуючих реалізацій ймовірнісних мов. Автори планують реалізувати самостійну ймовірнісну мову – ProPL, модель інтерпретатора якого буде реалізована в сучасній версії ANSI Common Lisp [5].

Список літкратури

1. Баклан І.В. Сучасні засоби ймовірнісного програмування / І.В.Баклан, В.Г.Василенко, В.В.Ширій // Матеріали III Міжнародної науково-технічної Internet-конференції «Сучасні методи, інформаційне, програмне та технічне забезпечення систем управління організаційно-технічними та технологічними технологічними комплексами», 23 листопада 2016 р. [Електронний ресурс] – К: НУХТ, 2016 р. – 286 с.
2. Василенко В.Г. Особливості нової парадигми програмування – ймовірнісного програмування / В.Г.Василенко, В.В.Ширій, І.В.Баклан // Матеріали V Всеукраїнської конференції науково-практичної конференції студентів, аспірантів та молодих вчених з автоматичного управління, присвяченої дню космонавтики / Під ред. Г.В. Рудакової та ін. – Херсон: ПП Вишемирський В.С., 2017. – С. 31-33.

3. Goodman N. D. The principles and practice of probabilistic programming //ACM SIGPLAN Notices. – 2013. – Т. 48. – №. 1. – С. 399-402.
4. Van Roy P. et al. Programming paradigms for dummies: What every programmer should know //New computational paradigms for computer music. – 2009. – Т. 104.
5. Грэм П. ANSI Common Lisp. – СПб.: Символ-Плюс, 2012. – 448 с.

УДК 658.517

ВЕЦКО В.І.

ПРОГНОЗУВАННЯ СТАНУ БАТАРЕЇ ЕЛЕКТРИЧНИХ ТРАНСПОРТНИХ ЗАСОБІВ ДЛЯ ОЦІНКИ ЖИТТЄЗДАТНОСТІ ПРИ КАРШЕРИНГУ

Каршерінг компанії впроваджують електричні транспортні засоби (EVs) в свій автопарк. Однак дані свідчить про те, що в даний момент з використанням електромобілів не вдається досягти задовільної комерції. Потенційною причиною цього є більш нагружене використання транспортного засобу, що характерно для короткочасної оренди автомобілів, а також їх наслідки для стану батареї (SoH). У цій статті ми прогнозуємо SoH двох однакових електромобілів, що використовуються в різних практиках автомобільного обміну. Для цього ми використовуємо дані отримані від зарядних станцій і різних датчиків EV. Отримані результати показують, що розуміння водіння користувачів та поведінки зарядки може служити цінним орієнтиром для системи короткочасної оренди автомобілів. Зокрема, результати прогнозування показують, що в той момент, коли батарея електромобіля досягає теоретичного кінця життя може відрізнятися на чверть часу, якщо транспортні засоби експлуатуються в різних умовах.

Ключові слова: ПРОГНОЗУВАННЯ СТАНУ; ОРЕНДА АВТОМОБІЛІВ; ЕЛЕКТРИЧНИЙ ТРАНСПОРТНИЙ ЗАСІБ (EV); ВОДІННЯ І ПОВЕДІНКА ЗАРЯДКИ; СТАН ЗДОРОВ'Я БАТАРЕЇ (SoH); ДЕГРАДАЦІЯ БАТАРЕЇ; СПІЛЬНА ЕКОНОМІКА

Car-sharing companies are introducing electric vehicles into their fleet. At this point shared electric vehicles systems are failing to reach satisfactory commercial viability. A potential reason for this is the effect of higher vehicle usage, which is characteristic of car sharing, and the implications on the battery's state of health (SoH). In this paper, we forecast the SoH of two identical EVs being used in different car-sharing practices. For this purpose, we use real life transaction data from charging stations and different electric vehicles sensors. The results indicate that insight into users' driving and charging behavior can provide a valuable point of reference for car-sharing system designers. In particular, the forecasting results show that the moment when the battery of an electric vehicle reaches its theoretical end of life can differ in as much as a quarter of the time when vehicles are shared under different conditions.

Keywords: FORECASTING THE STATE; RENT A CAR; ELECTRIC VEHICLE (EV); DRIVING BEHAVIOR AND CHARGING; HEALTH BATTERY (SOH); DEGRADATION BATTERY; JOINT BUSINESS

1. Вступ

В останні роки відбулося переосмислення особистої мобільності. Є дві основні мотивації для цього. По-перше, після десятиліть використання автомобілів, ми досягли точки, де транспорт відповідає за 23% світових викидів. Очевидно, що теперішня система мобільності є нестійкою в її нинішньому вигляді і що необхідні нові, більш стійкі та енергоефективні рішення. По-друге, дослідження показали, що особисті транспортні засоби використовуються в середньому близько години в день [1,2]. Припарковані більшу частину часу, вони займають цінний простір для суспільства. Цей ефект особливо помітний в міських районах, де населення продовжує зростати. За даними Всесвітньої організації охорони здоров'я, 54% від загального світового населення проживає в міських районах. Тільки в Європі, в міських районах проживає понад дві третини населення Європейського Союзу.

Одним з рішень є спільне використання автомобілів. Оренда автомобілів дозвлить задовольнити потребу в особистій мобільності, при цьому забезпечуючи більш низькі витрати для фізичних осіб і більш високу зручність використання транспортних засобів, що робить автомобілі більш економічно ефективним [3]. Fellows і Pitfield аналізуючи витрати і вигоди для оцінки оренди автомобіля виявили, що люди отримують економічну вигоду за рахунок скорочення подорожей за ціною до 50%, а економіка в цілому виграє за рахунок зменшення пробігу транспортних засобів, збільшення середньої швидкості та економії в паливі, зменшення аварій та викидів. Більш систематичні результати в географічно більшому масштабі можна знайти в дослідженні Шахін і Коена [2], які визначили, що кожен автомобіль спільного користування зменшує потребу на 4-10 приватних транспортних засобів у Європі, 6-23 в Північній Америці, і 7-10 в Австралії.

В цій статті порівнюється вплив двох різних практик використання автомобілів спільного користування на продуктивності батареї. Це робиться за рахунок докладних даних електрокарів і даних про підзарядку для прогнозування SoH батарей двох однакових електромобілів в різних практиках. Основні дослідницькі матеріали роботи можуть бути розташовані в наступних областях: (1) детальний аналіз спільного водіння користувачів електрокарів і частоти підзарядок, сподіваючись таким чином забезпечити додакову інформацію для планування транспортної системи; (2) вплив двох різних методів обміну електрокарів на SoH батареї, оцінити життєздатність цих методів.

2. Практики каршерінга та їх користувачів

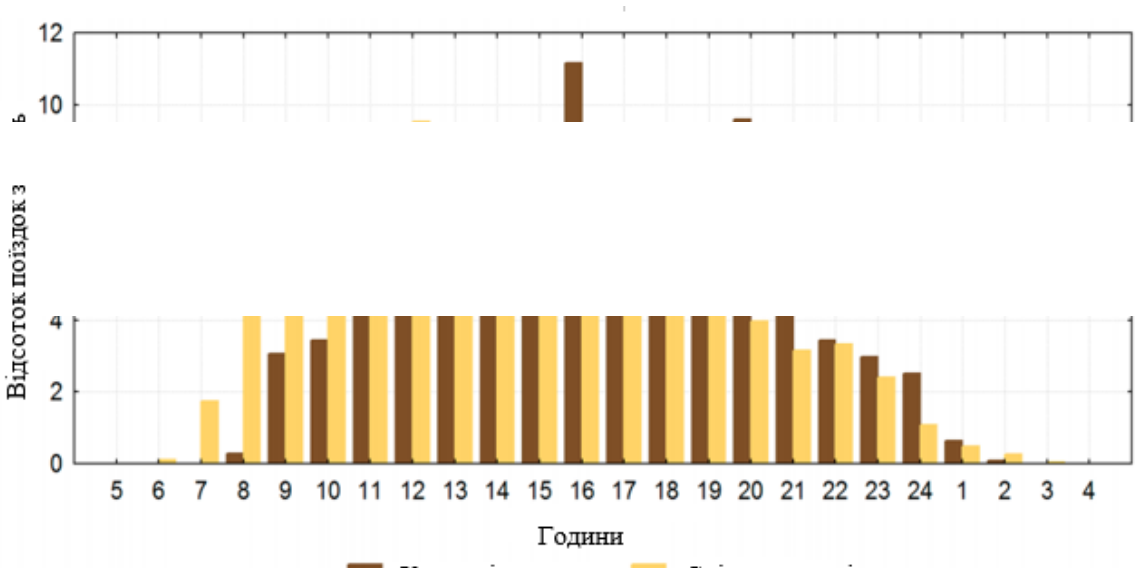
Ми порівнюємо два однакових електромобілі, що орендуються за двох різних практик каршерінгу. Перший автомобіль є васністю компанії, що здає в оренду більш ніж 800 електричних і звичайних транспортних засобів. Дані автомобілі достпні для оренди більш ніж 24000 користувачів. Правилами користування прописано, що після використання користувач зобов'язаний підключити автомобіль для підзарядки, що гарантує максимально заряджений акумулятор для наступного користувача. В системі бронювання користувачі вказують приблизно скільки кілометрів і часу триватиме їхня оренда, для спрощення планування обслуговування. Другий автомобіль знаходиться в спільній власності житлового комплексу, де жильці займаються доглядом спільного майна. Підхід заснований на понятті спільної економіки. Автомобіль активно експлуатується серед 35 членів. Система бронювання передбачає вказування часових інтервалів, протягом якого буде експлуатуватися автомобіль. Підзарядка транспортного засобу здійнюється тільки з ініціативи користувача. В обох випадках автомобіль повинен бути повернутий в початковий пункт. Обидва автомобілі експлуатуються в одинакових кліматичних умовах та регіоні.

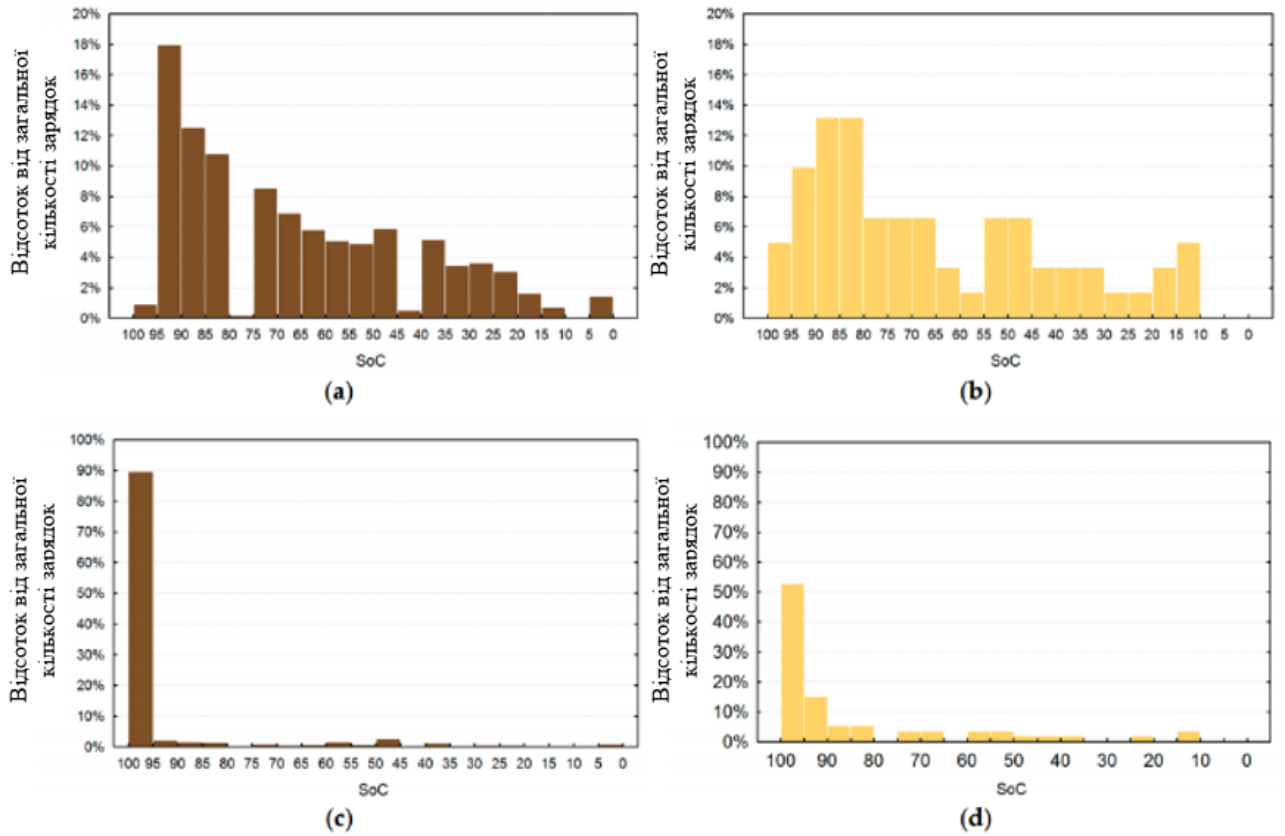
3. Розуміння споживачів – манера водіння та повернення

Для того, щоб забезпечити якомога повне розуміння того, як були використані електромобілі, детально досліджувалися манера водіння і зарядка обох груп користувачів. Дані зарядки було зібрано від зарядних станцій. Дані водіння було зібрано з транспортних засобів за допомогою GPS (переміження) та CAN шини (швидкість автомобіля, струм, напруга, часові мітки, заряд, стан двигуна).

З огляду на використання загальних електромобілі, автомобіль обміну членів компанії, як правило, використовують його в основному в якості другого автомобіля, з високою частотою в другій половині дня і у вихідні дні дисків (22% всіх водіїв були зроблені у суботу). Члени спільного житла були більш схильні використовувати загальний EV в ранкові години (Рис 1). При розгляді питання про тривалість зроблених поїздок, 72% поїздок користувачів спільно для будинку були коротше, ніж в 10 км. Згідно з офіційною статистикою стверджує, що середня поїздка автомобіля у Фландрії становить приблизно 34,4 км; середня поїздка зроблені електромобілі належать автомобіля обміну компанії було 32,56 км, а шляхом спільного житла, 8,4 км.

З огляду на перезарядку, члени спільного





житла, як правило, підзаряджали автомобіль зранку або пізно ввечері. Автомобіль каршерінгової компанії підключався після кожного використання. Таким чином автомобіль не буде від'єднано від зарядного пристрою до наступного користування, що іноді може зайняти більше тижня. В цілому, 7,6% підзарядок були зроблені з super charge. Всі перезарядки членами спільного житло були коротше, ніж день, де 33% з них були мене півгодини, і 4% з super charge.

Правила оренди також впливають на кількість зарядок в день. Для автомобілів каршерінгової компанії, число перезарядок переважно відповідає кількості користувачів в день (переважно один). Для спільного користування, транспортний засіб часто заряджається більше, ніж один раз в день. Беручи до уваги SoC батареї, то в 20% випадкахавтомобілі компанії заряджалися, але SoC батареї було вище, ніж 90%. У більшості випадків, батарея залишалася до повної зарядки. Для членів спільно користування, батарея повністю зарядилася тільки в половині випадків, в той час як значення SoC, при якому акумулятор був підключений на підзарядку, був більш рівномірно розподілений (Рис. 2)

4. Методологія

Для того, щоб оцінити вплив практика каршерінгу, поведінку водіння та зарядки батареї електрокару, ми досліджували SoH, що визначаються як різниця між корисною місткостю і кінцевою місткостю [6]. SoH зазвичай виражається у вигляді відсотка від номінальної потужності і є мірою довгостроковості батареї [4,5]. У порівнянні з SoH, SoC визначається як відсоток від доступної ємності і є мірою короткочасної здатності батареї. Більш детально, SoC показує залишковий заряд батареї в даний момент, в порівнянні з енергією при повному заряді, що дає уявлення про те, скільки батарея працюватиме до наступної перезарядки. В аналогії зі звичайними автомобілями, SoC відповідає паливному манометру, в той час як SoH буде відповідає здатності паливного бака зберігати паливо. У цій аналогії, паливний бак матиме змінний доступний обсяг.

У даній роботі SoH розраховується з даних експлуатації, в той час як SoC зібрана з CAN шини. Дані з CAN шини засобів зібрані з частотою 10 Гц. На Рис. 3. показано миттєва передача потужності батареї (W) з плином часу під час експлуатації.

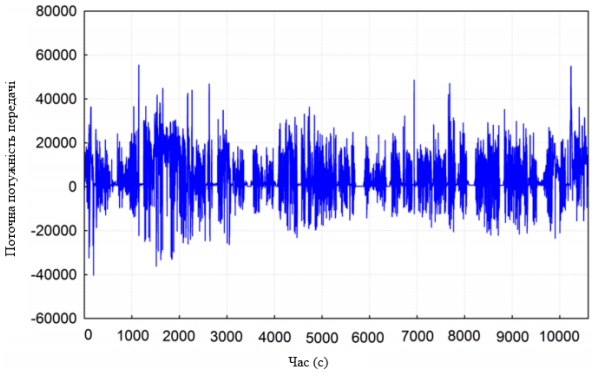


Рис. 3. Графік V*I(W)

На основі миттєвої передачі потужності батареї сумарна чиста енергія подається від акумуляторної батареї може бути обчислена шляхом трапецієподібної чисельного інтегрування струму батареї (I) і напруги (V) з плином часу, як показано рівнянням:

$$E_{te} = \int V * I \, dt$$

Крім того, знаючи SoC, на початку (SoC_1) і в кінці (SoC_2), SoH може бути визначений на основі рівняння:

$$SoH = \frac{E_{te}}{SoH_{100\%} * (SoC_1 - SoC_2)}$$

5. Вплив каршерінгу на стан батареї

Для обох електромобілів, використовуючи обчислення SoH з секції методології, ми визначили значення SoH для кожного циклу розрядки. Кількість записаних циклів трохи розрізнялися, для транспортного засобу, що належить компанії, було 59 і для автомобілю, що є спільній власності – 63. У літературі вказується, що протягом перших 500 циклів або близько того, ємність змінюється лінійно, обчислені значення SOH були використані для визначення цього лінійного тренда.

Для оцінки лінійного тренда доброти нападі і його застосовності для прогнозування майбутніх SoH елементів живлення, ми розрахували найменше квадратичне відхилення, середнє відхилення, відносну квадратичну похибку та відносне абсолютне відхилення.

$$LSD = \frac{\sum_{i=1}^{N}(E_i - O_i)^2}{N-1}$$

$$AD = \frac{\sum_{i=1}^{N}|E_i - O_i|}{N-1}$$

$$RSE = \frac{\sum_{i=1}^{N}[(E_i - O_i)/E_i]^2}{N-1}$$

$$RAD = \frac{\sum_{i=1}^{N}|E_i - O_i|/E_i}{N-1}$$

де: N – число спостережень або сума ваг; E_i – передбачене значення випадку і.

Таблиця 1 містить більш детальний аналіз оцінки лінійного тренда для обох автомобілів

Табл. 1. Достовірність лінійних оцінок

Достовірність виміру	Спільна власність	Каршерінг компанія
LSD	1.605182	1.699434
AD	0.949703	1.07033
RSE	0.000184	0.00030
RAD	0.010116	0.01410

Як правило, це прийнято в автомобільній промисловості, що авершеніе батареї життя є кількість повних циклів заряду-розряду, батарея може виконувати до його номінальної потужності падає нижче 80% від його первісної номінальної потужності [7]. Рис. 4 забезпечує екстраполяцію лінійних трендів як для EV батареї, поки вона не перетне кінця життя кордону.

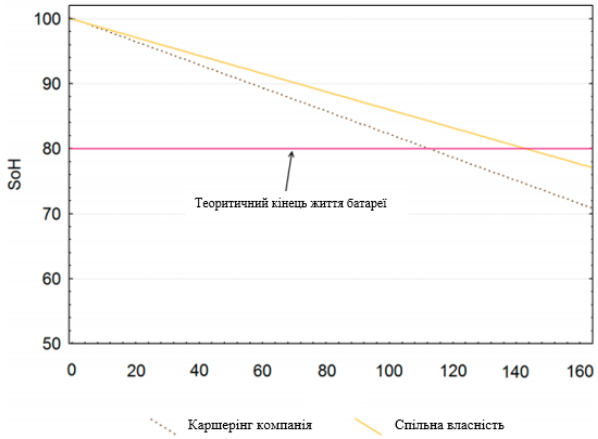


Рис. 4. Лінійна екстраполяція SoC тенденції і теоретичний кінець життя кордону

6. Висновки

Ми виявили, що дані від зарядних станцій і датчиків електрокарів можуть бути успішно використані для прогнозування SoH батареї. Крім того, ми збагатили існуючі імітаційні моделі в області оцінки SoH з емпіричним підтвердженням аналізу. Аналіз оснований на даних з двох однакових електромобілях, використання яких відрізнялися в середньому SoC, DoD, і відсоток використання super charge. Результати вказують на те, що відстрочення зарядки і нижче менеше використання supper charge можуть уповільнити процес деградації батареї.

Крім того, деградація батареї пов'язана з вартістю батареї і вартості автомобіля (вартість батареї близько 54% від загальної вартості автомобіля [8]). Дані висновки можуть бути цінним довідником для каршерінгу електрокарів, так як вони вказують

на потенційні оновлення, які можуть бути інтегровані в існуючі системи каршерінгу.

Список літератури

1. Meijkamp, R. Changing consumer behaviour through eco-efficient services : An empirical study of car sharing in the Nerherlands. Bus. Strategy Environ. **1998**, 7, 234–244.
2. Shaheen, S.A.; Cohen, A.P. Growth in worldwide carsharing: An international comparison. Transp. Res. Rec. J. Transp. Res. Board **2007**, 1992, 81–89.
3. Fellows, N.; Pitfield, D. An economic and operational evaluation of urban car-sharing. Transp. Res. D Transp. Environ. **2000**, 5, 1–10.
4. Nikolian, A.; Firouz, Y.; Gopalakrishnan, R.; Timmermans, J.-M.; Omar, N.; van den Bossche, P.; van Mierlo, J. Lithium ion batteries—Development of advanced electrical equivalent circuit models for nickel manganese cobalt lithium-ion. Energies **2016**, 9, 360.
5. Le, D.; Tang, X. Lithium-ion battery state of health estimation using Ah-V characterization. In Proceedings of the Annual Conference of Prognostics and Health Management (PHM) Society, Montreal, QC, Canada, 20–23 June 2011.
6. Marra, F.; Træholt, C.; Larsen, E.; Wu, Q. Average behavior of battery-electric vehicles for distributed energy studies. In Proceedings of the 2010 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT Europe), Gothenburg, Sweden, 11–13 October 2010.
7. Magnor, D.; Gerschler, J.B.; Ecker, M.; Merk, P.; Sauer, D.U. Concept of a battery aging model for lithium-ion batteries considering the lifetime dependency on the operation strategy. In Proceedings of the European Photovoltaic Solar Energy Conference, Hamburg, Germany, 21–25 September 2009.
8. Cars 21. How to Reduce EV Production Costs? EV Battery Tech USA. 2011. Available online: http: //www.cars21.com/news/view/670 (accessed on 18 October 2016).

УДК 004.75

ГОНТАРЕНКО І.В.

МОДЕЛЬ БАЛАСУВАННЯ НАВАНТАЖЕННЯ У ВІРТУАЛЬНОМУ ОБЧИСЛЮВАЛЬНОМУ КЛАСТЕРІ НА ОСНОВІ КОНТЕЙНЕРІВ ДОДАТКІВ

Розглядається створення віртуальної розподіленого обчислювального середовища, яке дозволить конфігурувати надані обчислювальні ресурси у відповідності до вимог програми, а також систематизація та порівняння поширених методів масштабування додатків та алгоритмів балансування навантаження.

The subject of this article is creating a virtual distributed computing environment that will configure the computing resources provided in accordance with the program, as well as ordering and comparing common methods of application scaling and load balancing algorithms.

1. Введення

В даній статті розглянуто практичне використання розподіленого обчислювального середовища, яке дозволить конфігурувати надані обчислювальні ресурси у відповідності до вимог програми, а також систематизація та порівняння поширених методів масштабування додатків та алгоритмів балансування навантаження.

У сучасному світі людина з усіх сторін оточена обчислювальними пристроями. Потужності обчислювальної техніки протягом всього періоду її розвитку постійно зростають, але навіть незважаючи на закон Мура [1], завжди існували та існують завдання, яким існуючих потужностей поодиноких комп'ютерів виявляється недостатньо. Саме тому останнім часом велику популярність та важливу роль відіграють високонавантажені системи обчислення. Ефективним вирішення таких завдань є масштабування додатків, балансування навантаження та об'єднання комп'ютерів у різні обчислювальні мережі або кластери. Для забезпечення узгодженої роботи вузлів обчислювальної мережі на стороні провайдера використовується спеціалізоване проміжне програмне забезпечення, що забезпечує моніторинг стану обладнання і програм, балансування навантаження, забезпечення ресурсів для вирішення завдання. Також актуальною проблемою є вибір засобу балансування навантаження, який би найкращим чином відповідав задачам створення конкретного сервісу в конкретних умовах. Для порівняння і налаштування систем балансування навантаження згідно поставлених вимог важливо мати достатньо повний набір відповідних критеріїв.Хмарні обчислення є сукупністю кількох моделей і підходів, надаючи послуги управління IT-серівісамі. Кожен з фахівців трактує це по-різному, деякі вважають хмарні обчислень хостингом віртуальних машин, інші ж вважають, що хмарні обчислення - це такі програми як Дропбокс і Гугл Драйв, таких різних варіацій визначень на просторах інтернету величезна кількість. Повернемося до витоків створення хмарних обчислень, ідея належала Джозефу Ліклайдеру (1915 - 1990, відомий в науковій і IT-середовищі як JCR або «Lick»). Трактування якої полягала в тому, що кожна людина на землі буде підключений до мережі, з якої він буде отримувати не тільки дані, але і програми.

«Хмара» - одне з небагатьох слів, яке вже довгий проміжок часу використовується фахівцями IT-сфери для візуалізації складної обчислювальної інфраструктури, що приховує свою внутрішню організацію за певним інтерфейсом, на мережевих діаграмах.

Однак термін «Хмарні обчислення» з'явився на світ відносно недавно. Згідно з результатами аналізу пошукової системи Google, термін «Хмарні обчислення» («Cloud Computing») почав висвітлюватися

в кінці 2007 - початку 2008 року, поступово витісняючи словосполучення «Грід-обчислення» («Grid Computing»). Однією з перших компаній, що дали світові цей термін, стала компанія IBM, яка розгорнула на початку 2008 року проект «Blue Cloud» та проспонсорувала Європейський проект «Joint Research Initiative for Cloud Computing». [1]

Визначення хмарних технологій від відомих діячів сфери IT-технологій:

: «Хмарні обчислення - це парадигма великомасштабних розподілених обчислень, заснована на ефекті масштабу, в рамках якої пул абстрактних, віртуалізованних, динамічно - масштабованих обчислювальних ресурсів, ресурсів зберігання, платформ і сервісів за запитом зовнішнім користувачам через Інтернет». (Ян Фостер)

«Хмарні обчислення - це не тільки додатки, які поставляються як послуг через Інтернет, а й апаратні засоби і програмні системи в центрах обробки даних, які забезпечують надання цих послуг ...».(Лабораторія Берклі)

«Облік - це великий пул легко використовуваних і легкодоступних віртуалізованних ресурсів (таких як апаратні комплекси, сервіси та ін.). Ці ресурси можуть бути динамічно перерозподілені (масштабовані) для підстроювання під динамічно змінюється навантаження, забезпечуючи оптимальне використання ресурсів. Цей пул ресурсів зазвичай надається за принципом «оплата у міру використання». При цьому власник хмари гарантує якість обслуговування на основі певних угод з користувачем ». (Луїс Вакуеро)

США Хмарні обчислення - інформаційно-технологічна концепція, що передбачає забезпечення повсюдного та зручного мережевого доступу на вимогу до загального пулу конфігуруються обчислювальних ресурсів (наприклад, мереж передачі даних, серверів, пристроїв зберігання даних, додатків і сервісів - як разом, так і окремо), які можуть бути оперативно надані та звільнені з мінімальними експлуатаційними витратами або зверненнями до провайдера. (Національний інститут стандартів і технологій)

Національний інститут стандартів і технологій США виділив нижчеперелічені характеристики хмарних виділень:

Самообслуговування на вимогу - споживач самостійно визначає і змінює обчислювальні потреби, такі як серверний час, швидкості доступу та обробки даних, обсяг збережених даних без взаємодії з представником постачальника послуг;

Універсальний доступ по мережі - послуги доступні споживачам через мережу передачі даних незалежно від використовуваного термінального пристрою;

Об'єднання ресурсів (англ. *Resource pooling)* - постачальник послуг об'єднує ресурси для обслуговування великого числа споживачів в єдиний *пул* для динамічного перерозподілу потужностей між споживачами в умовах постійної зміни попиту на потужності; при цьому споживачі контролюють тільки основні параметри послуги (наприклад, обсяг даних, швидкість доступу), але фактичний розподіл ресурсів, що надаються споживачеві, здійснює постачальник (в деяких випадках споживачі все-таки можуть управляти деякими фізичними параметрами перерозподілу, наприклад, вказувати бажаний центр обробки даних з міркувань географічної близькості);

Еластичність - послуги можуть бути надані, розширені, звужені в будь-який момент часу, без додаткових витрат на взаємодію з постачальником, як правило, в автоматичному режимі;

Облік споживання - постачальник послуг автоматично обчислює спожиті ресурси на певному рівні абстракції (наприклад, обсяг збережених даних, пропускна здатність, кількість користувачів, кількість транзакцій), і на основі цих даних оцінює обсяг наданих споживачам послуг.

З точки зору постачальника, завдяки об'єднанню ресурсів і непостійного характеру споживання з боку споживачів, хмарні обчислення дозволяють економити на масштабах, використовуючи менші

апаратні ресурси, ніж були потрібні б при виділених апаратних потужностях для кожного споживача, а за рахунок автоматизації процедур модифікації виділення ресурсів істотно знижуються витрати на абонентське обслуговування.

З точки зору споживача ці характеристики дозволяють отримати послуги з високим рівнем доступності та низькими ризиками непрацездатності, забезпечити швидке масштабування обчислювальної системи завдяки *еластичності* без необхідності створення, обслуговування і модернізації власної апаратної інфраструктури.

Зручність і універсальність доступу забезпечується широкою доступністю послуг і підтримкою різного класу термінальних пристроїв (персональних комп'ютерів ,мобільних телефонів , інтернет-планшетів) [2].

Модель обслуговування визначає автоматизований рівень IT процесів інфраструктурі, існує 3 типи обслуговування.

Перший тип це програмне забезпечення як послуга (SaaS), прикладами даного забезпечення є сервіси Google Docs і Gmail.

Другим типом є платформа як послуга (PaaS), прикладом якої є Google Apps, яка надає застосункі для бізнесу в режимі онлайн, доступ до якіх відбувається за помощью Інтернет-браузера тоді як ПЗ и дані зберігаються на серверах Google.

Третім типом є модель, інфраструктура як послуга (IaaS), прикладом є такі компанії як: Amazon, Microsoft, VMWare, Rackspace та Red Hat.

Також існують 4 моделі розгортання: приватна хмара, публічне хмара, суспільне хмара, гібридне хмара.

Приватна хмара - концепція побудови IT, спрямована на реалізацію принципу «IT, як послуга». Сервери об'єднуються в пул, з якого на вирішення певних завдань виділяються обчислювальні ресурси, не прив'язані до конкретних фізичних серверів. Надання потужностей з такого динамічного пулу здійснюється за запитами в службу Service Desk або через спеціальний портал - «вітрину сервісів». При цьому можуть бути виділені віртуальні сервери під завдання бізнес-додатків або тестування, надані окремі додатки або цілі віртуальні робочі місця користувачів, а ПК на столах співробітників можуть бути замінені «тонкими клієнтами» - невеликими пристроями, тихими і економічними.

Публічне хмара фізично розташовується і знаходиться у власності, управлінні та експлуатації провайдера. Також є найбільш простим способом реалізації концепції IaaS (Infrastructure as a Service) - надання комп'ютерної інфраструктури як послуги.

На відміну від приватного хмари, публічне хмара дозволяє отримати готову інфраструктуру без початкових витрат.

Публічне хмара володіє практично необмеженими можливостями масштабування.

Громадське хмара - модель хмарного розміщення, відповідно до якої хмарні ресурси використовуються конкретним спільнотою споживачів з організацій, що мають спільні завдання.

Гібридне хмара - модель хмарного розміщення, в якій об'єднані дві і більше ICOT, що належать різним організаціям або типам моделей (приватним, громадським або публічним

Платформа ICOT - система програмних і програмно-апаратних засобів, що реалізують концепцію «хмарних обчислень» відповідно до моделі хмарного розміщення і видом наданих хмарних послуг.

Окрім основних моделей, існують також перелік додаткових моделей обслуговування в хмарних системах:

Апаратне забезпечення як послуга (Hardware as a Service - HaaS): користувачеві надається обладнання на правах оренди, яке він може використовувати для власних цілей. По суті HaaS нагадує IaaS за винятком того що ви маєте голе обладнання на основі якого розвертаєте свою власну інфраструктуру з використанням найбільш підходящого ПО. Перевагою HaaS є можливість економити на обслуговуванні обладнання.

Робоче місце як послуга (Workplace as a Service - WaaS): компанія використовує хмарні обчислення для організації робочих місць своїх співробітників, налаштувавши і встановивши все необхідне програмне забезпечення, необхідне для роботи персоналу.

Дані як послуга (Data as a Service - DaaS): користувачеві надається дисковий простір, яке він може використовувати для зберігання великих обсягів інформації.

Безпека як послуга (Security as a Service): користувачам надається можливість швидко розгортати продукти, що дозволяють забезпечити безпечне використання веб-технологій, електронного листування, а також локальної системи, що дозволяє економити на утриманні своєї власної системи безпеки.

Все як послуга (Everything as a Service - EaaS): користувачеві буде надано все від програмно -аппаратноє частини і до управління бізнес процесами, включаючи взаємодію між користувачами, при наявності доступу в мережу Інтернет.

Список літератури

1.http://k504.xai.edu.ua/html/ucheba/rss/RSS_Lekciya_10.pdf

2.https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BB%D0%B0%D1%87%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F

3.Джорд Різ “Хмарні обчислення”, Пер. з англ. - СПб .: БХВ-Петербург, 2011. - 288 с. ISBN 978-5-9775-0630-4

УДК 378.4+004.738.5

КОБЕЦЬ Н.М.,
КОВАЛЮК Т.В.

АСПЕКТНО-ОРІЄНТОВАНИЙ АНАЛІЗ ВИСЛОВЛЮВАНЬ В ІНФОРМАЦІЙНИХ СИСТЕМАХ

У даній статті розглянуто основні задачі та підзадачі аналізу висловлювань (opinion mining), продемонстровано зв'язок між ними, визначено проблеми, що постають перед дослідниками та існуючі методи їх вирішення.

The article discusses the main tasks and subtasks expression analysis (opinion mining), demonstrated a link between them, defined the problems faced by researchers and existing methods for their solution

1. Вступ

З розвитком інформаційних технологій та їх впровадженням в усі сфери життя важливим є аналіз висловлювань (opinion mining), що полягає в інтелектуальному автоматичному видобуванні суб'єктивної інформації (думок, оцінювальних суджень, емоцій, почуттів, вірувань тощо) з текстової інформації. Виявлення та оцінювання позитивності чи негативності висловлювань стосовно певного об’єкта дослідження може бути застосовано до різноманітних галузей, зокрема до промисловості, маркетингу, освіти тощо. Практичне застосування аспектно-орієнтованого аналізу висловлювань можливо в контент-аналізі як формалізованому методі аналізу тексту. Маркетингові дослідження показують, що онлайн-відгуки значно впливають на поведінку рівня продаж того чи іншого товару [1]. Однак їх зростаючий обсяг призводить до того, що споживачам стає неможливо ознайомитися з кожним. З іншого боку, інтернет-відгуки забезпечують виробників інформацією про те, чи задоволені споживачі їх продукцією. Отже, задача визначення змісту та емоційного окрасу висловлювань споживачів, які стосуються окремих аспектів товарів (aspect-based opinion mining), в системі оцінювання, адаптованій до українського ринку, є актуальною та важливою.

2. Постановка проблеми

Для аналізу відгуків користувачів потрібно обробляти складні синтаксичні конструкції висловлювань, фрази, що були вжиті у переносному сенсі, ідентифікувати спам, шум і сарказм тощо. Отже, розроблення новітніх інформаційних технологій у галузі opinion mining зводиться до таких завдань:

- знаходження у текстових даних позитивних та негативних висловлювань;
- присвоєння позитивним чи негативним висловлюванням певного числового еквівалента;
- узагальнення позитивних та негативних висловлювань до певного інтегрального показника з метою порівняння об’єктів дослідження.

3. Мета статті

Мета статті – визначення методів та алгоритмів емоційного окрасу висловлювань споживачів, які стосуються окремих аспектів товарів в системі оцінювання, адаптованій до українського ринку.

Для цього необхідно вирішити такі задачі:

- провести аналіз існуючих методів крос мовного аналізу висловлювань;
- розглянути методи та алгоритми, які дозволять адаптувати механізми aspect-based opinion mining до відгуків українською мовою.

4. Методи та задачі аспектно-орієнтованого аналізу висловлювань

Методи, засновані на аналізі частотних характеристик тексту (frequency-based), для вилучення аспектів застосовують прості фільтри на іменникових конструкції. Щоб уникнути недоліків цих методів, були запропоновані методи, засновані на відношеннях, які використовують обробку

природної мови для пошуку відносин між аспектами та пов'язані з ними почуття. Враховуючи переваги обох підходів, були запропоновані гібридні методи, які використовують природної мови відносини для фільтрації аспектів, що часто зустрічаються. Точність гібридних методів значно вище, ніж двох попередніх. Однак, як в двох попередніх випадках, гібридні методи потребують ручного налаштування різних параметрів. Щоб уникнути необхідності ручного налаштування параметрів, використовують методи навчання з учителем, який автоматично вивчає параметри моделі даних. Методи навчання без учителя, а також йомовірнісні моделі дозволяють визначити, про що говориться в тексті, тобто семантику тексту. Взаємозв'язок задач галузі аналізу висловлювань подані на рис.1. Задачі opinion mining на рівні документа (document-level):

- виявити у тексті суб'єктивні оцінки;
- визначити емоційний окрас тексту, тобто його полярність;
- визначити якість та корисність відгуку;
- виявити спам.

Задачі opinion mining на рівні окремого речення (sentence-level):

- пошук та аналіз висловлювань;
- питально-відповідальна система;
- узагальнення знайдених висловлювань;
- застосування opinion mining для порівняльних речень.

Задача opinion mining на рівні окремої фрази (phrase-level) - аналіз аспектів (окремих характеристик) продукту.

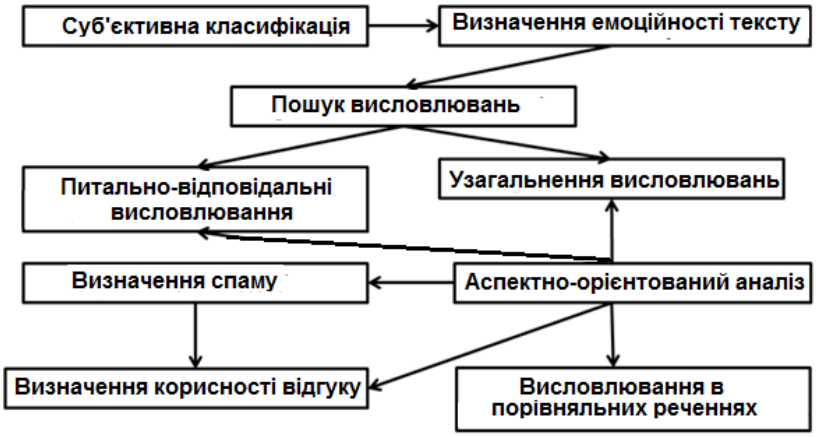


Рис. 1. Взаємозв'язок задач аналізу висловлювань

5. Математична постановка задачі

Кожне висловлювання можна представити у вигляді наступного п'ятивимірного вектора:

$\left(e_j, a_{jk}, so_{ijkl}, h_i, t_l\right)$

де e_j – сутність, для якої виконується аналіз висловлювань

a_{jk} – один з аспектів сутності e_j

h_i – автор висловлювання

t_l – час, коли автор h_i залишив своє висловлювання

so_{ijkl} – емоційна направленість висловлювання, залишеного автором h_i по відношенню до аспекта a_{jk} сутності e_j в час t_l. Може бути позитивною, негативною або нейтральною, або виражати різні рівні інтенсивності від 1 до 5.

Пара e_j та a_{jk} завжди виражає ціль висловлювання.

6. Кросмовна латентна семантична асоціація

Кожний аспект продукту зазвичай позначається за допомогою набору термів. Наприклад, англійською мовою «photo», «picture», «image» та українською мовою «фотографія», «картинка», «зображення» – відносяться до одного і того самого аспекту.

Кросмовна категоризація аспектів товару сфокусована на їх категоризації в єдину семантичну категоріальну структуру. Нехай X є простір характеристик для представлення екземплярів мультимовних характеристик товару, а Y – множина позначок для семантичних категорій. Нехай

$p_s(x, y)$ та $p_t(x, y)$ є прогнозованими основним розподілом категорій та справжнім основним розподілом категорій. Очікується, що $p_s(x, y)$ краще апроксимує $p_t(x, y)$ без використання будь-яких помічених даних.

Кросмовна категоризація характеристик товарів, яка базується на лексичному порівнянні, як правило, не є спроможною визначити основний семантичний розподіл різних мультимовних характеристик. Багато термінів, які позначають однакові аспекти не схожі на лексичному рівні. Такі приховані семантичні асоціації між словами надають можливість для визначення основного семантичного розподілу в домені.

Для подальших досліджень використана модель М для визначення кросмовних латентних семантичних асоціацій між мультимовними термами, що позначають аспекти товару. Ця модель навчається на непоміченому тексті висловлювань користувачів. В процесі навчання кожен аспект товару характеризується за багатовимірним вектором-ключом.

Характеристики семантичних асоціацій в моделі – це приховані випадкові змінні, які виведені з даних. Очевидно, що модель М може краще визначати кросмовні латентні семантичні асоціації поміж аспектами товарів. Використовуючи модель, можна краще апроксимувати справжні розподіли семантичних категорій $p_t(y|x; M)$ без необхідності використання помічених даних.

7. Кросмовні віртуальні контекстні документи

Для того, щоб визначити приховані взаємозв'язки мультимовних термів, кожен терм аспекту товару характеризується за кросмовним віртуальним контекстним документом.

Дано терм аспекту товару pf, його кросмовний віртуальний контекстний документ cvd_{pf}, який складається з таких багатовимірних прихованих семантичних ключів:

- поточного терму pf;
- терму pf^T, який є автоматичним перекладом терму pf
- множини складових компонентів з pf

$$PMI(x_j, pf) = \log_2 \frac{P(x_j, pf)}{P(x_j) \cdot P(pf)}, \quad (1)$$

та pf^T, які позначаються як W_{pf} та W_{pf^T} відповідно;

- приховані семантичні теми складових компонентів з pf та pf^T, які позначаються як S_{pf} та S_{pf^T} відповідно, на рівні слів;
- одномовні латентні семантики pf рівня аспектів товару, які позначаються як MFS_{pf}.

Отже,

$$cvd_{pf} = \left\{pf, pf^T, W_{pf}, W_{pf^T}, S_{pf}, S_{pf^T}, MFS_{pf}\right\}$$

У конструкції кросмовного віртуального контекстного документу, генерують одномовні приховані семантичні теми на рівні аспектів товару та слів, використовуючи алгоритми, представлені в [8].

Компонентні слова згруповані в множину прихованих тем, відповідно до їхнього контексту в одномовну збірку (corpus). Одномовна прихована семантична тема на рівні аспектів товару створюються відповідно до їх прихованої семантичної структури і контекстних уривків у відповідній збірці. Повний документ машинного перекладу зазвичай використовується для визначення семантичних асоціацій між аспектами написаними на різних мовах. Для того, щоб зменшити шум, який виникає при машинному перекладі, кросмовний віртуальний контекстний документ використовує лише переклад окремого терму аспекту товару замість перекладу повного тексту рецензії.

cvd_{pf} звичайно описує багатовимірні кросмовні приховані семантичні характеристики аспекту pf в рецензіях. В

моделі будується вектор для pf зі всіма розглянутими ознаками із cvd_{pf}:

$Vector(cvd_{pf}) = \{x_1, \ldots, x_j, \ldots, x_m\}$,

де x_j – описує j-ую контекстну ознаку, пов'язану з pf,

m – загальна кількість ознак в cvd_{pf}

Вага кожної контекстної ознаки x_j в cvd_{pf} розраховується за індексом *PMI* (*pointwise mutual information*) між x_j та pf:

де $P(x_j, pf)$ - ймовірність того, що pf та x_j зустрінуться в тексті поряд;

$P(x_j)$ - ймовірність того, що x_j зустрінеться в тексті;

$P(pf)$- ймовірність того, що pf зустрінеться в тексті;

Вага нормалізована як невід'ємна за допомогою логарифмічної функції.

8. Навчання моделі

Розглянемо детальніше процес навчання моделі M на кросмовних віртуальних семантичних контекстуальних документах. Як вхідні дані використовують непомічені збірки рецензій R_{l_1} та R_{l_2} написані на мовах l_1 та l_2. Будемо розглядати терми аспектів товарів, які написані на мові l. В конструкції cvd_{pf} латентні теми з компонентних слів генеруються з використанням одномовної моделі θ^l_{wd} рівня слів. Одномовна латентна семантика MFS_{pf} кожного аспекту товару згенерована за допомогою одномовної тематичної моделі θ^l_{mp} рівня аспектів термів. Вага кожного елемента cvd_{pf} розраховується за допомогою індексу PMI за формулою (1). Далі модель M з розподілом Діріхлє генерує множину кросмовних віртуальних контекстних документів $CVD\,Set$, використовуючи алгоритм з [2]. Алгоритм навчання моделі детально описує повний процес навчання, де:

- функція $MT(pf_i)$ позначає, результат автоматичного перекладу терму pf_i.
- функція $TP(data, \theta)$ генерує латентну тему для аргументу $data$, використовуючи латентну тематичну модель θ;
- θ^l_{wd} описує одномовну тематичну модель рівня слів для даної мови l;
- θ^l_{mp} описує одномовну тематичну модель рівня аспектів товару для даної мови l.

Досліджувана модель вивчає апостеріорну ймовірність декомпозиції мультимовних аспектів термів та їх віртуальні контекстні документи в теми. Вона розширює традиційні тематичні моделі «bag of words» до контекстно залежної кросмовної концептної асоціаційної моделі.

Висновки

Аспектно-орієнтований аналіз є найбільш деталізованим серед усіх рівнів аналізу висловлювань та необхідним для більшості практичних застосувань. У даній статті розглянуті математична постановка задачі аспектно-орієнтованого висловлювання, кросмовна латентна семантична асоціація, розглянута характеристика терм аспекту товару за кросмовним віртуальним контекстним документом та процес навчання моделі.

Список літератури

1. Andrew Lipsman. Online consumer-generated reviews have significant impact on offline purchase behavior. Technical report, Comscore Inc., 2007. [Електронний ресурс] / Г. Andrew Lipsman. – Режим доступу: http://www.comscore.com/Insights/Press_Releases/2007/11/Online_Consumer_Reviews_Impact_Offline_Purchasing_Behavior

2. Pang B. Opinion mining and sentiment analysis [Текст]/ Pang B., Lee L // N.Y.:Now Publishers Inc., 2008. – 155 с.

3. Wiebe J. Development and use of a goldstandard data set for subjectivity classifications [Текст] : In Proceedings of the 37th annual meeting of the Association for Computational Linguistics on

Computational Linguistics, ACL ’99 / Wiebe J., Bruce R., O’Hara T. – Stroudsburg, PA, USA: Association for Computational Linguistics, 1999. – c. 246–253

УДК 004.023

ЛІНЕВИЧ О.С.

ОГЛЯД ТЕХОЛОГІЙ ОБРОБКИ НАДВЕЛИКИХ МАСИВІВ ДАНИХ

В даній статті розглянуто новітні програмні платформи для обробки великих масивів даних. Порівняно переваги і недоліки таких платформ. Проаналізовано платформи чотирьох напрямків: напрямок паралельної обробки даних(data parallel), напрямок обробки за допомогою паралельних задач (task parallel), напрямок обробки даних за допомогою графів(graph parallel), напрямок обробки потоків даних(stream processing programming).

In the article described the outstanding data software platforms for processing large datasets. Comparing the advantages and disadvantages of each. Platforms of four directions - the direction parallel processing (data parallel), direction processing using parallel tasks (task parallel), the direction of data using graphs (graph parallel), direction processing data streams (stream processing programming) - are analysed here.

1. Технологія паралельної обробки даних

Принцип роботи програмних платформ даної технології полягає в тому, щоб працювати із даними, які розділені на менші частини і розподілені між паралельними обчислювальними вузлами. Кожен вузол виконує одну і ту саму задачу на різних частинах розподілених даних, після чого результати виконаної роботи поєднуються і надаються користувачу для подальшої їх обробки. Приклади паралельних систем - Хадуп(Hadoop), Спарк(Apache Spark).

1) Хадуп використовує таку програмну модель, як MapReduce. Ця модель дає можливість зробити прості обчислення без знань складних деталей паралелізму, а працює із двома операціями - map та reduce, які в свою чергу потребують лише вхідні дані у вигляді пари "ключ/значення". Хадуп дає можливість організовувати і обробляти дані в той же час, коли вони знаходяться на вихідному кластері зберігання даних.

Переваги Хадуп:

Масштабованість. Дана технологія дає можливість працювати із великими даними на звичайних(не потужних) комп'ютерах, налаштувавши в системі архітектуру "master-slave".

Відмовостійкість. Ця здатність існує із-за використання моделі MapReduce, яка в свою чергу дублює дані в декілької місцях, що дає можливість при збої одного кластера скористатись іншим і там дістати необхідні дані.

2) Спарк – використовує модель програмування, яка дає можливість поєднувати такі оператори, як mappers, reducers, joins, group-bys, та filters. Поєднання цих операцій допомагає в легкий спосіб виконати складні обчислення. Складні обчислення виконуються по одному із таких двох шляхів: стійкі розподілені масиви даних(resilient distributed datasets) чи паралельні операції над цими даними. Перший представляє ітераційні обчислення, другий – неітераційні.

Переваги: Масштабованість і відмовостійкість із-за наявності MapReduce.

Легкість у використанні. Спрощене написання коду із-за довільного поєднання операторів. Просте комбінування пакетних, інтерактивних і потокових задач в одній програмі, внаслідок чого задачі виконуються швидше у 100разів, ніж в системі Хадуп.

3)Dryad[4] - дослідницький проект фірми Майкрософт. Цей проект призначений для створення паралельних і розподілених програм для масштабування від маленького кластера до великого центру обробки даних.

Обчислення структуровано у вигляді орієнтованого графа: програми - вершини графа, в той час як канали – ребра графа. Завдання - генерація графа, який може

синтезувати будь-орієнтований ациклічний граф. Ці графи можуть змінюватись під час виконання, у відповідь на важливі події в обчисленнях.

Dryad використовує такі механізми обчислень, як map-reduce Google, або реляційна алгебри та дає вручну створити роботу та управляти нею, управляти ресурсами, моніторити роботу і візуалізувати.

Переваги:

Масштабованість - призначена для масштабування до більших реалізацій, до тисяч комп'ютерів.

Продуктивність.

може виконувати завдання, що містять сотні тисяч вершин, обробки багатьох терабайт вхідних даних протягом декількох хвилин. Microsoft зазвичай використовує додатки Дріада для аналізу петабайт даних на кластерах тисяч комп'ютерів.

Гнучкість.

Програмісти можуть легко використовувати тисячі машин і створити великомасштабне розподілені додатки, не вимагаючи від них опанувати будь-який паралелізм програмування поза можливості намалювати графік залежностей даних їх алгоритмів.

2. Технологія паралельної обробки за допомогою виконання окремих задач

Принцип даної технології полягає в тому, щоб обробляти дані на різних процесорах в паралельному обчислювальному середовищі. Потоки(threads) розподілені по різним паралельним обчислювальним вузлам. Потоки передають дані один одному.

Приклад програмної платформи - система HTCondor [5].

HTCondor – це пакетна система для використання циклів простою на персональних робочих станціях. Надає механізм черг для робіт, схему пріоритетів, моніторинг та управління ресурсами. За її допомогою можна створювати контрольні точки і переміщувати роботу в іншу машину. Для цих машин не потрібно поділяти одну файлову систему на двох. М HTCondor можна розширити, підключивши специфічні протоколи, такі, як GridFTP і Globus Online.

Переваги:

Гнучкість. Задачі можуть перебувати одночасно в двох станах: обов’язкова(першого рівня важливості. job requirements) задача чи необов’язкова(job preferences).

Ефективність. Висопродуктивна обчислювальна система, яка використовує ефективно обчислювальні ресурси.

3. Технологія паралельної обробки за допомогою графів

Платформи даного напрямку представлені представлені обчислювальними програмами, які є вершинами графа. Ці програми виконуються паралельно і взаємодіють по краям графа. Із-за такої взаємодії обмежується взаємодія з іншими вершинами(із графічною структурою), що допомагає оптимізувати компонування даних і їх зв'язки. Прикладами програмних платформ такого напрямку є системи, які використовують для аналізу соціальних мереж - Pregel, Graphlab, GraphX.

1) Модель програмування Pregel.створена для обробки великих графів в розподілених середовищах. Це вершинно-орієнтована модель, яка визначає серію дій для однієї вершини, потім програма пропускає вершини через граф і після цього програма видає результат.

Ця програма створена для вирішення широкомасштабних обчислень графів, які потребують сучасних систем таких, як мережі інтернет та веб-графи.

Переваги: дасштабність. Добре масштабований. Працює із розширеними графами, які мають мільйони вершин.

Відмовостійкість - досягається за рахунок контрольних точок.

Продуктивність.

Pregel[6] дуже швидкий в порівнянні із фреймворками без графів. Але в ході реалізації його чекають повільні робочі, які зменшують швидкість роботи програми.

Гнучкість. Гнучкий завдяки можливості використовувати різні алгоритми.

2) Модель GraphLab створена для обробки великих масивів даних, особливо для обробки великих графів. Надає можливість

робити ітераційні обчислення, гнучке планування(scheduling). Зосереджена на програмах, які використовують послідовний обчислювальний шаблон(coherent computational pattern). Цей шаблон кодує широкий спектр алгоритмів машинного навчання та сприяє ефективній реалізації.

Переваги.

Масштабований у виконанні задач обробки даних та машинному навчанні.

Здатний виразити складні обчислювальні залежності з графіком даних і забезпечити складні примітиви планування.

Механізм C ++ оптимізує GraphLab для використання багатоопоточності і асинхронного вводу-виводу.

Великий інструментарій для машинного навчання.

Дана модель має багато методів для машинного навчання. Користувача можуть також додавати свої алгоритми до GraphLab.

3) Фреймворк GraphX[7] – фреймворк для оброблення графів, підтримує велику кількість алгоритмів ітеративних графів. Це бібліотека Спарка, головною ціллю якої є розподілити потік даних фреймворку. Graphx ефективно розподіляє графіки табличних структур даних за рахунок використання нових ідей в своїх представленнях. Аналогічним чином, Graphx використовує в пам'яті обчислення і відмовостійкість за рахунок використання удосконалення систем потоку даних. Graphx також спрощує побудову графіка і процес трансформації, надаючи нові потужні операції. При використанні цих примітивів, можна реалізувати абстракції PowerGraph і Pregal за допомогою декількох рядків. Також можна завантажувати, перетворювати і обчислювати в інтерактивному режимі на масивних графіках.

Переваги.

Масштабованість. Може бути вбудований в Спарк і наслідувати властивість масштабування. На відміну від Pregel та GraphLab працює і з графами і з даними таблицях. Підтримує декларативну мову SQL.Програмісти не мають думати про ітерації в графі, а лише мають визначити програму вершин.

4. Технологія обробки потоків даних

Багато даних отримується в реальному часі і значну цінність складає те, як швидко дані приходять. Наприклад, необхідно відслідкувати помилки в своєму онлайн сервісі в секундах чи необхідно дізнатись, який користувач чи скільки користувачів відвідали певний сайт. Для швидкої передачі даних необхідно за лічені секунди проводити громіздкі обчислення і мати сотні вузлів, де поміщати дані. Такими обчисленнями займається система Storm.

Storm – це відкрита система розподілених обчислень в реальному часі. Оброблює потоки. Використовує топології, а не задачі, як система Хадуп. Має два види вузлів – «мастер» і «робітник». Вузол «мастер» запускає демона «Nimbus». Nimbus відповідає за розподілення коду по кластеру, присвоєнні задач машинам та моніторингу помилок.

На кожному вузлі «робітник» запускає демона, якого називають «Supervisor». Цей демон перевіряє чи не присвоїли машині задачу, починає і закінчує процеси. Кожен робочий процес – це підмножна топології. Працююча топологія складається з багатьох робочих процесів і поширюється на багатьох машинах.

Переваги.

Масштабованість. Легко додавати чи видаляти вузли із кластеру без порушення існуючих потоків даних.

Відмовостійкість. Гарантує, що дані будуть обрблені. Легкий у використанні. Розширяємість.

Дає можливість виклакати зовнішні функції, але потребує налаштування фреймворку для роботи із розширеннями.

Ефективність. Використовує велику кількість технік, включаючи зберігання обчислювальні структури даних в пам’яті. Дуже швидкий в обробці.

5. Висновки

Більш нові платформи програмування, як Storm, Spark Streaming мають більшість атрибутів, необхідних для ефективного аналізу великих масивів даних. Багато досліджень проводяться для розробки всіх алгоритмів машинного навчання для нових

систем. Для інтерактивного аналізу можна використовувати Storm, і Spark. Хадуп можна використовувати, наприклад, для аналізу логів, але немає сенсу використовувати для роботи із невеликою кількістю даних, бо продуктивність MapReduce значно зменшується. Також Хадуп має жорстке ділення ресурсів, що створює їх низьку утилізацію. Pregel зручний для роботи із графами(картами, наприклад), але вимагає багато пам'яті і у нього простоюють процесори із-за синхронізації. Також важливо зазначити, що частина із вище оглянутих платформ не вимагає знань у сфері паралелізму чи теорії графів, тобто користувач працює із інтерфейсом і не бачить самі алгоритми, не знає, які дані оброблюються, а які – ні і це може вплинути на обробку даних, подальші дані, і майбутню роботу з ними. Також важливою проблемою є захищеність даних. При роботі із цими платформами вшиті за умовчуванням алгоритми можуть аналізувати конфіденційні дані і передавати третім програмам/особам. Наостанок, не вистачає платформи, яка була б оптимальною і за її допомогою можна було аналізувати дані різного виду і різного розміру. Отже, робимо висновок, що платформ, які мають хоча б одну з таких рис - стабільно працюють, гарно масштабуються та ефективно оброблюють різні види даних досить широко представлені на ринку, але не вистачає платформи, яка б поєднала всі ці риси.

Список литературы

1. Albert Y. Zomaya. Handbook of Big data Technologies. – Australia, The University of New South Wales Eveleigh, 2017. – С. 70-76.
2. M. Frank, P. Roehrig, B. Pring. When machines do everything. Hot to get ahead in a workd of al, algorithms, bots, and big data.. - USA, New Jersey: 2017, с. 13-20
3. Jean-Pierre Dijcks. Oracle: Big Data for the Enterprise. – USA, Redwood Shores: 2013. – 10 с.
4. https://www.microsoft.com/en-us/research/project/dryad/
5. https://research.cs.wisc.edu/htcondor/
6. http://www.dcs.bbk.ac.uk/~dell/teaching/cc/paper/sigmod10/p135-malewicz.pdf
7. http://spark.apache.org/docs/latest/graphx-programming-guide.html

УДК 004.89

ОНИСЬКО П.І.

ЗАСТОСУВАННЯ НЕЙРОННОЇ МЕРЕЖІ ДЛЯ ПОБУДОВИ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ З ВИБОРУ ТУРИСТИЧНИХ НАПРЯМКІВ

Розглядається застосування алгоритмів нейронної мережі для побудови рекомендаційної системи з пошуку оптимальних туристичних напрямків за параметрами користувача. Описується принцип побудови рекомендаційної системи та її математична модель. Надано детальний опис та результати роботи системи.

The subject of this article is an application of neural network algorithms for creating recommender system to select travel destinations according to user settings. The construction principle of recommender system and its mathematical model are exposed. Article includes description and results of the recommender system.

1. Введення

В даній статті розглянуто практичне використання нейронних мереж у якості ключового елементу для побудови рекомендаційної системи на прикладі рекомендацій з вибору туристичних напрямків.

Основна задача будь-якої рекомендаційної мережі – зацікавити користувача в користуванні сервісом, на якому працює ця система, а також підвищити ймовірність того, що користувач скористається послугами та стане клієнтом сервісу. Завдяки бурхливому розвитку інформаційних технологій ринок туристичних послуг дуже стрімко переходить в Інтернет, через це з'являється багато різних Інтернет сервісів по підбору туристичних маршрутів, сервісів покупки авіаквитків, бронювання готелів тощо. Відповідно до цього основною задачею цих сервісів є виграш конкуренції у собі подібних, що можна досягти розробкою різних сторонніх допоміжних систем. Одним з найпрогресивніших наразі методів є розробка рекомендаційної системи на основі нейронної мережі [1].

2. Обґрунтування доцільності дослідження

На сьогоднішній день практично кожна система електронної комерції має в своєму арсеналі певні інстументи для побудови рекомендацій. Найчастіше це стосується різноманітних інтернет-магазинів, які в маркетингових цілях підбирають рекомендації під кожен товар для своїх користувачів.

Існує досить велика кількість рекомендації систем або мереж в системах пошуку кінофільмів, музики, певної літератури тощо [3]. Найвідомішими рекомендаційними системами такого типу є:

- LastFM;
- MovieLens;
- Neflix.

Однак, на сьогоднінній день немає достатньо відомого ресурсу для підбору таких рекомендацій у сфері туристичних послуг. Доцільно буде побудувати таку систему, яка може самостійно рекомендувати мандрівникам чи бажаючим провести відпустку найбільш релевантні варіанти туристичних напрямків. Такого роду систему можна використовувати туристичним компаніям як для інформаційну систему для своїх клієнтів, а також як маркетингову рекомендаційну систему для своїх постійних клієнтів.

3. Вибір та аналіз вхідних даних

Для обраного предметного середовища ключовою задачею стає вибір вхідних параметрів, за якими можна будувати нейронну мережу для рекомендаційного сервісу. Зазвичай всі мандрівники знають практично точно та можуть подати на вхід системі наступні параметри: кількість дорослих та дітей, максимально

допустимий бюджет подорожі, тривалість та дати подорожі. Цей перелік даних є мінімально необхідним для будь-яких туристичних сервісних систем. Відштовхуючись від цього, логічно прийти до висновку, що вхідними параметрами для аналізу та розрахунку рекомендацій можуть бути наступні:

- кількість днів до початку подорожі;
- дата подорожі;
- бюджет подорожуючих;
- кількість подорожуючих;
- тривалість подорожі/відпочинку.

Отже, задачею рекомендаційної мережі є побудова рейтингів популярності туристичних напрямків на основі вказаних вище вхідних параметрів, які надаються користувачами системи. Фактично мережа повинна надати відсортований за рекомендаціями список напрямків на основі аналізу поведінки користувачів.

Поведінкою користувача вважається вибір певного напрямку, яким він зацікавиться. Наприклад, це перехід на сторінку більш детальної інформації про туристичний напрямок зі списку запропонованих системою.

Параметрами нейронної мережі можуть бути різні числові значення. Фактично, це певний вектор $\mathbf{X}$, який містить у собі будь-яку кількість елементів (чим більша кількість цих елементів, тим довше бути працювати навчання мережі та отримання результатів з неї) [1].

- Представимо цей вектор у вигляді послідовності з 6 чисел, які будуть вхідними параметрами нейронної мережі. На підставі проведеного аналізу параметрів запитів в існуючих пошукових системах туристичного бізнесу були визначені наступні обмеження для цих вхідних параметрів:
- кількість днів до початку подорожі/відпочинку (ціле числове значення від 1 до 365);
- порядковий номер місяця, на який шукається подорож (ціле числове значення від 1 до 12);
- бюджет подорожі (ціле числове значення – максимальна кількість грошейу долларах, які готовий витратити мандрівник на свою подорож);
- кількість відпочиваючих дорослих (ціле числове значення від 1 до 5);
- кількість відпочиваючих дітей (ціле числове значення від 1 до 5);
- тривалість відпочинку (ціле числове значення від 1 до 21).

4. Постановка задачі в загальному випадку

Нехай розглядається певний тип відпочинку, за яким наявно n туристичних напрямків. За заданим вектором параметрів $\overline{\alpha} = \alpha_1 \dots \alpha_k$, де $k = 6$ необхідно надати певний вектор $\overline{\beta} = \beta_1 \dots \beta_n$, що являє собою вектор рекомендаційних оцінок для n напрямків у чисельному інтервалі від 0 до 1.

При цьому під туристичним напрямком розуміється країна, місто, регіон або певний ідентифікатор, який може бути рекомендований користувачеві.

Тип відпочинку – це певна множина однотипних туристичних напрямків.

5. Підхід до побудови рекомендаційної мережі

Як було зазначено вище, алгоритмом вирішення даної задачі є побудова нейронної мережі, яка буде адаптовуватись під заданий тип відпочинку (під адаптовуваністю розуміється набір відповідних тренувальних сетів для нейронної мережі) та самостійно навчатись на основі поведінки користувача.

Мережа приймає на вхід 6 параметрів, які були описані в розділі вище, далі інформація обробляється та видається на вихід у вигляді вектору з*n* елементів.

Розглянемо детальніше вихідний вектор системи. Припустимо, що туристична компанія може запропонувати користувачеві 10 різних країн для відпочинку на морі (*n=10*). Проставивши кожному напрямку відповідний порядковий номер,вихідний вектор матиме наступний вигляд:

$$\overline{\beta} = \beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7, \beta_8, \beta_9, \beta_{10}$$

Під кожним значенням β_i розуміється рекомендована оцінка у діапазоні від 0 до 1. Відсортувавши цей вектор, ми

отримаємо відповідні оцінки кожного з напрямків та їх список за популярністю.

Одним з головних завдань проектування нейронної мережі є задача створення початкових тренувальних сетів (до того часу, як мережа почне напряму взаємодіяти з користувачем та навчатись на його поведінці) [3].

Сам процес навчання являє собою процедуру підбору підстроювання параметрів нейронної мережі (вагових коефіцієнтів) відповідно до існуючих бажаних результатів. Таких підхід називається навчання з учителем.

Навчання мережі найчастіше проходить до тих пір, доки не досягається бажана похибка. Однак при великій кількості даних іноді досягти необхідно точного результату неможливо. Тому потрібно ще й наперед обмежувати максимальну кількість прогонів початкових тренувальних даних, щоб уникнути зациклювання або ж надто довгого процесу навчання нейронної мережі [7].

Припустимо, що надано інформацію про деяку поведінку користувачів на веб-сайті певної туристичної компанії, тобто наявна база даних про те, які сторінки (сторінки яких напрямків) обирались за певними параметрами, які були описані вище.

За цією базою даних необхідно сформувати максимально можливу кількість тренувальних сетів для нейронної мережі, тобто пари відношень вектора $\overline{\alpha}$ до вектору $\overline{\beta}$, які ми вважатимемо за правильними варіантами поведінки нейронної мережі.

Формат представлення цих векторів програмно буде представлено у наступних розділах.

6. Алгоритм побудови нейронної мережі

Розглянемо більш детально алгоритм роботи розроблюваної нейронної мережі.

Як відомо, штучний нейрон (елемент та основна складова нейронної мережі) складається з синапсів, по пов’язують входи нейрону з ядром, ядра нейрону, яке виконує обновку вхідних сигналів та аксону, що зв’язує нейрон з нейронами наступного рівня. Кожен синапс має свою певну вагу, яка визначає наскільки сильно впливає відповідний вхід нейрону на його стан [7]. Стан нейрону визначається за формулою:

$$S = \sum_{i=0}^{n} x_i w_i$$

де:

n - число входів нейрону (як було визначено вище, це число дорівнює 6),

x_i– значення і-го входу нейрону,

w_i– значення ваги і-го синапсу.

Далі визначається значення аксону нейрона. Це значення виводиться за формулою:

$$Y = f(S)$$

де f - певна обрана функція активації. Найбільш популярні та актуальні функції активації для нейронних мереж є наступні:

- сигмоїдальна функція;
- експоненційна функція;
- тангенціальна функція;
- квадратична функція;

Для нашої системи доцільно використати активаційну функцію сигмоїду:

$$f(x) = \frac{1}{1 + e^{-ax}}$$

$$f'(x) = af(x)(1 - f(x))$$

При зменшенні параметру a сигмоїд стає більш пологим, вироджуючись в горизонтальну лінію на рівні 0,5 при $a = 0$. При збільшенні a сигмоїд все більше наближається до функції одиничного скачка.

Для обраної задачі найбільш доцільним є використання нейронної мережі зворотного поширення, яка є потужним інструментом пошуку закономірностей, прогнозування та якісного аналізу. Таку назву мережі отримали через використовуваний алгоритм навчання, в якому похибка розповсюджується від вихідного рівня до вхідного, тобто в напрямку, протилежному напряму розповсюдження сигналу при нормальному функціонуванні мережі.

В загальному випадку задача навчання нейронної мережі зводиться до знаходження деякої функціональної залежності між вхідним та вихідним

векторами нейронів [7]. Така задача при обмеженому наборі вхідних даних має нескінченно велику множину рішень. Для обмеження простору пошуку при навчанні ставиться задача мінімізації цільової функції помилки нейронної мережі, яка знаходиться за методом найменших квадратів:

$$E(w) = \frac{1}{2}\sum_{j=0}^{p}(y_i - d_i)^{-ax}$$

де:

y_i–значення і-го виходу нейронної мережі,

d_i– цільове значення і-го виходу,

p– число нейронів на вихідному рівні.

Навчання мережі проводиться методом градієнтного спуску, тобто на кожній ітерації зміна ваги синапсів проводиться за формулою:

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}, (1)$$

де:

η – параметр, що визначає швидкість навчання.

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial y_i} \cdot \frac{dy_j}{dS_j} \cdot \frac{\partial S_j}{\partial w_{ij}}$$

де:

y_i – значення і-го виходу нейрону,

S_j – зважена сума вхідних сигналів.

При цьому множник

$$\frac{\partial S_j}{\partial w_{ij}} = x_i$$

де:

x_i – значення і-го входу нейрону.

$$\frac{\partial E}{\partial y_i} = \sum_k \frac{\partial E}{\partial y_k} \cdot \frac{dy_k}{dS_k} \cdot \frac{\partial S_k}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k} \cdot \frac{dy_k}{dS_k} \cdot w_{jk}^{(n+1)}$$

де:

k– число нейронів в слої $n+1$.

$$\delta_j^{(n)} = \frac{\partial E}{\partial y_j} \cdot \frac{dy_j}{dS_j}$$

В такому випадку можемо визначити рекурсивну формулу для визначення n-го рівня, якщо нам відомо значення слою n+1:

$$\delta_j^{(n)} = \left[\sum_k \delta_k^{(n+1)} \cdot w_{jk}^{(n+1)}\right] \cdot \frac{dy_j}{dS_j}, (2)$$

Знаходження останнього рівня, з якого можна почати розрахунок, не є складним, оскільки нам відомий цільовий вектор, тобто вихідний вектор з тих значень, які має видавати нейронна мережа при даному наборі вхідних даних.

$$\delta_j^{(n)} = (y_i^{(N)} - d_i) \cdot \frac{dy_j}{dS_j}, (3)$$

Запишемо формулу (3.6.1) у розкритому вигляді:

$$\Delta w_{ij}^{(n)} = -\eta \cdot \delta_j^{(n)} \cdot x_i^n, (4)$$

Розглянемо тепер послідовно повністю алгоритм навчання нейронної мережі:

подати на вхід нейронної мережі один з можливих векторів вхідних параметрів і визначити значення виходів нейронів нейронної мережі;

розрахувати для вихідного рівня нейронної мережі за формулою (3) і розрахувати зміну вагів вихідного рівня N за формулою (4);

розрахувати за формулою (4) зміну вагових коефіцієнтів $\Delta w_{ij}^{(n)}$ для остальниї рівнів;

скоректувати всі ваги нейронної мережі за формулою:

$$w_{ij}^{(n)}(t) = w_{ij}^{(n)}(t-1) + \Delta w_{ij}^{(n)}(t)$$

розрахувати помилку та якщо помилка не задовольняється, то перейти на перший крок.

7. Результати дослідження

Розглян

Результатами роботи нейронної мережі вважатимемо рейтинги рекомендацій, що оцінені в діапазоні від 0 до 1. Ці рейтинги і будуть рекомендаційними оцінками для вибору напрямку на задані параметри.

Проведемо тестування роботи нейронної мережі, обравши можливі країни, як множину туристичних напрямків, у які може порекомендувати система. Візьмемо 15 000 вхідних сетів для тренування нейронної мережі, та спробуємо наживо перевірити якість її роботи на введених нами даними.

Введем

Візьмемо такий набір вхідних параметрів:

- 50 днів до вильоту;
- Місяць - серпень;
- Бюджет – 1500;
- Кількість дорослих - 2;
- Кількість дітей - 0;
- Довжина подорожі – 10 днів;

Виконавши прогон нейронної мережі за даними параметрами, отримуємо результати, які зображено у таблиці 1.

Табл. 1. Результати роботи мережі

Напрямок	Оцінка
Іспанія	0.169
Італія	0.160
ОАЕ	0.147
Ізраїль	0.132
Кіпр	0.102
Греція	0.086
Португалія	0.076
Турція	0.032
Хорватія	0.030
Чорногорія	0.026
Франція	0.022
Тайланд	0.020
Єгипет	0.018
Грузія	0.016
Мальта	0.014
США	0.013
Болгарія	0.004
Росія	0.001
Австралія	0.001

Отримані результати свідчать, що найвищі рейтинги для обраних вхідних даних отримали такі туристичні напрямки, як Іспанія і Італія. Саме вони дозволяють в рамках визначеного бюджету, для обраних тривалості подорожі і пори року отримати якісні туристичні послдуги. У той же час низький рейтинг такого напрямку, як Австралія, обумовлений в першу чергу його високою реальною вартістю і неможливістю його практичної реалізації в рамках обраного бюджету.

В цілому можна зробити висновок, що отримані результати є адекватними реальній ситуації у сфері туристичного бізнесу, про що свідчить їх відповідність тим рейтингам туристичних напрямків, котрі можна знайти у інформаційних джерелах.

8.Висновки

Таким чином, у статті було розглянуто використання нейронних мереж для побудови рекомендаційної системи з вибору туристичних напрямків. Для цього було проведено відбір і аналіз вхідних даних для рекомендаційного сервісу, здійснено постановкузадачі для обраного предметного середовища, а також розроблено алгоритм побудови такої системи, що здатна адаптовуватись під заданий тип відпочинку.

Використання такої системи дозволить її клієнтам здійснювати ефективне планування туристичних напрямків з максимальним урахуванням їх побажань та можливостей. Це, в свою чергу,сприятиме отриманнюкористувачами якісних послуг іспиятиме розвитку туристичної галузі в цілому.

Список літератури

1. Michael D. Ekstrand, John T. Riedl and Joseph A. Konstan.K. Collaborative filtering recommender systems. 2001.

2. Introduction to Recommender Systems: Non-Personalized and Content-Based[Електронний ресурс] // Режим доступу: https://www.coursera.org/learn/recommender-systems-introduction.

3. Upendra Shardanand. Social Information Filtering for Music Recommendation. 1994.

4. Pascal Francq. Collaborative Search and Communities of Interest. Hershey 2005.

5. Адаптивний метод і алгоритм пошуку центрів кластерів за допомогою нейронної мережі / [Стех Ю.В., Файсал М.Е. Сардіх, Лобур М.В., Домброва М.С., Арцибасов В.Є.] // Моделювання та інформаційні технології. – 2010. – Вип.57.

6. Саймон Хайкин, Нейронные сети Полный курс. 2006

УДК

ОПРИШКО А.О.,
КОВАЛЮК Т.В.

МОДЕЛЮВАННЯ ТА ІНФОРМАЦІЙНО-ТЕХНОЛОГІЧНИЙ СУПРОВІД ІНДИВІДУАЛЬНИХ ОСВІТНІХ ТРАЄКТОРІЙ

Передбачується підхід, який може бути використаний для побудови освітніх траєкторій на основі вимог стейкхолдерів з використанням методів прийняття рішень та кластерізації даних. Підходу демонструється на прикладі підходу побудови освітніх дисциплін на основі методології Тюнинга [1].

Peredbachuyetsya approach that can be used to build educational paths based on the requirements of stakeholders using methods of making and clustering of data. Approach is demonstrated by the example of construction of educational disciplines approach based on tuning methodology [1].

1. Вступ

Побудова освітніх траєкторій є основним з важливих питань, яке виникає у освіті. Враховуючи набір вимог стейкхолдерів, які є оцінками компетенції заданих спеціальностей ІТ сфери, потрібно прийняти рішення про множину дисциплін, які будуть задовольняти цім вимогам. Такі рішення стають особливо важливі у наш час, тому що технології розвиваються дуже швидко, завдяки чому з'являються нові компетенції, які потрібно освоїти спеціалістам ІТ сфери. Це дає змогу підготувати висококваліфікованих спеціалістів, які будуть більше відповідати вимогам бізнесу, уникнути, або значно зменшити витрати бізнесу на підготовку спеціалістів та допоможе освітній системі державних та приватних вищих навчальних закладів бути більш конкурентоздатною, порівняно з різноманітними короткостроковим вузькоспеціалізованим курсам.

У зв'язку з цим актуальною науковою задачею є розробка моделей та методів прийняття рішень та кластеризації даних, що дозволить комплексно розв'язати задачу побудови освітніх траєкторій, яка допоможе збільшити якість освіти на основі вимог стейкхолдерів.

2. Кроки побудови освітніх дисциплін

В даної роботі пропоную наступні кроки процесу створення освітньої програми, які базуються на методології Тюнінга [1]:

1) розбиття стейкхолдерів за профілями компетенцій;
2) на основі вимог стейкхолдерів сформувати очікуваний результат навчання для кожного профілю компетенцій;
3) побудувати дисципліни, які формують бажані компетенції освітньої програми з урахуванням результатів виконання кроку 5 для попередніх ітерацій створення освітніх дисциплін;
4) формування отриманого результату навчання;
5) порівняння очікуваного результату навчання та отриманого результату навчання та знаходження дисциплін, для яких результати очікуваного та отриманого результатів навчання збіглися, а для яких ні.

3. Позначення

Для формалізації задачі введемо наступні позначення:

1) матриця співвідношення результату навчання, якій очікує стейкхолдер, та компетентностей CTk [5], де k_i ($i = \overline{1,n}$ де n – кількість компетентностей) – компетенції, які взято з Європейської рамки ІКТ-компетенцій 2.0 частина 1-3 [2-4], CT_j ($j = \overline{1,r}$ де r – кількість стейкхолдерів) – оцінка важливості компетенції j стейкхолдером, $CTk_{ij} = \overline{1,10}$ – оцінка важливості i компетенції j стейкхолдером;
2) матриця співвідношення дисциплін та компетентностей, що формує дисципліна Y, де k_i ($i = \overline{1,n}$ де n – кількість компетентностей) – компетенції, які взято з Європейської рамки ІКТ-компетенцій 2.0 частина 1-3 [2-4], d_j ($j = \overline{1,m}$ де m – кількість дисциплін у освітньому плані); Y_{ij} = 0, якщо результат навчання j не формує компетентність і та Y_{ij} = 1, якщо результат навчання j формує компетентність і.
3) Матриця співвідношення результатів навчання та компетентностей PHk, де k_i ($i = \overline{1,n}$ де n – кількість компетентностей) – компетенції, які взято з Європейської рамки ІКТ-компетенцій 2.0 частина 1-3 [2-4], PH_i ($j = \overline{1,z}$ де z – кількість експертів, які оцінюють результати навчання студентів), $PHk_{ij} = \overline{1,10}$ – оцінка j стейкхолдера, на скільки якісно i компетенція була засвоєна.

Схематично зв'язки між матрицями представлено на рис 1.

4. Задачі побудови освітніх дисциплін

Для побудови освітніх дисциплін на основі вимог стейкхолдерів необхідно вирішити наступні задачі:

1) визначення матриці CTk для кожного профілю компетентностей:
   a) знаходження профілів компетентностей на основі вимог стейкхолдерів;
   b) ранжування компетенції K ($k_i \in K$, $i = \overline{1,n}$ де n – кількість компетентностей) для кожного профілю;
2) побудова матриці Y для кожного профілю компетентностей:
   a) визначення матриці кореляцій компетентностей A_{ij} ($i,j = \overline{1,n}$ де n – кількість компетентностей);
   b) побудова векторів освітніх дисциплін враховуючі D^+ та D^- з кроку 4 для попередніх ітерацій створення освітніх дисциплін;
3) побудова матриці PHk для кожного профілю компетентностей;
4) порівняння CTk та PHk та знаходження дисциплін, для яких результати очікуваного та отриманого результатів навчання збіглися ($D^+: {d_i}^+ \in D^+$, $i = \overline{1,m^+}$ де m^+ – кількість дисциплін, для яких результати очікуваного та отриманого результатів навчання збіглися) та не збіглися ($D^-: {d_j}^- \in D^-$, $j = \overline{1,m^-}$ де m^- – кількість дисциплін, для яких результати очікуваного та отриманого результатів навчання не збіглися).

5. Розрахунок кількості дисциплін

Вимоги МОН [6] до кількості компетенцій:

- <= 25 професійних компетенцій (всього 39 e-CF [2-4]);
- <= 15 загальних компетенцій (всього за TUNING їх 31 [1]).

ВНЗ самостійно визначає кількість та склад навчальних дисциплін, визначаючі:

1) максимальна кількість дисциплін в семестрі (7 для КПІ[7]);
2) кількість кредитів на дисципліну (4-5 для КПІ[7]);
3) тривалість семестру в тижнях (18 для КПІ[7]).

На основі цих даних можна визначити максимальну кількість предметів, яким може навчатися бакалавр в КПІ, отримаємо 56 дисциплін (8 семестрів, 7 дисциплін в семестрі). За положенням

про організацію освітнього процесу в НТУУ «КПІ» [7] 40% дисциплін, яким навчається студент у ВНЗ мають бути нормативними, тобто 23 дисциплін. Тобто має бути список ненормативних дисциплін, з яких студент мав би змогу вибрати дисципліни до свого індивідуального плану. Кількість цих дисциплін має бути не меншим ніж 32, тому мінімальна кількість дисциплін, з яких можна вибирати складає 88 дисципліни.

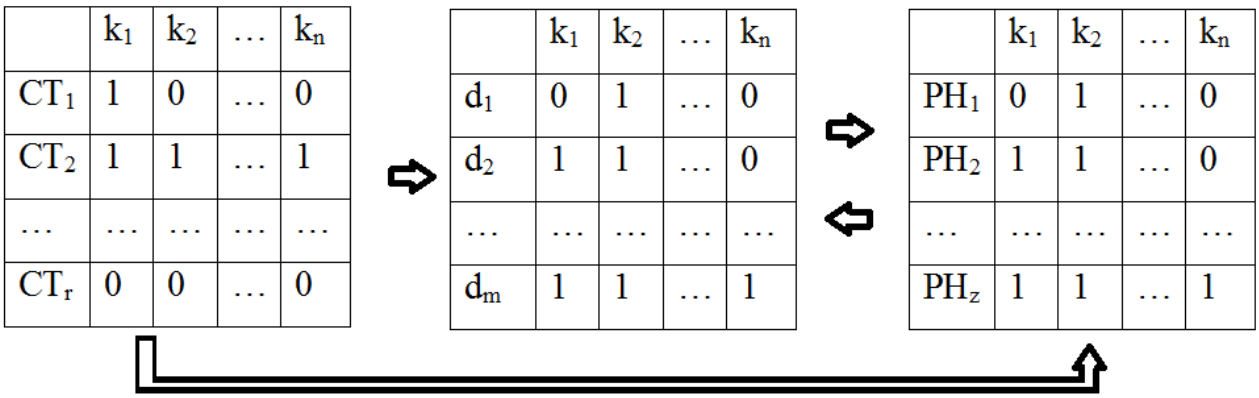


Рис. 1. Зв'язки між матрицями CTk, Y та PHk.

6. Висновок

На прикладі побудови освітніх дисциплін за методологією Тюнинг було продемонстровано підхід, який надає можливість побудови освітніх дисциплін на основі вимог стейкхолдерів.

Сучасна освіта у сфері ІТ не може справлятися з зростаючими потребами бізнесу у кваліфікованих кадрах, тому що освітня система державних та приватних вищих навчальних закладів не вспіває реагувати на зміни курсу бізнесу, завдяки чому освіта програє короткостроковим вузькоспеціалізованим курсам, які дають змогу оволодіти найбільш актуальними компетенціями. Також бізнес витрачає колосальні кошти на те, щоб підготувати тих, які тільки почали перши кроки освоєння своїх спеціальностей, які можна було б уникнути, якщо б освіта взяла цю відповідальність на себе.

У зв'язку с цим є потреба в методах побудови освітніх дисциплін, які враховують вимоги стейкхолдерів до компетентностей, якими потрібно оволодіти. Ефективні методи побудови освітніх дисциплін значно зменшать кошти, які витрачають на освіту, за рахунок скорочення кількості дисциплін, які не актуальні, або не мають великого значення для оволодіння деякої спеціальністю. Також з допомогою цього методу є можливість контролювати процес підготовки спеціалістів та редагувати освітні траєкторії для отримання спеціалістів с необхідним набором компетентностей та необхідним рівнем їх оволодіння, що дозволить не витрачати кошти на додаткове навчання.

Список литературы

1. TUNING (для ознайомлення зі спеціальними (фаховими) компетентностями та прикладами стандартів. [Електронний ресурс]. Режим доступу: http://www.unideusto.org/tuningeu/.
2. CEN WorkShop ICT Skills. European e-Competence Framework 2.0 – Part 1: A common European framework for ICT Professionals in all industry sectors [Текст] / CEN WorkShop ICT Skills // Пер. с анл. – М.: ТОВ «ІАОЦ», 2011. – 78 с.
3. CEN WorkShop ICT Skills. European e-Competence Framework 2.0 – Part 2: User guidelines for the application of the European e-Competence Framework 2.0 [Текст] / CEN WorkShop ICT Skills // Пер. с анл. – М.: ТОВ «ІАОЦ», 2011. – 33 с.
4. CEN WorkShop ICT Skills. European e-Competence Framework 2.0 – Part 3: Building the e-CF -a combination of sound methodology and expert contribution [Текст] / CEN WorkShop ICT Skills // Пер. с анл. – М.: ТОВ «ІАОЦ», 2011. – 28 с.
5. Захарченко В.М. Розроблення освітніх програм. Методичні рекомендації [Текст] / В.М. Захарченко, В.І. Луговий, Ю.М. Рашкевич, Ж.В. Таланова // За ред. В.Г. Кременя. – К. : ДП «НВЦ «Пріоритети», 2014. – 120 с.
6. Закон України «Про вищу освіту»/ [Електронний ресурс]: http://zakon2.rada.gov.ua/laws/show/2984-14 (15.10.2014).
7. Тимчасове положення про організацію освітнього процесу в НТУУ «КПІ» [Текст] / Уклад.: В. П. Головенкін (розд.: 1-8, 10, 12), С. В. Мельниченко (розд.: 9, 11); за заг. ред. Ю.І. Якименка. – К.: НТУУ «КПІ», 2015. – 102 с.

УДК 004.023

ОСІПЕНКО Д.С.,
ОЛІЙНИК Ю.О.

ЗАСТОСУВАННЯ ЗАДАЧІ КЛАСИФІКАЦІЇ ТА ГІПЕРПРОСТОРОВОЇ КЛАСТЕРИЗАЦІЇ ДЛЯ ПОБУДОВИ ПЕРСОНАЛІЗОВАНОГО РОЗКЛАДУ ДНЯ ДЛЯ ПІДТРИМКИ ВЕДЕННЯ ЗДОРОВОГО СПОСОБУ ЖИТТЯ

В даній статті розглянуто застосування задачі класифікації та гіперпросторової кластеризації для побудови персоналізованого розкладу дня для підтримки ведення здорового способу життя. Виділені важливі фактори, для побудови розкладу та описаний алгоритм персоналізації на основі п'ятивимірного простору. Описано використання відсоткової рангової шкали для обрахування прогресу користувача.

Ключові слова: ЗАДАЧА КЛАСИФІКАЦІЇ, ГІПЕРПРОСТОРОВА КЛАСТЕРИЗАЦІЯ, ЗДОРОВИЙ СПОСІБ ЖИТТЯ, ВІДСОТКОВА РАНГОВА ШКАЛА.

This article is concerned with applying classification and hyperspace clustering for creating personal schedule to maintain a healthy lifestyle. Marked important values for schedule and created the algorithm of personalization, which based on five-dimensional space. Described usage of percentage scale of measure for calculating user progress.

Keywords: CLASSIFICATION PROBLEM, HYPERSPACE CLUSTERING HEALTHY LIFESTYLE PERCENTAGE RANK SCALE.

1. Задача класифікації

Задача класифікації – це задача розбиття множини об'єктів або спостережень на апріорно задані групи, називані класами, всередині кожної з яких вони вважаються схожими один на одного, та мають приблизно однакові властивості й ознаки [1]. При цьому рішення здійснюється на основі аналізу значень атрибутів (ознак). Під час аналізу необхідно визначити, до якого з відомих класів відносять досліджувані об'єкти, тобто як їх класифікувати. Зустрічається 2 види класифікації - одновимірна (за однією ознакою) і багатовимірна (за двома і більше ознаками).

Задачу класифікації розглядають як задачу визначення значення одного з параметрів аналізованого об'єкта на підставі значень інших параметрів. Досліджуваний параметр часто називають залежною змінною, а параметри, що беруть участь у його визначенні - незалежними змінними.

Процес класифікації складається із двох етапів: конструювання моделі та її використання.

1. Конструювання моделі: опис множини визначених класів.

- кожен приклад набору даних відноситься до одного визначеного класу;
- на даному етапі використовується навчальна множина, на ній відбувається конструювання моделі;
- отримана модель представлена класифікаційними правилами, деревом рішень або математичною формулою.

2. Використання моделі: класифікація нових або невідомих значень.

- Оцінка правильності (точності) модель:

а) Відомі значення з тестового прикладу порівнюються з результатами використання отриманої моделі.

б) Рівень точності - відсоток правильно класифікованих прикладів в тестовій множині.

в) Тестова множина, тобто множина, на якій тестується побудована модель, не повинна залежати від навчальної множини.

- Якщо точність моделі допустима, можливе використання моделі для класифікації нових прикладів, клас яких невідомий.

Для отримання максимально точної моделі класифікації до навчальної вибірки пред'являються такі основні вимоги:

- кількість об'єктів, що входять до вибірки, має бути досить великою величиною;
- до вибірки мають входити об'єкти, що представляють усі можливі класи у

задачі класифікації або всю область значень у задачі регресії;
- для кожного класу в задачі класифікації або кожного інтервалу області значень у задачі регресії вибірка має містити достатню кількість об'єктів.

Основні методи, що застосовуються для вирішення задач класифікації:
- класифікація за допомогою дерев рішень;
- байєсівська (наївна) класифікація;
- класифікація за допомогою штучних нейронних мереж;
- класифікація методом опорних векторів;
- статистичні методи, зокрема, лінійна регресія;
- класифікація за допомогою методу найближчого сусіда;
- класифікація CBR-методом;
- класифікація за допомогою генетичних алгоритмів.

Оцінка точності класифікації може проводитися за допомогою крос-перевірки. Крос-перевірка (Cross-validation) - це процедура оцінки точності класифікації на даних з тестової множини, що також називають крос-перевірочною множиною. Точність класифікації тестової множини порівнюється з точністю класифікації навчальної множини. Якщо класифікація тестової множини дає приблизно такі ж результати по точності, як і класифікація навчальної множини, вважається, що дана модель пройшла крос-перевірку.

Оцінювання класифікаційних методів слід проводити, виходячи з таких характеристик: швидкість, роботоздатність, інтерпритованість, надійність.
- Швидкість характеризує час, який потрібен на створення моделі і її використання.
- Роботоздатність, тобто стійкість до будь-яких порушень вихідних передумов, означає можливість роботи з зашумленими даними і пропущеними значеннями в даних.
- Інтерпритованість забезпечує можливість розуміння моделі аналітиком.
- Надійність методів класифікації передбачає можливість роботи цих методів при наявності в наборі даних шумів і викидів.

Для вирішення задачі побудови персоналізованого розкладу дня для підтримки ведення здорового способу життя. Спочатку потрібно вирішити задачу класифікації та до кожного класу розробити шаблон розпорядку дня. Застосований метод для вирішення поставленої задачі - дерево рішень. Даний метод припускає виконання п’яти етапів:

Етап 1. Формулювання завдання.

Насамперед необхідно відкинути всі фактори, що не стосуються проблеми, а серед безлічі тих, що залишилися, виділити істотні і несуттєві. Це дозволить привести опис завдання прийняття рішення у форму, що підлягає аналізу. Повинні бути виконані такі основні процедури: визначення можливостей збору інформації для експериментування і реальних дій; складання переліку подій, що з певною ймовірністю можуть відбутися; установлення тимчасового порядку розташування подій, у наслідках яких міститься корисна і доступна інформація, і тих послідовних дій, які можна розпочати.

Етап 2. Побудова дерева рішень.

Етап 3. Оцінка ймовірностей станів середовища, тобто зіставлення шансів виникнення кожної конкретної події. Слід зазначити, що вказані ймовірності визначаються або на підставі наявної статистики, або експертним шляхом.

Етап 4. Установлення виграшів (чи програшів, як виграшів зі знаком мінус) для кожної можливої комбінації альтернатив (дій) і станів середовища.

Етап 5. Вирішення завдання.

В умовах вирішення задачі класифікації користувачів для побудови персоналізованого розкладу дня перший етап значно спрощується, тому що усі фактори будуть суттєвими. Фактори: вік, стать, вага, ріст та фізична активність.

Вік має чотири категорії: школяр, підліток, дорослий та людина похилого віку.

За вагою, ростом та віком можна обчислити індекс маси тіла.

Індекс маси тіла поділяється на: нормальний, менше норми, більше норми (надлишок ваги, зайва вага та сильне ожиріння).

І останній важливий фактор, що впливає на те, до якого класу буде віднесений користувач - фізична активність, що вимірюється коефіцієнтом фізичної активності та

поділяється на дуже малу, легку, середню, велику і дуже велику.

Далі обрахуємо декартовий добуток усіх факторів за формулою:

$$X_1 \times X_2 \times ... \times X_n = \{(x_1, x_2, ..., x_n) | x_1 \in X_1, x_2 \in X_2, ..., x_n \in X_n\}$$

Отримавши декартовий добуток множин виділених факторів, ми і отримаємо усі класи та відповідні значення факторів за якими потім будемо визначати клас нового користувача.

2. Метод персоналізації шаблонного розкладу дня для підтримки ведення здорового способу життя

Для персоналізації розкладу для підтримки ведення здорового способу життя виділимо основні важливі компоненти дня, що впливають на організм: фізична активність, сон, баланс води, базовий рівень метаболізму, спожита енергія за день. Усе ці фактори можна відобразити чисельно. Фізична активність поділяється на категорії, яким співвідносяться такі числові значення коефіцієнта фізичної активності (КФА):

- дуже мала: люди, які дуже мало рухаються, КФА = 1,2;
- легка: люди, які виконують легкі вправи 1-3 дні на тиждень, КФА = 1,375;
- середня: люди, що виконують середньої важкості вправи, а саме 3-5 дні на тиждень, КФА=1,55;
- велика: що виконують важкі вправи 6-7 днів на тиждень, КФА = 1,725;
- дуже велика: люди, що виконують дуже важкі вправи двічі на день, або працівники з великими фізичними навантаженнями, КФА = 1,9. [2]

Сон вимірюється в годинах, але ще також має коефіцієнт його якості.

Водний баланс обраховується як різниця між добовою потребою води в організмі та спожитою. У здорової людини ця різниця має дорівнювати нулю.

Базовий рівень метаболізму - це кількість калорій, яку організм спалює за добу у стані спокою.

Спожита енергія за день - це кількість отриманої енергії від спожитої їжі за день. [3]

Маємо п’ять основних факторів. Тому побудуємо або спробуємо уявити п’ятивимірний простір з осями виділених факторів. Для простоти розуміння будемо називати його простором варіантів розкладу. В даному просторі кожен користувач буде займати якусь точку.

Для кожного сформованого класу після проведення задачі класифікації отримаємо класи, які будуть займати певний простір у просторі варіантів розкладу. Для кожного класу потрібно виділити абстрактного ідеального користувача. Цей користувач матиме певний окіл у просторі, значення якого будуть задовільняти ідеальні показники виділених вище компонентів.

Побудуємо вектор відстані від реального користувача до ідеального - це і буде найкоротший шлях який має пройти користувач, щоб досягнути ідеального результату.

В кінці кожного дня шлях буде перебудовуватися та формуватися наступний розпорядок дня для досягнення поставленого результату.

Для того, щоб результати були більш ефективними будемо використовувати дані про користувача не тільки за один попередній день, а за сім. Так ми зможемо перехід до ідеальних показників зробити більш плавним та комфортнішим.

3. Відсоткова рангова шкала

Відсоткову рангову шкалу будемо використовувати для наочної демонстрації відносної позиції користувача. Таку шкалу поділяє на дві частини медіана – п’ятдесят відсотків. Значення більше медіани вказує на результати, вищі від середніх, і навпаки.

Сто відсотків – це відстань від ідеального користувача до найгіршого в межах одного класу.

Відстань в n-мірному просторі між двома точками визначається за формулою:

$$l = \sqrt{\sum_{k=1}^{n} (x_k - y_k)^2}$$

Відсоткова рангова шкала не дає змоги обчислити середню арифметичну величину чи величину розсіювання, однак визначає позицію досліджуваного всередині певної групи

Список літератури

1. Померанцев А. Класифікація [Електронний ресурс] : Російське хемометричне суспільство. – 2011. – Режим доступу: http://rcs.chemometrics.ru/Tutorials/classification.htm#Ch1.3.
2. Здоровий спосіб життя [Електронний ресурс] : 4BRAIN – 2012-2017. – Режим доступу: https://4brain.ru/zozh/.
3. Гуров В.А. Здоровий спосіб життя: наукові уявлення і реальна ситуація / / Валеологія. - 2006. - № 1. - С. 53.

УДК 004.93(015.7)

ПРОХОРОВА К.С.,
ГУЛЯНИЦЬКИЙ Л.Ф.

МАТЕМАТИЧНА ПОСТАНОВКА ТА ОГЛЯД МЕТОДІВ РОЗВ'ЯЗУВАННЯ ЗАДАЧІ ПОБУДОВИ ТУРИСТИЧНИХ МАРШРУТІВ

Розглянуто варіанти постановок задачі побудови туристичних маршрутів. Обґрунтовано вибір постановки задачі для досліджуваної проблеми і наведено математичну модель. Наведено короткий огляд методів розв'язання для обраної постановки задачі.

Overview of problem statements for tourist trip design problem is covered. Reasoning for choosing particular problem statements and mathematical model are provided. A brief outline of solving methods for chosen problem statement is given.

1. Вступ

Розвиток туризму має позитивний вплив на економічну ситуацію країни. На жаль, ринок туристичних послуг в Україні знаходиться в кризовому стані, незважаючи на наявний рекреаційний потенціал. Світова спільнота мало знайома з туристичними об'єктами України. Впровадження нових рішень, включаючи рішення з інформаційних технологій, необхідні для покращення існуючої ситуації. Такого роду заходом є впровадження автоматизованих систем для планування туристичних маршрутів, що ставлять на меті ознайомити своїх користувачів з якомога більшою кількістю пам'яток.

Створення такого роду програмного забезпечення тісно пов'язане з розв'язанням задачі побудови туристичних маршрутів, що відома в літературі як Tourist trip design problem (TTDP) [1]. Вона полягає у складанні маршрутів, що максимально задовільнять побажання туристів відвідати певні пам'ятки, враховуючи різноманітні обмеження, пов'язані з цими пам'ятками чи побажаннями туристів. Постановки задачі TTDP варіюються в залежності від того, наскільки математична модель повинна відображати умови реального світу і які додаткові обмеження накладаються. Очевидно, ступінь відповідності умовам реального світу і врахування додаткових умов позначаються на складності математичної моделі задачі. В наступному розділі буде наведено огляд задач, у термінах яких можна сформулювати TTDP в порядку зростання складності самих задач і їх математичних моделей відповідно.

2. Огляд постановок задачі TTDP

TTDP можна сформулювати у термінах задачі Orienteering problem (OP). OP задана на графі, в якому вершинами є пам'ятки, а ребрами – маршрути між ними. Ряд розширень даної задачі дозволяють моделювати додаткові обмеження і умови. OP – найпростіше трактування, в якому вершинам приписана корисність їх відвідування. Необхідно побудувати маршрут максимальної корисності від вершини до вершини (початкова і кінцева вершини задані), який обмежений в часі певною величиною. В [2] наведено класичне і два нові формулюваннями цієї проблеми. Для побудови декількох маршрутів сумарної максимальної вартості, що не перетинаються, постановка формується в термінах задачі Team orienteering problem (TOP) [3]. Таке формулювання задачі дозволяє складати маршрути для турів на декілька днів. Якщо задані обмеження на час, коли допустиме відвідування пам'яток, то задача постановка формується у термінах Orienteering problem with time windows (OPTW) [4]. Коли час на переміщення між пам'ятками залежить від моменту початку переміщення, то розглядається задача Time-dependent orienteering problem TDOP) [5]. Якщо туристичні маршрути будуються по дорогах, а не конкретних місцях, то постановку задачі можна сформувати у термінах Orienteering arc routing problem (OAP) [6]. Прикладом таких

доріг є видові ділянки. OAP задана на направленому графі, де корисність вказана не для вершин, а для дуг. Крім того, на відміну від OP, можна задати дуги, що обов'язково повинні бути включені в маршрут. Всі варіації, що були наведені для OP вище, можна застосувати і до задачі OAP. Їх огляд, математичні постановки і методи розв'язування наведено в [6].

Коли туристичний маршрут включає і місця і дороги, то таку задачу можна сформувати у термінах Mixed orienteering problem (MOP). Для цього випадку корисності визначені і на дугах, і на вершинах. Подібна задача зустрічається у літературі і під іншими назвами [7-10]. В цих роботах розглядаються такі задачі: [7] – задача Bus touring problem, [8] – задача Outdoor activity tour, [9] – задача Outdoor activity tour suggestion problem, [10] – задача Most attractive cycle tourist path problem.

Далі буде наведено обгрунтування вибору постановки задачі у термінах Mixed team orienteering problem with time windows (MTOPTW), математичну модель і методи розв'язування.

3. Постановка задачі TTDP у термінах MTOPTW

Найчастіше туристичні маршрути включають як вулиці, так і пам'ятки, а час, коли можна відвідувати деякі з них, лежить у певному часовому інтервалі, тому для розв'язування задачі TTDP обрано постановку задачі в термінах Mixed orienteering problem with time windows (MOPTW). Враховуючи те, що маршрут, побудований в результаті розв'язування MOPTW, включає лише обмежений набір місць (один маршрут), що цікавлять туристів, доцільніше розглянути побудову декількох туристичних маршрутів одночасно, причому вони не повинні перетинатися.

Ця умова враховується, якщо розширити MOPTW до MTOPTW.

Зазначена задача є NP-складною. Це твердження випливає з того, що MTOPTW являє собою модифікацію задачі OP, а в [11] було показано, що OP є NP-складною задачею. Це означає, що на великих розмірностях дана задача не може бути ефективно розв'язана точними методами, тому для її розв'язування застосовуються наближені методи.

Оскільки MTOPTW є MOP з додатковими обмеженнями, то методи, що беруться в основу для їх розв'язування, однакові. Розглянемо підходи, що застосовувались для MOP. В [7,8] для розв'язування задачі використано евристичні методи. В [9] задача сформульована як задача цілочислового програмування і розв'язана за допомогою комерційного програмного забезпечення для розв'язування задач цілочислового програмування. В [7-9] задача була розв'язана для випадку орієнтованого графу. В [10] наближеними методами задача була розв'язана для орієнтованого і неорієнтовного графів, крім того, було надано алгоритм зведення MOP до AOP. Тобто, для розв'язування MOP можна використати довільний евристичний алгоритм, застосовний для розв'язування AOP, і отримати результат без втрати точності внаслідок проведеної трансформації. В [10] наведено трансформацію MOP до OP для направленого графа. В [12] наведено розв'язування для розширеної версії MOP для ненаправленого графа, у якого ціна ребра відрізняється при обході в різних напрямках. Для цього використано такі метаевристики: повторюваний локальний пошук та імітаційний відпал (Simulated annealing).|

4. Математична модель MTOPTW

Наведемо математичну постановку [12] для розв'язування задачі MOPTW. Задача задана на неорієнтовному мультиграфі, в якому витрати на проходження ребер у різних напрямках можуть відрізнятись. У графі $G=(V,E)$ множина V являє собою множину вершин, а $E-$ множину ребер. Кожне ребро може обходитись у двох напрямках e^+ і e^-. Початок і кінець для кожного напрямку $d\in\{-,+\}$ позначається як $h(e^d)$ і $t(e^d)$ відповідно. В множині ребер E виділяються дві множини: E' і E''. Множина E' включає ребра, що є найкоротшими шляхами між довільними вершинами з V і не впливають на загальну

корисність маршруту. Множина $E^{''}$ включає ті ребра, що мають корисність. Між двома вершинами завжди розташовані два ребра з множини $E^{'}$ і $E^{''}$, навіть якщо ребро, що має корисність, є найкоротшим шляхом між вершинами. Для кожної вершини $u \in V$ можливо два варіанти. Перший – коли через вершину проходить маршрут, але вона не відвідується. Другий – коли вершина відвідується, що впливає на загальну корисність маршруту.

Дано ціле значення K, що позначає кількість маршрутів (турів), які буде сконструйовано. Кожна вершина чи ребро асоційовані з K часовими вікнами $\left[O_x^i, C_x^i\right], i = 1,...,K$, де O_x^i – час відкриття, а C_x^i – час закриття i-го часового вікна для вершини чи ребра x. Відвідування вершини можливе тільки в рамках її часового вікна. Щоб маршрут просто проходив через цю вершину, не потрібно враховувати обмеження часових вікон. Для кожного туру $W_i, i = 0,...,K-1$, задано початкову вершину sl_i, якій відповідає час початку st_i, і вершину в якій тур завершується – el_i з часом завершення $et_i (sl_i, el_i \in V)$. Припускається що для кожного туру тільки одне з K вікон доступне, тобто множина $\left[O_x^j, C_x^j\right] \cap [st_i, et_i], i, j = 1,...,K$, не містить елементів при $i \neq j$. Це потрібно для того, щоб можна було моделювати тури послідовні у часі, наприклад на K днів вперед.

Допустимий розв'язок задачі MTOPTW складається з K маршрутів $W_0,...,W_{K-1}$, де кожен маршрут подано у вигляді $W_i = (w_0^i, w_1^i,..., w_{l_i-1}^i)$, причому $w_0^i = sl_i$, $w_{l_i-1}^i = el_i$. Час прибуття у першу вершину туру – st_i, а час прибуття у останню вершину не більше et_i. Для кожного ребра $\{w_m^i, w_{m+1}^i\}$ та тих вершин w_m^i, що відвідуються, а не просто присутні у маршруті, час проходження повинен лежати в рамках заданих часових вікон. Метою є знаходження таких допустимих маршрутів, у яких сумарна корисність була б максимальною. Враховується, що корисність від відвідування вершини і проходження по ребру можна отримати лише раз.

У математичній моделі присутні такі додаткові змінні:

– $y^k(e^d)$ – бінарна змінна значення якої 1, коли ребро e проходиться у напрямку $d \in \{-,+\}$ в маршруті $W_k, k = 0,...,K-1$, і 0 в іншому випадку;

– z_u^k – бінарна змінна, значення якої 1, коли вершина u відвідується у маршруті $W_k, k = 0,...,K-1$, і 0 – в іншому випадку;

– $p^k(e^d)$ – бінарна змінна, значення якої 1, коли проходження ребром e у маршруті W_k відбувається у напрямку d одразу після відвідування вершини $h(e^d)$, і 0 у іншому випадку;

– $q^k(e^d)$ – бінарна змінна, значення якої 1, коли у маршруті у маршруті W_k вершина $t(e^d)$ відвідана перед проходженням ребра e в напрямку d, і 0 у іншому випадку;

– $r^k(e_1^{d_1}, e_2^{d_2})$ – бінарна змінна, що визначена лише для ребер e_1, e_2 з $h(e_1^{d_1}) = t(e_2^{d_2})$. Її значення дорівнює 1 тільки коли ребро e_1 проходиться у напрямі d_1 в маршруті W_k перед тим, як ребро e_2 було пройдено в напрямі d_2 і вершина $h(e_1^{d_1})$ не відвідується в цей момент, а проходиться. $start_u$ – дійсна змінна, що відображає початок візиту в вершину u;

– $start_e$ – дійсна змінна, що відображає початок проходження по ребру e;

– M – велике число порівняно з параметрами задачі;

– $[K]$ – множина $\{0,1,...,K-1\}$.

Щоб уникнути часткових випадків у формулюванні задачі, кінцевим вершинам ставлять нульову корисність. Крім того, кінцева і початкова вершини не співпадають. Якщо ці умови не

виконуються, то вводяться фіктивні вершини. Враховуючи все вищесказане, можна навести математичну модель задачі MTOPTW. Цільова функція має вигляд (1):

$$\max P = \sum_{k=0}^{K-1} \sum_{u \in V} P_u z_u^k + \sum_{k=0}^{K-1} \sum_{e \in E \wedge d \in \{-,+\}} P_e y^k(e^d). \quad (1)$$

Для того, щоб врахувати для кожного туру початок в st_i і закінчення в et_i, вводяться такі два обмеження:

$$\sum_{e \in E \wedge d \in \{-,+\} \wedge t(e^d) = sl_k} y^k(e^d) + 1 = \sum_{e \in E \wedge d \in \{-,+\} \wedge h(e^d) = sl_k} y^k(e^d),$$

$$\sum_{e \in E \wedge d \in \{-,+\} \wedge t(e^d) = el_k} y^k(e^d) + 1 = \sum_{e \in E \wedge d \in \{-,+\} \wedge h(e^d) = el_k} y^k(e^d), \quad k \in [K].$$

Щоб гарантувати проходження ребрами не більше разу в обох напрямках, вводиться наступне обмеження:

$$\sum_{d \in \{-,+\} \wedge k \in [K]} y^k(e^d) \le 1, \forall e \in E.$$

Обмеження (2) є умовою збереження потоку. Обмеження (3-10) забезпечують узгодженість змінних $p^k(e^d)$, $q^k(e^d)$, $r^k(e_1^{d_1}, e_2^{d_2})$ зі змінними $y^k(e^d)$ і z_u^k. Обмеження (11-13) забезпечують узгодженість переходу від вершини до суміжного ребра. Наприклад, обмеження (11) забезпечує, що відвідування вершини може відбутись лише після того, як пройдене ребро перед цією вершиною.

$$\sum_{e \in E \wedge d \in \{-,+\} \wedge h(e^d) = u} y^k(e^d) = \sum_{e \in E \wedge d \in \{-,+\} \wedge h(e^d) = u} y^k(e^d), \forall u \in V - \{sl_k, el_k\} \in [K]. \quad (2)$$

$$p^k(e^d) = 0, \forall k \in [K], d \in \{+,-\}, e \in E \quad якщо\ h(e^d) = sl_k. \quad (3)$$

$$q^k(e^d) = 0, \forall k \in [K], d \in \{+,-\}, e \in E \quad якщо\ t(e^d) = el_k. \quad (4)$$

$$p^k(e^d) \le y^k(e^d), \forall e \in E, d \in \{+,-\}, k \in [K]. \quad (5)$$

$$q^k(e^d) \le y^k(e^d), \forall e \in E, d \in \{+,-\}, k \in [K]. \quad (6)$$

$$z_u^k = \sum_{e \in E \wedge d \in \{-,+\} \wedge h(e^d) = u} p^k(e^d), \forall u \in V - \{sl_k\}, k \in [K]. \quad (7)$$

$$z_u^k = \sum_{e \in E \wedge d \in \{-,+\} \wedge t(e^d) = u} q^k(e^d), \forall u \in V - \{el_k\}, k \in [K]. \quad (8)$$

$$y^k(e^d) = \sum_{e' \in E \wedge d' \in \{-,+\} \wedge h(e'^{d'}) = t(e^d)} r^k(e'^{d'}, e^d) + q^k(e^d), \forall e \in E, d \in \{-,+\}, k \in [K]. \quad (9)$$

$$y^k(e^d) = \sum_{e' \in E \wedge d' \in \{-,+\} \wedge h(e^d) = t(e'^{d'})} r^k(e^d, e'^{d'}) + p^k(e^d), \forall e \in E, d \in \{-,+\}, k \in [K]. \quad (10)$$

$$start_e + T(e^d) - start_{h(e^d)} \le M(1 - p^k(e^d)), \forall e \in E, d \in \{-,+\}, k \in [K]. \quad (11)$$

$$start_{t(e^d)} + T_{t(e^d)} - start_e \le M(1 - q^k(e^d)), \forall e \in E, d \in \{-,+\}, k \in [K]. \quad (12)$$

$$start_e + T(e^d) - start_{e'} \le M(1 - r^k(e^d, e'^{d'})), \forall e, e' \in E, d, d' \in \{-,+\}$$
$$якщо h(e^d) = t(e'^{d'}), k \in [K]. \quad (13)$$

$$O_e^k - M(1 - y^k(e^d)) \le start_e, \forall e \in E, d \in \{-,+\}, k \in [K]. \quad (14)$$

$$start_e + T(e^d) \le C_e^k + M(1 - y^k(e^d)), \forall e \in E, d \in \{-,+\}, k \in [K]. \quad (15)$$

Дане обмеження гарантує, що кожна вершина може бути відвідана не більше ніж раз:

$$\sum_{k=0}^{K-1} z_u^k \le 1, \forall u \in V.$$

Часовий інтервал для початку і завершення туру задається таким чином:

$$start_{sl_k} = st_k, \forall k \in [K].$$

$$start_{el_k} \le et_k, \forall k \in [K].$$

Обмеження на значення, які можуть приймати змінні, такі:

$$y^k(e^d), p^k(e^d), z_u^k, r^k(e^d, e'^{d'}) \in \{0,1\},$$

$$\forall e, e' \in E, u \in Vd, d' \in \{-,+\}, k \in [K]$$

$$start_e, start_u \in R, \forall e \in E, u \in V.$$

Щоб гарантувати, що в початкову вершину не входять ребра маршруту, який будується, а з кінцевої не виходять ребра маршруту, вводяться такі обмеження:

$$p^k(e^d) = \begin{cases} 0, якщо\ h(e^d) = sl_k \\ 1, якщо\ h(e^d) \ne sl_k \end{cases}.$$

$$q^k(e^d) = \begin{cases} 0, якщо\ t(e^d) = el_k \\ 1, якщо\ t(e^d) \ne el_k \end{cases}$$

$$z_{sl_k}^k = z_{el_k}^k = 1, \forall k \in [K].$$

$$\forall k \in [K], d \in \{+,-\}, e \in E.$$

Обмеження (14),(15) гарантують, що ребра будуть відвідані в рамках їх часових вікон, а обмеження (16),(17) – що вершини будуть відвідані в рамках їх часових вікон.

$$start_u + T_u \le C_u^k + M(1 - z_u^k), \forall u \in V, k \in [K]. \quad (16)$$

$$O_u^k - M(1 - z_u^k) \le start_u, \forall u \in V, k \in [K]. \quad (17)$$

Список літератури

1. Gavalas D., Mastakas K., Charalampos K., Pantziou G. A survey on algorithmic approaches for solving tourist trip design problems // Journal of Heuristics. – 2014. – №20. – P. 291–328.
2. Kara I., Derya T., Bicakci P. New Formulations for the Orienteering Problem // Procedia Economics and Finance. – 2016. – №39. – P. 849–854.
3. Bianchessi N., Speranza M., Mansini R.A branch-and-cut algorithm for the Team Orienteering Problem // International Transactions in Operational Research. – 2017.
4. Özdemir M. The Team orienteering problem with time windows by using artificial bee colony algorithm // 15th International Symposium on Econometrics, Operations Research and Statistics. – 2014
5. Li J., Li X., Wu Q., Zhu D. Study on the Time-Dependent Orienteering Problem // ICEEE. – 2010.
6. Speranza M., Archetti C. Arc routing problems with profits // Arc routing problems: Problems, Methods, and Applications / M. Speranza, C. Archetti. – Philadelphia: SIAM, 2014. – (MOS/SIAM Series on Optimization). – (20). – P. 281–300.
7. Deitch R., Ladany S. The one-period bus touring problem: Solved by an effective heuristic for theorienteering tour problem and improvement algorithm // Eur. J. Oper. Res. – №127. – P. 69–77.
8. Maervoet J., Brackman P., Verbeeck K., de Causmaecker P., Vanden Berghe G., Tour Suggestion forOutdoor Activities // 12th International Symposium on Web and Wireless GeographicalInformation Systems. – 2013. – №7820. – P. 54–63.
9. Cerna A., Cerny J., Malucelli F., Nonato M., Polena L., Giovannini A. Designing Optimal Routes forCycle-Tourists // Transportation Research Procedia. – 2014. – №3. – P. 856–865.
10. Gavalas D., Konstantopoulos C., Mastakas, K., Pantziou G., Vathis N. Approximation algorithms for thearc orienteering problem // Inf. Process. Lett. – 2015. – №115. – P. 313–315.
11. Garey M. Computers and Intractabilit // M. Garey, D. Johnson. – San Francisco: Freeman, 1979.

Gavalas D., Konstantopoulos C., Mastakas, K., Pantziou G., Vathis N. Efficient Metaheuristics for the Mixed Team Orienteering Problem with Time Windows // MDPI Algorithm

УДК 004.62, 004.93'1

СКРИПНИК А.В.
ХАЛУС О.А.

АНАЛІЗ ПОВЕДІНКИ КОРИСТУВАЧА МОБІЛЬНОГО ПРИСТРОЮ ДЛЯ ПОБУДОВИ ТЕХНОЛОГІЇ СТВОРЕННЯ РЕКОМЕНДАЦІЙ

Лайфлогінг (англ. life-logging, від life і log - "життя" і "бортовий журнал") — автоматичне фіксування повсякденного життя людини на цифровий носій з використанням портативних компактних технічних пристроїв і систем. Проблемами лайфлогінгу переймаються багато сучасних компаній та корпоративних проектів. Проте усі вони вирішують лише певні задачі та сконцентровані на певних корпоративних вимогах. Клієнтські рішення сконцентровані на статичних показниках та представленні даних у вигляді графіків.

Проте технології та напрямок лайфлогінгу дуже перспективний – дозволяє робити більш вірні прогнози для медиків та страхових компаній. Для роботодавця – показники життєвого балансу між роботою та відпочинком. Метою дослідження є розробка рекомендаційної системи на основі контекстних даних користувача. У даній статті розглядається перший етап – збір та статичний аналіз отриманих даних.

Ключові слова: ЕКСПЕРТНА СИСТЕМА, ОБРОБКА КОНТЕКСТНИХ ДАННИХ КОРИСТУВАЧА, АЛГОРИТМ ГЕНЕРУВАННЯ РЕКОМЕНДАЦІЙ, ПРОЦЕС ЗБОРУ ПЕРСОНАЛЬНИХ ДАНИХ КОРИСТУВАЧА, IOS ЗАСТОСУВАННЯ, ВЕБ СЕРВЕР

Lifelogging (Eng. Life-logging, and log on life - "life" and "logbook") - automatic recording of daily life in the digital media using compact portable technical devices and systems. The problems concerned lifelogging many modern companies and corporate projects. But all they solve only some problems and focused on specific corporate requirements. Custom solutions focused on static parameters and view in graphs.

But technology and a very promising direction lifelogging - allows more faithful forecasts for physicians and insurance companies. For the employer - indicators of life balance between work and rest. Purpose and objectives of the study is research is to develop a recommender system based on user context information. This article discusses the first stage - the collection and analysis of the static data.

Keywords: expert systems, contextual DATA PROCESSING USER GENERATION ALGORITHM RECOMMENDATIONS, THE PROCESS OF COLLECTING PERSONAL USER DATA, IOS USE, THE WEB SERVER

1. Вступ

Проблема полягає у великій зашумленості вихідних даних та малому кількості публічних інструментів для її комплексної обробки в рамках задачі аналізу поведінки користувача.

Наявні додатки та системи пропонують лише статичний аналіз даних у вигляді графіків, а усі рекомендації базуються на підрахунку показників користувача і порівняння їх із експертними оцінками.

Програмна технологія покликана вирішити проблему аналізу та надання рекомендацій як окремим користувачам так і компаніям, що аналізують поведінку своїх працівників під час робочого дня, проте спершу потрібно розібратися із якістю даних, що надходять для аналізу та провести оцінку коректності даних.

2. Опис проблеми

Сучасні мобільні пристрої мають достатньо точні сенсори для підрахунку кроків, пересувань, місцезнаходження користувача. Вони надають вже частково оброблені дані розробникам мобільних застосунків, проте алгоритми, що вони використовують – комерційна таємниця і невідомо, наскільки коректні дані ми

отримуємо, невідомо також похибки вимірювання та інші статистичні данні, що допомогли би нам для оцінки коректності отриманих даних.

Дані будуть передаватися за допомогою бездротових мереж Edge, 3G та WiFi [1].

Додатково потрібно враховувати, що сенсори телефону можуть дуже сильно впливати на кількість енергії, що використовує пристрій для роботи. Тому потрібно реалізувати три режими роботи додатку – Low, Middle, Accuracy.

Low – данні збираються лише при значних змінах геопозиції користувача.

Middle – данні користувача збираються кожні 120 секунд.

Accuracy – данні користувача збираються кожні 10 секунд під час пішохідних прогулянок та велосипеді, кожні 30 секунд під час автомобільних поїздках, кожні 120 секунд у стаціонарному положенні.

3. Алгоритм збору даних із мобільних пристроїв, транспортування на сервер, узгодження конфліктів та статистична обробка

КРОК 1. Користувач обирає один із трьох режимів роботи додатку (Low, Middle, Accuracy) для економії енергії телефону під час процесу збору інформації.

КРОК 2. Збираємо дані до мобільного пристрою із різноманітних джерел, до яких є доступ.

КРОК 3. Накопичуємо дані у стек перед відправкою для оптимізації навантаження серверу.

КРОК 4. Робимо первісну обробку даних, прибираємо колізії, для зменшення шумів.

КРОК 5. Нормалізуємо абсолютні значення даних.

КРОК 6. При досягнені ліміту стеку – відправляємо оброблені дані на сервер.

КРОК 7. Під час прийому даних – сервер додатково вирішує merge-conflict проблеми та зберігає дані.

КРОК 8. Під час запиту даних користувача – сервер формує активності користувача, розподіляючи вихідні дані по типам та категоріям (сон, робота, активність, дозвілля та пересування).

КРОК 9. Результатом роботи алгоритму є посортовані дані у JSON форматі.

4. Результати

Отже, зазначений алгоритм збору та обробки було реалізовано у вигляді iOS [2] додатку та .NET серверу. Алгоритм був протестований на iPhone 7, у різних умовах і має наступні результати. У тестуванні приймали участь 4000 чоловік. Мешканці Києва. Чоловіки (70%) та жінки (30%) віком 18-24 – 12%, 25-34 – 74%, 35-54 – 13%. Більше 65% цікавляться бізнесом і технологіями.

4.1. Кар'єра

На рисунку 1 зображено загальна кількість часу, що проводять користувачі на роботі у робочі дні, за вирахуванням часу, що люди проводять за обідом.

Середній час: 5:40

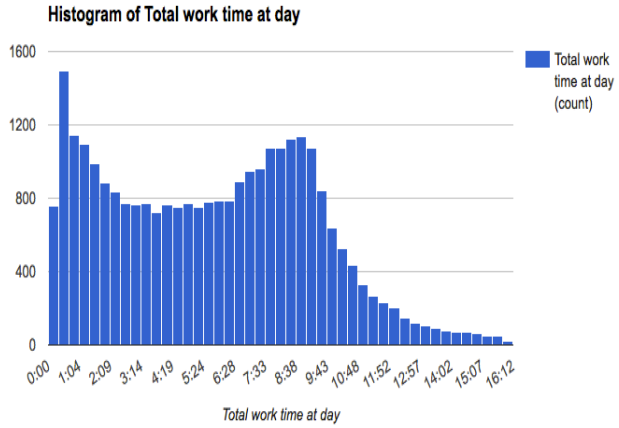


Рис. 1. Розподіл робочого часу упродовж дня

На рисунку 2 зображено кількість часу, що проводять користувачі на роботі у робочі дні, без перерви.

Середній час: 3:12

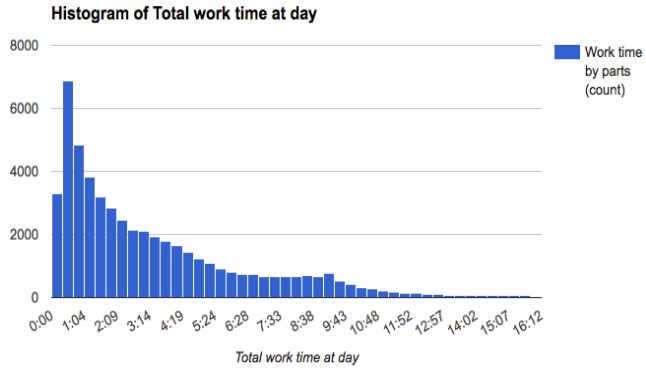


Рис. 2. Розподіл робочого часу упродовж дня по частинам

На рисунку 3 зображено години, коли користувачі приходять на роботу.

Середній час: 10:59

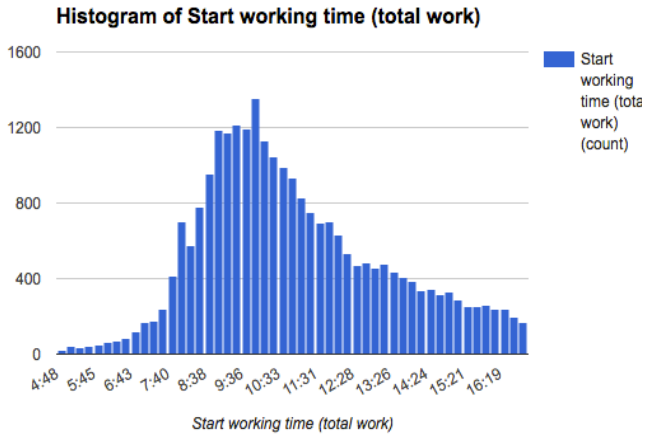


Рис. 3. Час початку робочих годин упродовж дня

На рисунку 4 зображено години, коли користувачі покидають роботу.

Середній час: 18:17

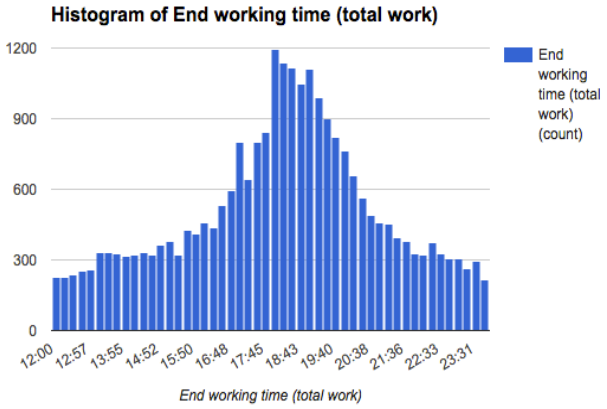


Рис. 4. Час кінця робочих годин упродовж дня

4.2. Сон

На рисунку 5 зображено час, що користувачі сплять.

Середній час: 7:37

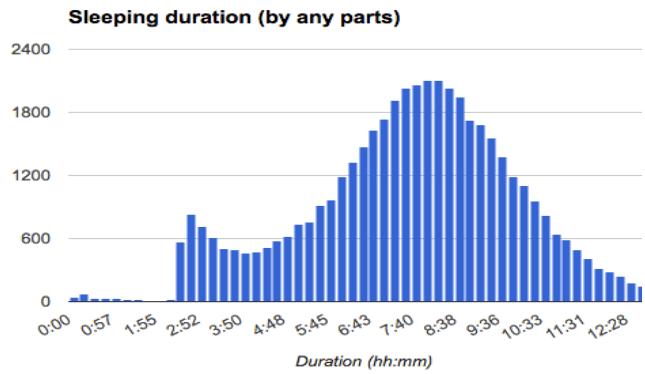


Рис. 5. Розподіл тривалості часу сну упродовж ночі

4.3. Активність

На рисунку 6 зображено середній час активності у день.

Середній час: 1:53

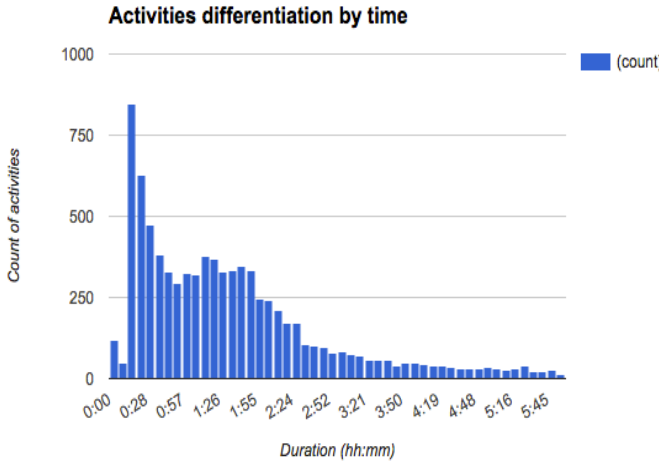


Рис. 6. Розподіл тривалості часу сну окремими інтервалами

На рисунку 7 зображено середню кількість активності кожен день.

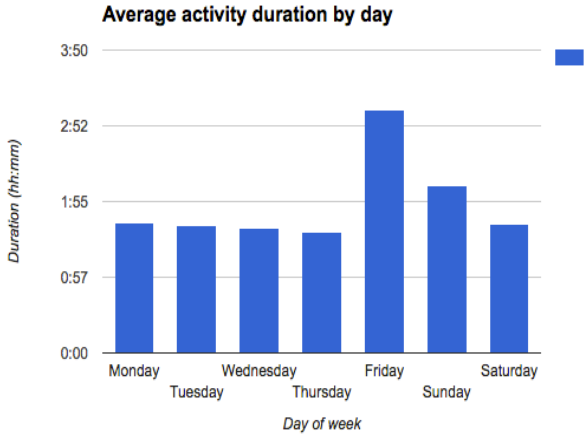


Рис. 7. Розподіл середнього часу активності по дням

5. Висновки

Таким чином, реалізований алгоритм працює у вигляді застосування для мобільної операційної системи iOS та серверу на .NET [3]. На графіках видно доволі велику кількість даних, правдивість яких потрібно перевірити. Записи про кількість часу роботи менше 4 годин в день, тривалість сну більше 10 годин, закінчення роботи раніше 14:00 годин дня.

У подальших планах – продовжувати розробку та покращення роботи алгоритму, зокрема розробити інтерактивну систему підтвердження

коректності даних із боку користувача. при зборі даних на пристроях. Також планується понизити % похибки

Список літератури

1. Sustainable Energy-Efficient Wireless Applications Using Light Mohsen Kavehrad, Pennsylvania State University, IEEE Communications Magazine, December 2010.

2. Apple iOS Documentation. [Electronic resourse] – Electronic data. – Apple.com April 2017 – Mode of access: World Wide Web: https://developer.apple.com/library/content/documentation/UserExperience/Conceptual/LocationAwarenessPG/CoreLocation/CoreLocation.html (viewed on April 5, 2017). - Apple, Getting the User's Location

3. Microsoft Developer Documentation. [Electronic resourse] – Electronic data. – Microsoft April 2017 – Mode of access: World Wide Web: https://docs.microsoft.com/en-us/dotnet/ (viewed on April 10, 2017). - .NET Documentation | Microsoft Docs.

УДК 519.684.6

ХРАМЧЕНКО М.С.
МУХА І.П.

МОБІЛЬНИЙ ДОДАТОК ДЛЯ ВЗАЄМНОГО ОРІЄНТУВАННЯ КОРИСТУВАЧІВ НА МІСЦЕВОСТІ

Розглянутий підхід до побудови мобільного навігаційного додатку, який забезпечує взаємну навігацію і комунікацію між обмеженою групою користувачів. Проведено аналіз існуючих мобільних додатків для вирішення даного класу задач, виділені їх недоліки, запропоновані конкретні рішення по розширенню функціональних можливостей таких додатків.

The approach to building a mobile navigation application that provides navigation and mutual communication between a limited group of users. The analysis of existing mobile applications for solving this class of problems highlighted their shortcomings, offered specific solutions to expand the functionality of these applications.

Постановка проблеми

Рівень розвитку сучасних технологій дозволяє обладнувати різноманітні мобільні пристрої (мобільні телефони, смартфони, планшети і т. ін.) засобами доступу до глобальної системи навігації GPS та до бездротового мобільного Інтернету. Це дає змогу створювати і використовувати мобільні додатки для вирішення задач навігації на місцевості, що значно прискорює процес пошуку об’єктів, як наприклад, пошук будинку за адресою у чужому місті.

Важливим аспектом створення таких додатків є розробка навігаційних додатків для групового використання, основними завданнями яких є взаємна навігація і комунікація.

Існуючі рішення

Натепер існує певна кількість навігаційних додатків, які забезпечують взаємну навігацію і комунікацію між обмеженою групою користувачів.

1. *GPS TrackerPro* (*FindMyFriends*!) - мобільний додаток, що дозволяє швидко знайти членів сім'ї та друзів. Дозволяє розшарювати локацію усім членам групи, однак у ньому відсутня можливість обмеження групи людей, що можуть відстежувати місцезнаходження та маршрут в даний момент часу будь-кого з членів групи. Даний недолік може бути зручним для батьків, які хочуть відстежувати маршрути переміщення дітей. Відсутня можливість побудови спільних маршрутів.

2. *GPS-Trace* – додаток, що дозволяє визначити місцезнаходження об’єкта в режимі реального часу, зберігає історію маршрутів і повідомляє користувачів про збої в роботі GPS або мережі Інтернет. Даний додаток дозволяє створювати групи користувачів, але не більше ніж на 5 осіб. Відсутня можливість створення статичних точок, та побудови спільних маршрутів до них.

3. *IM MapNavigator* - додаток, що дозволяє знайти друзів або членів сім'ї за допомогою мобільного телефону в режимі реального часу. Присутня функція миттєвих повідомлень з медіаконтентом. Є можливість відображення на карті незнайомих людей, що знаходяться поруч. Основний недолік – відсутність можливості створення спільних маршрутів між членами групи.

Таким чином, усі, вище перераховані програмні засоби, мають ряд спільних недоліків, а саме:

- відсутність можливості додавання статичних точок на карту;
- відсутність можливості побудови спільних маршрутів для членів групи;
- відсутність інтегрованої системи обміну повідомленнями.

Розробка навігаційного додатка, який би підтримував дані можливості, значно розширило б функціональність такого програмного забезпечення і підвищило б його популярність серед користувачів.

Запропоноване рішення

З урахуванням стану розвитку сучасних технологій проектування програмних засобів, для вирішення поставленої задачі обрано клієнт-серверну архітектуру програмного додатку (рис. 1).

Клієнт через Інтернет-з'єднання підключається до сервера і виконує необхідні і допустимі для нього дії.

Відповідно, даний додаток вимагає наявності GPS-датчика Інтернет-з'єднання.

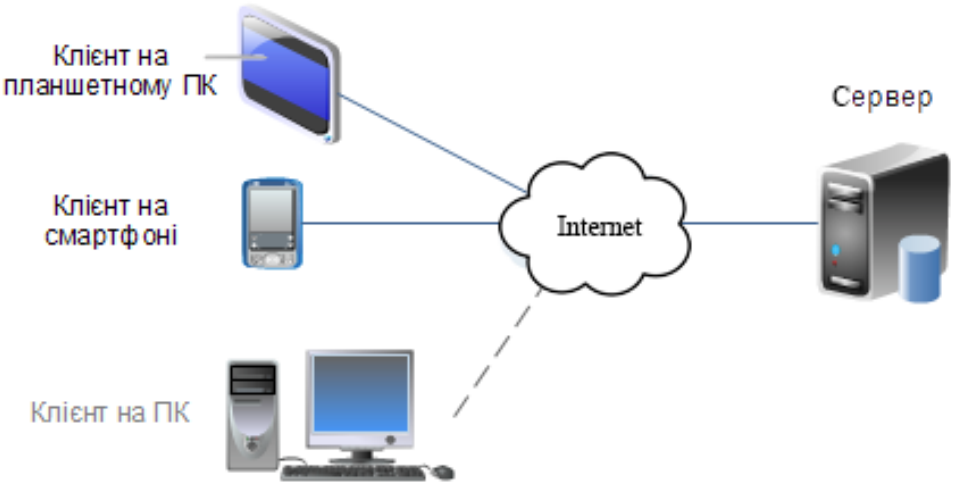


Рис. 1 Принципова схема клієнт-серверної моделі

Для побудови додатку використовується набір програмного забезпечення, який орієнтований на розробку мобільних клієнт-серверних додатків та містить наступні компоненти: MongoDB в якості бази даних, Express.js в якості сервера, iOS SDK та AndroidSDK у якості основи для клієнтської частини та Node.js – JavaScript платформа для серверної розробки. Модель даного додатку наведена на рис. 2.

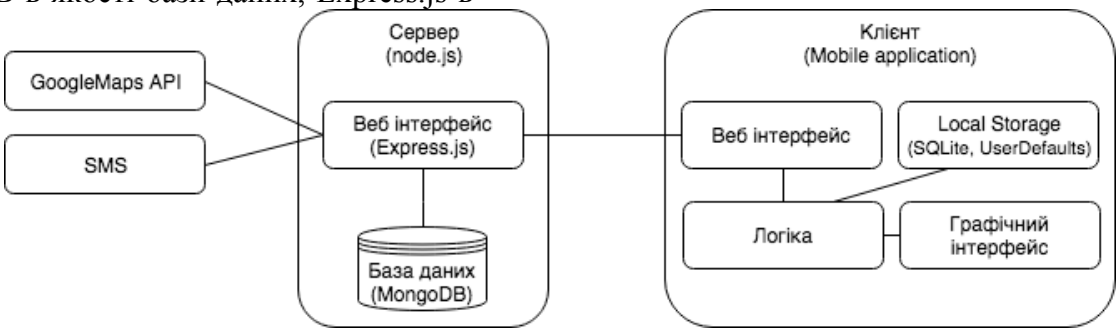


Рис. 2. Модель додатку

Серверна частину системи зберігає статичні і динамічні дані і забезпечує комунікацію між групами користувачів. Клієнтська частина системи надає інтерфейси для навігації та комунікації.

Узагальнений алгоритм роботи додатку: програма визначає координати користувачів, відправляє їх на сервер з попередньо визначеним ім'ям групи, логіном (номер телефону) і паролем, а потім за запитом роздає цю інформацію користувачам цієї групи.

Сценарій роботи додатку наступний:

1. Реєстрація користувача за допомогою мобільного телефону та підтвердження реєстрації шляхом введення секретного

коду, який відправляється СМС-повідомленням.

2. Вибір аватару у вікні налаштувань та підключення до групи.

3. Якщо користувач ввів правильну назву групи і пароль до неї, то він отримує доступ до спільної навігації і може відстежувати переміщення інших членів групи та будувати спільні маршрути.

4. За бажанням, користувач може обмінятися коротким текстовим повідомленням з будь-яким іншим членом групи. Для цього йому потрібно відкрити вікно повідомлень.

Екранні форми для виконання зазначених дій представлені на рис. 3.

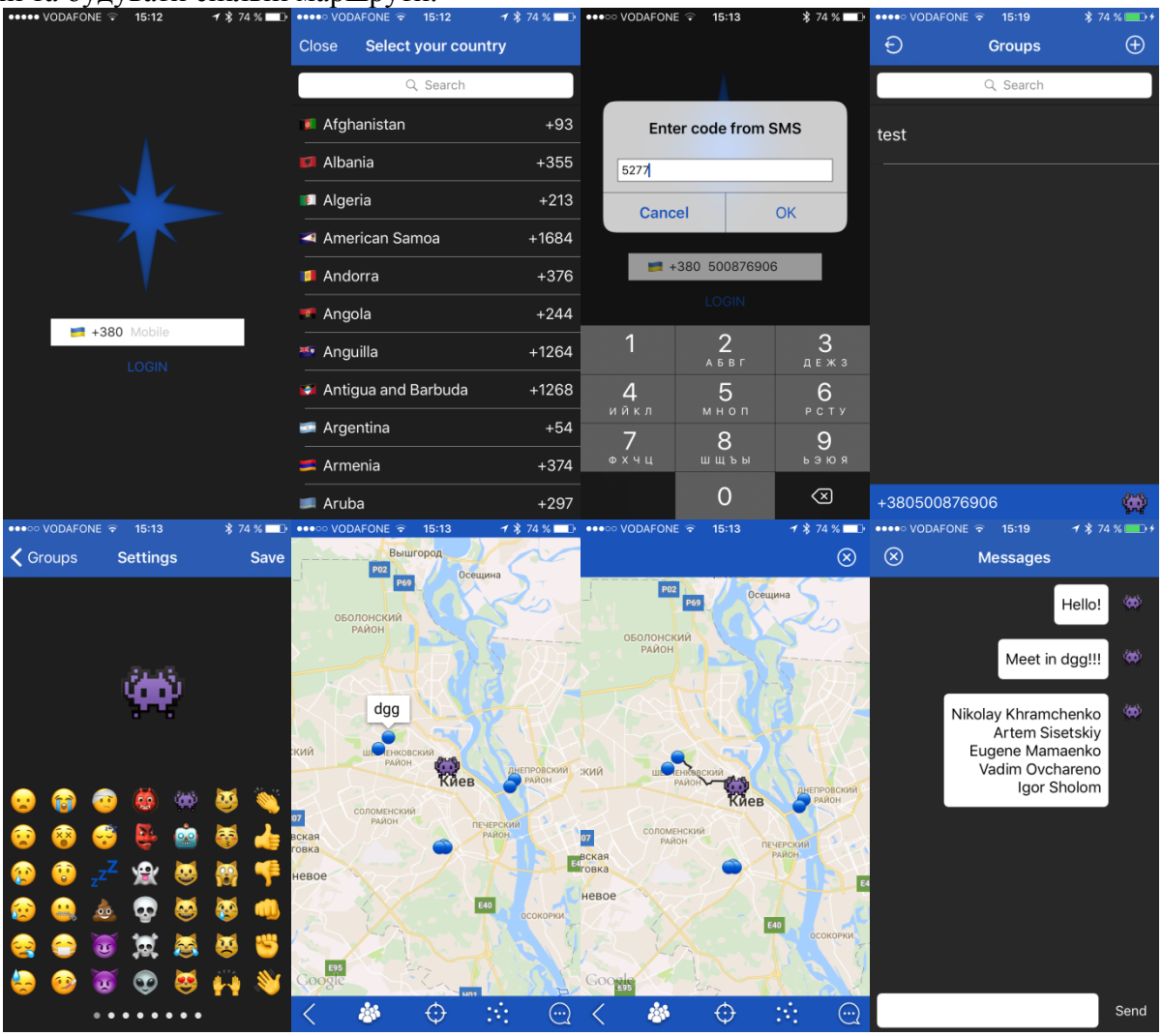


Рис. 3. Графічний інтерфейс додатку

Основні функціональні можливості програми:

- додавання / оновлення свого запису в базі даних;
- перегляд даних членів своєї групи;
- додавання на карту статичних точок;
- побудова спільних маршрутів для членів групи (рис. 4);
- інтегрована система обміну повідомленнями.

Клієнтська частина розробленої системи може працювати у всіх версіях операційної системи iOS, починаючи з iOS 9.0, та Android, починаючи з версії 5.0.

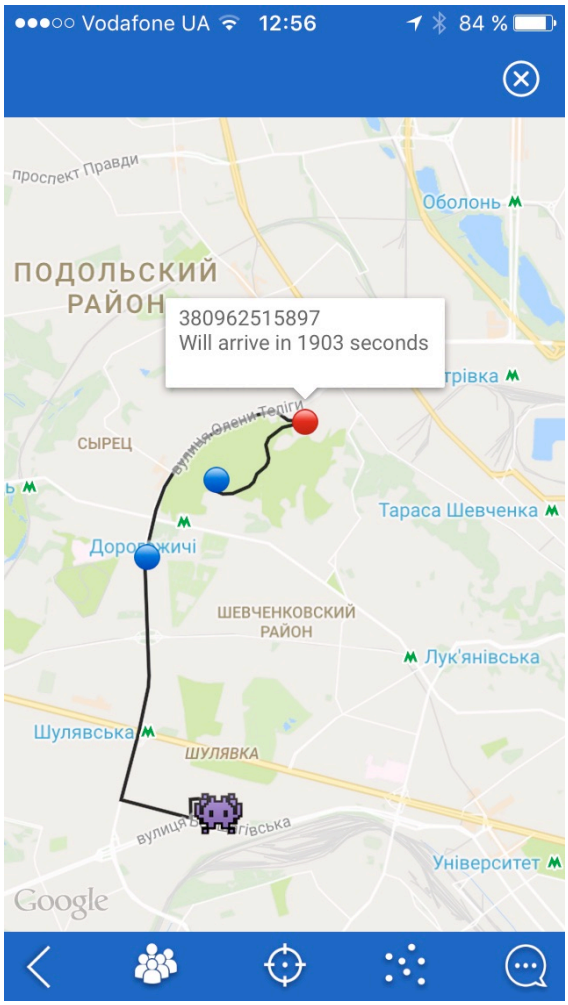


Рис. 4. Побудова спільних маршрутів для членів групи

Висновки

Запропоновано підхід до організації взаємної навігації за рахунок розробки клієнт-серверного мобільного додатку для взаємного орієнтування користувачів на місцевості з можливістю побудови спільних маршрутів та обміну миттєвими повідомленнями між членами групи.

Програма досить проста у використанні та може використовуватись для вирішення різноманітних задач, пов'язаних із взаємною орієнтацією користувачів.

Список літератури

1. GPSPhoneTracker [Електронний ресурс] Режим доступу: https://play.google.com/store/apps/details?id=com.fsp.android.c&hl=en
2. GPS-Trace [Електронний ресурс] Режим доступу: https://play.google.com/store/apps/details?id=com.gurtam.gps_trace_orange&hl=en
3. PhoneTracker-IM MapNavigator [Електронний ресурс] Режим доступу: https://play.google.com/store/apps/details?id=nfadev.sn.immnavigator&hl=en
4. Компоненты сетевого приложения. Клиент-серверное взаимодействие и роли серверов. [Електронний ресурс] Режим доступу: http://www.4stud.info/networking/lecture5.html
5. Архитектура мобильного клиент-серверного приложения. [Електронний ресурс] Режим доступу: https://habrahabr.ru/post/246877

УДК 683.519

ЛІЩУК К.І,
ШЕВЧЕНКО Є.Т.

БАЛАНСУВАННЯ НАВАНТАЖЕННЯ МІЖ КЛАСТЕРОМ СЕРВЕРІВ З УРАХУВАННЯМ ТИПУ ЗАДАЧ

Наведено опис оптимізованого алгоритму балансування навантаження між сереверами з урахуванням типу задачі для визначення ноди кластеру яка являється найменше завантаженою в момент винекнення такої задачі.

The description of the algorithm optimized for load balancing between servers considering the type of task to determine which cluster node is the least busy at the moment task solving.

1. Вступ

З кожним роком кількість нових інтернет-ресурсів, які виконують різноманітні задачі зростає експоненціально відносно кількості нових користувачів в мережі Інтернет. Лише за 2016 рік було зафіксовано приріст користувачів мережі близько семи відсотків. Але зростає не тільки кількість нових ресурсів а й навантаження на вже існуючі. Великі за обсягом трафіку компанії для підвищення надійності та достсупності своїх ресурсів використовують так званні методи горизонтального маштабування, а саме збільшення кількості серверів для обробки даних.

Для розподілення навантаження між великими кластерами серверів використовується проміжне програмне забезпечення, так звані балансувальники (Load Balancers). В переважній більшості випадків такі балансувальники працюють методом простого й почергового перебору всіх доступних серверів, тобто при надходженні нової задачі балансувальник нічого не врахову окрім черговості серверів.

Опис проблеми. Уявімо ситуацію коли у розпорядженні простого балансувальника є 5 серверів які займаються обробкою ресурсоємких задач, задачі по рівню їх складності можуть займати від кількох секунд до кількох годин роботи серверу в режимі максимального завантаження. Метод простого перебору серверів буде не оптимальним в разі якщо перша задача була максимально ресурсоємна а всі наступні були прості та швидкі по часу виконання. В такому випадку шоста задача буде довгий час очікувати вирішення першої, хоча задачі з другої по п’яту до цього моменту були вже вирішені а сервери готові отримувати наступні задачі. Одним з найпопулярніших рішень цієї проблеми є введення максимального часу відповіді серверу про прийняття задачі на вирішення (timeout). Тобто це час після якого балансувальник спробує передати задачу наступному серверу. Такий підхід є тільки частковим вирішенням проблеми тому що:

1. Вводится певний timeout, під час якого задача буде простоювати, а отже час отримання відповіді збільшиться на невідому величину.
2. Додаткове навантаження на оперативну пам’ять балансувальника.

Пропоноване рішення. Для більш ефективного розподілу навантаження між серверами пропонується розробити таке програмне забезпечення балансувальника яке не буде зберігати в пам’яті нічого що стосується самої задачі а лише буде підтримувати певний протокол правил за якими дана задача буде розподілена по кластеру. Даний підхід можливий тільки для

балансування на прикладному рівні, в свою чергу це означає що програмне забезпечення буде окремо налаштовуватися для кожного набору задач.

2. Короткий опис існуючих рішень

На даний час веб-сервісам знаходять все більш широке застосування. Вони використовуються в різних випадках і ситуаціях. Але швидке збільшення кількості веб-сервісів і користувачів цих сервісів потребує підвищення продуктивності веб-серверів, для зменшення часу відгуку на запити, які, в свою чергу можуть з'явитися в будь-який момент часу. При цьому необхідно забезпечити ще й надійність таких веб-серверів.

Для реалізації високопродуктивних і надійних веб-серверів використовують розподілені веб-сервери. Розподілені веб-сервери представляють собою набір (N-ну кількість серверів) веб-серверів. Це продубльовані ресурси для одночасного надання послуг багатьом користувачам. Вхідні запити можуть бути розподілені між серверами згідно певних стратегій розподілення завантаження, і тому ці запити можуть бути опрацьовані в певних часових межах (час відгуку). Розподілені веб-сервери можуть бути організовані різними способами:

• вони можуть бути інтегровані в кластер веб-серверів, з'єднаних через локальну обчислювальну мережу, щоб працювати як один потужний сервер;

• вони можуть використовуватись в різних географічних місцях через глобальну обчислювальну мережу;

Розподілені веб-сервери можуть легко розширюватися, та мають високу степінь масштабованості. Кількість може бути збільшено простим додаванням нового сервера в локальну мережу .

Для забезпечення надійної масштабованості високопродуктивних веб-серверів потрібно проведення балансування завантаження усіх веб-серверів. Вхідні запити від користувачів повинні бути розподілені відповідно до стратегії навантаження між серверами, щоб ко ристувач отримував відповідь на запит в певних часових межах. Як і у випадках з розподіленими обчисленнями та розподіленим моделюванням, роботу з перевантажених серверів необхідно перемістити на не завантажені, що сприяє підвищенню пропускної здатності системи. В іншому випадку може бути ситуація, коли запит користувача буде знаходитись в черзі безкінечно довго. В таких випадках сервер може відхилити запит користувача.

У термінології комп'ютерних мереж, балансування навантаження, або вирівнювання – метод розподілу завдань між декількома мережевими пристроями (наприклад, серверами) з метою оптимізації використання ресурсів, скорочення часу обслуговування запитів, горизонтального масштабування кластера (динамічне додавання / видалення пристроїв), а також забезпечення відмовостійкості (резервування).

У комп'ютерах, балансування навантаження розподіляє навантаження між декількома обчислювальними ресурсами, такими як комп'ютери, комп'ютерні кластери, мережі, центральні процесори або диски. Мета балансування навантаження - оптимізація використання ресурсів, максимізація пропускної здатності, зменшення часу відгуку і запобігання перевантаження будь-якого одного ресурсу. Використання декількох компонентів балансування навантаження замість одного компонента може підвищити надійність і доступність за рахунок резервування. Балансування навантаження передбачає зазвичай наявність спеціального програмного забезпечення або апаратних засобів, таких як багаторівневий комутатор або система доменних імен, як серверний процес.

3. Загальна математична модель

Формальну постановку задачі балансування навантаження проведемо з використанням моделі замкненої мережі масового обслуговування з центральним

обслуговуючим сервером. Вона дає змогу відобразити архітектуру кластера (рис.1).

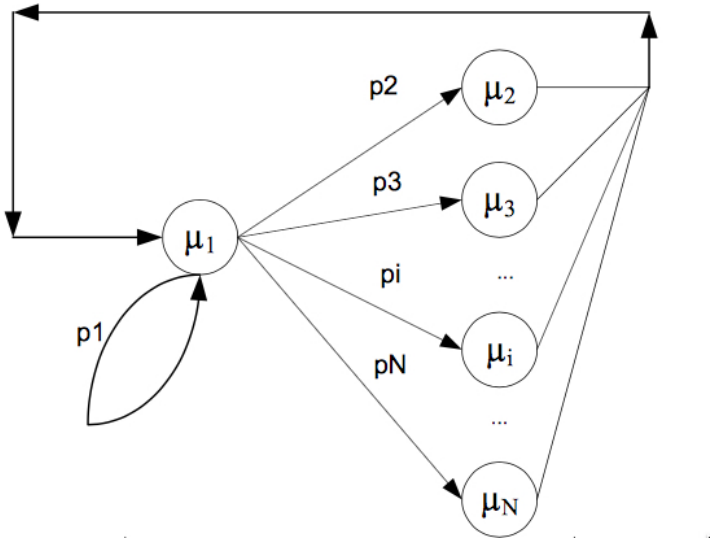


Рис. 1 Модель з центральним обслуговуючим сервером

Модель дозволяє описати роботу обчислювальної системи (з фіксованим числом частин), в яку допускається точно К завдань. Ці завдання циркулюють в системі нескінченно довго, колективно використовуючи N її ресурсів. Центральний обслуговуючий сервер (вузол 1) являє собою кореневої вузол кластера, а решта N-1 вузлів - периферійні вузли кластера. У такій системі завдання циркулюють між вузлами кластера, вимагаючи звернення спочатку до кореневого вузла, а потім до деякого периферійному вузлу кластера, після цього знову вимагаючи обслуговування в центральному вузлі, а потім знову звернення до якогось периферійного вузла кластера і т. д. Отже, весь час завдання повертаються в кореневий вузол кластера. Перехідні ймовірності – це ймовірності переходу завдання в вузол J з вузла I; в даній моделі з центральним обслуговуючим приладом маємо

$$R_{ij} = \begin{cases} Pj, \ j = 1, 1 \le j \le N \\ 1, \ 2 \le i \le N, j = 1 \\ 0, \text{в решті випадків} \end{cases}$$

де, $\sum_{j=1}^{N} Pj = 1$.

У реальній мультипрограмій ситуації більшість J = 1 завдань в кінці кінців залишають систему, і в моменти їхнього виходу з системи надходять нові завдання. Це враховано в моделі шляхом дозволу завданням прямого повернення в кореневий вузол (з ймовірністю p1), що означає відхід старого завдання і надходження замість нього нового завдання. Таким чином, число завдань в системі залишається постійним і рівним К. У моделі з центральним обслуговуючим сервером в кожному вузлі знаходиться один обслуговуючий сервер ($m_i = 1$), а час обслуговування в I-му вузлі розподілено по показовому закону з параметром μ_i. Нехай матриця R є матрицею перехідних ймовірностей P (R_{ij}) між вузлами мережі. Тоді випадок з центральним обслуговуючим сервером можна описати наступною простою матрицею:

$$R = \begin{bmatrix} P1 & \cdots & Pn \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 0 \end{bmatrix}$$

Завдання зводиться до вирішення рівняння $\lambda = \lambda R$; для розглянутої матриці рішення має вигляд:

$$\mu_i x_i = \begin{cases} \mu 1, i = 1 \\ \mu 1 pi, i = 2,3, \dots, N \end{cases}$$

За наведеними формулами отримуємо $A1\mu 1p1 = Ai\mu i$ $(i > 2)$, тобто інтенсивність, з якою завдання надходять в I-й вузол, дорівнює інтенсивності, з якою вони залишають цей

вузол. Ця рівність показує, що міра, визначаюча наскільки вузьке місце створюється в I-му вузлі, пропорційна швидкості зміни продуктивності в залежності від зростання інтенсивності обслуговування в цьому вузлі. При цьому продуктивність визначається як середнє число завдань, що обслуговуються за одиницю часу. Вона дорівнює A1μ1p1. Отже, збалансована система - це така система, для якої:

$$\frac{d}{d\mu_1} A_1\mu_1 p_1 = \frac{d}{d\mu_j} A_1\mu_1 p_1, 1 \le i,j \le N$$

Тепер, грунтуючись на результатах наведеного аналізу, розглянемо задачу балансування навантаження в серверній фермі.

Типовий запит складається з назви ресурсу (URL, ім'я файлу, номер повідомлення). При цьому до того, як клієнт пошле ім'я ресурсу веб-сервера, між ними може відбутися обмін додатковими повідомленнями, знову ж таки, коректними з точки зору протоколу (наприклад, з метою авторизації користувача). Всю сукупність повідомлень (включаючи запит і відповідь) між встановленням з'єднання між клієнтом і веб-сервером і його розривом, будемо називати сесією.

У разі асиметричної ферми, кожен реальний сервер веб-ферми не зобов'язаний вміти відповідати на будь-який клієнтський запит. Але для кожного клієнтського запиту в веб-фермі повинен бути присутнім реальний сервер, який здатний обробити даний запит (наприклад, всі URL типу GIF можуть розташовуватися на одному реальному сервері, а URL типу HTML - на іншому). Таким чином, сервіс балансування навантаження повинен бути здатний на підставі клієнтського запиту вибрати реальний сервер, який здатний обробити цей запит і перенаправити запит йому.

4. Опис загального рівня дисбалансу системи і кожного сервера

Необхідно ввести інтегровані значення загального рівня дисбалансу системи, а також середній рівень дисбалансу кожного сервера. Одна з інтегрованих метрик балансу навантаження описується наступним чином:

$$V = \frac{1}{(1-CPU)(1-RAM)(1-net)}$$

де CPU, RAM, net - середнє завантаження процесора, пам'яті і пропускної здатності мережі, відповідно, протягом кожного спостережуваного періоду. Велике значення V означає високу ступінь комплексного використання. Тому алгоритми міграції можуть бути засновані на даному вимірі. Це насправді є стратегією мінімізації комплексного використання ресурсів шляхом перетворення тривимірної інформації (3D) ресурсів в одновимірну величину (1D). Це перетворення може призвести до багатовимірної втрати інформації.

Також можуть бути розроблені інші показники для різних стратегій планування.

Розглянуто наступні параметри:

1. Середнє завантаження CPU i-го сервера визначається як середнє завантаження процесора протягом спостережуваного періоду. Наприклад, якщо період спостережень становить 1 хвилина, а завантаження процесора записується через кожні 10 секунд, тобто CPU - це середнє значення з шести записаних значень i-го сервера.

2. Середній коефіцієнт використання всіх процесорів в системі. Нехай CPUi - загальне число ЦПУ i-го сервера,

$$CPU_u^A = \frac{\sum_i^N CPU_i^u CPU_i^n}{\sum_i^N CPU_i^n}$$

де N - загальне число фізичних серверів в системі. Аналогічним чином, середній коефіцієнт використання пам'яті, пропускної здатності мережі i-го сервера.

3. Комплексне значення дисбалансу навантаження i-го сервера. Використовуючи дисперсію, інтегроване значення дисбалансу навантаження i-го сервера визначається як:

$SIL_i = a(D_i - CPU_u^A)^2 + b(D_i - RAM_u^A)^2 + c(D_i - NET_u^A)^2$,

де $D_i = aCPU_i^u + bRAM_i^u + cNET_i^u$

Значення дисбалансу застосовується для позначення рівня дисбалансу навантаження шляхом порівняння коефіцієнтів використання процесора, пам'яті і пропускної здатності мережі.

4. Значення дисбалансу всіх процесорів, пам'яті і пропускної здатності мережі. Використовуючи дисперсію, значення дисбалансу всіх процесорів в центрі обробки даних визначається як:

$$ISL_{tot} = \sum_{i}^{N} ISL_i$$

Список літератури

1. Джон Оллспоую. Искусство планирования мощностей – 126 стр.
2. Олега Бунина. Разработка высоконагруженных систем. По материалам конференции HighLoad++ 2010-2011, 416 стр

УДК 506+510

ХАЛИМОН А.Ю.
БАКЛАН І.В.

ПРОЦЕДУРА ЛІНГВІСТИЧНОГО МОДЕЛЮВАННЯ ЧАСОВИХ РЯДІВ

Дана робота є продовженням праці над створенням універсального методу аналізу часових рядів з використанням лінгвістичного моделювання. Розглянуто формальну модель такого методу, проаналізовано його можливі застосування.

This thesis is a continuation of work on creating universal method for time series analysis with help of the lingustic modeling. Formal model for such method is described, potential applications of this method are analyzed.

АКТУАЛЬНІСТЬ

Сучасний стан науково-технічного прогресу характеризується динамікою економічних, соціальних, науково-технічних та інших процесів. Іншим серйозним викликом є вибухоподібне зростання кількості даних, що накопичуються людством про ті самі ж процеси. Це проявляється в надзвичайному поширенні різноманітної техніки, яка потребує для свого створення і оптимізації достовірні моделі, з іншої ж сторони - з'являється можливість збирати велику кількість даних про діяльність машин, людей, природи (різноманітного роду датчики - GPS, гіроскопи, акселерометри, термометри, весь Internet of Things (IoT)), і засобів до обробки цих даних (GPU, нейронні мережі, бази даних для часових рядів (InfluxDB), Hadoop та ін.).

В зв'язку з цим постає необхідність дослідження даних процесів з використанням великої кількості вже накопичених і майбутніх даних. Природа самих процесів дуже часто невідома, і для роботи з ними необхідно їхнє моделювання. Більшість таких процесів можна описати з допомогою часових рядів, які переважно моделюються з допомогою методів регресійного аналізу. Незважаючи на простоту даних рішень та їхню ефективність, багато складних процесів не можна дослідити точно з їх допомогою.

Іншим, ще більш складним завданням є дослідження саме внутрішньої структури процесу, його фаз. Лінгвістичне моделювання, на відміну від апроксимації процесу вже відомими функціями, не надає моделі процесу форми вже відомої функціональної залежності, але гнучко описує внутрішню структуру процесу, даючи змогу не тільки проводити екстраполяцію ряду, а і дати досліднику розуміння природи процесу.

На даний момент, моделі, схожі з даною, було успішно застосовано до задач розпізнавання природної мови, однак лінгвістичне моделювання не обмежується цією областю, і існують підстави вважати, що коло застосування таких Ціллю моделювання є аналіз та передбачення поведінки модельованого об'єкта. Основна проблема моделювання - баланс між точністю моделі та її коректністю. Ускладнюючи модель, ми зменшуємо її загальність - можливість використання для інших часових рядів, а також можемо зробити гіпотези, які не є достовірними. Характерною відмінністю лінгвістичне моделювання є можливість глибокого аналізу причинно-наслідкових зв'язки, що існують в об'єкті, поєднуючи рух і час.

ПРИЧИНИ ТА ІСТОРІЯ ВИНИКНЕННЯ

Як вже було зазначено вище, причиною інтересу до моделювання часових рядів є велика кількість даних, представлених часовими та необхідність їхнього аналізу.

Лінгвістичне моделювання є розвитком ідеї структурного підходу до розпізнавання

образів, запропонованого K.S. Fu в [1]. Він говорив про необхідність гібридних методів розпізнавання образів, які б поєднували статистичний(що досліджує об'єкт як цілий, і сфокусований на виокремленні його характеристик) та структурний підхід (коли об'єкт розпізнається за ознаками своєї внутрішньої структури). Ключем до відновлення внутрішньої структури є інтервальна математика. Ідея інтервальної математики в можливості дискретизувати структуру часового ряду за значеннями, і виділити стійкі інтервали значень, які потім можна використати в процесі відновлення граматики - лінгвістичної моделі ряду. Маючи таку граматику, можна робити прогнози часового ряду.

Дана стаття є продовженням робіт [2], [3], [4]. Ідея інтервального підходу вперше описана в [5], сучасна теорія інтервального аналізу викладена в [6], одне з типових застосувань такого підходу описано в [7]. Існуючі статистичні методи прогнозування часових рядів розглянуті в [8]. Це авторегресія, ковзне середнє, прогнозування з трендом, дрейфом та інше.

Іншими ідеями, на яких базується метод, є приховані марківські мережі, використання яких для аналізу і прогнозування часових рядів розглянуто в [9] , та метод правдоподібних траєкторій. [10]

СТАДІЇ ПРОЦЕДУРИ ЛІНГВІСТИЧНОГО МОДЕЛЮВАННЯ

Збір даних, що описують схожі процеси. Маючи дані з декількох часових рядів, можна побудувати більш достовірну граматику.

Валідація даних та їх очистка від недостовірних або неефективних з точки зору моделювання (шум, впливи інших, домінуючих факторів). В цілісному процесі аналізу даних на цей етап може припадати до 80% від всіх зусиль, витрачених на моделювання.

Розбивка часового ряду на інтервали – розбивка навчальних даних на інтервали і присвоєння їм символів. Схема присвоєння довільна, єдиною вимогою є присвоєння різним (за характеристиками) інтервалам різних символів. Цей етап ми розглянемо детальніше в наступному розділі.

Розпізнавання синтаксису – отриману на попередньому етапі послідовність аналізують на наявність граматичних конструкцій. На виході отримуємо список граматичних конструкцій з ймовірностями їх наявності в процесі, а також матрицю ймовірностей переходу з символу в символ. Цей етап тісно перекликається з моделюванням (прихованих) марківських процесів, а також з методом правдоподібних траєкторій. Для одного і того ж ряду можна побудувати декілька граматик, які будуть перевірятися на наступному етапі. Процедура детально описана в [3].

Верифікація моделі – перевірка моделі на адекватність і корисність. Адекватність можна перевірити за допомогою прогону моделювання на тренувальній вибірці і застосування до тестової вибірки. Результати застосування моделі до тестової вибірки оцінюються за: збереженням алфавіту, збереженням граматики, збереженням параметрів розподілу букв та слів. Корисність моделі вимірюється евристиками. Такими можуть бути штрафи за

- короткі слова (граматика занадто проста і не надає корисної інформації про процес)
- довгі слова (модель занадто складна, і не є адекватною)
- кількість слів, що мають суттєву спільну частину (це вказуватиме на неправильну розбивку на інтервали - спільну частину можна було замінити однією буквою).

Отримавши сумарну оцінку по кожній моделі, вибирається найкраща, яка і використовується для моделювання цільового процесу.

Моделювання цільового процесу – використовуючи побудовану модель, будується опис синтаксичних конструкцій з вірогідностями їх наставання. Для кожної синтаксичної конструкції надається можливе трактування в термінах предметної області. Ця дана множина і використовується для аналізу часового ряду і прийняття рішень, для допомоги прийняття яких проводилося моделювання.

Розглянемо формально один з можливих варіантів побудови лінгвістичної моделі . Маючи часовий ряд $\{y(i)\}$, де, $y(i), i = \overline{1,N}$ - це деякі значення, які отримано в ході спостереження з якимось кроком, $\Delta t_i = const, i = \overline{1,N}$, що містить дані спостережень в часі , рахуємо:

- Різниці сусідніх значень ряду $\Delta y(i) = y(i) - y(i+1), \overline{1,N}$ між сусідніми членами ряду
- Відсортовуємо окремо додатні та від'ємні значення $\Delta y(i)$ за спаданням (чи за зростанням). Отримаємо дві послідовності $a(k)$ і $b(l)$, $K + L = N - 1$.
- Кожному члену з послідовностей $a(k)$ і $b(l)$ ставимо у відповідність символи абетки $a_i, b_j, i = \overline{1,K}, j = \overline{1,L}$ відповідно.
- Перепишемо послідовність $\Delta y(i)$ символами a_i і b_j, також будемо ставити між парами сусідніх символів a_q, b_p символ c, а між парами сусідніх символів a_m, b_n символ d, які будуть означати точки локальних екстремумів в послідовності $y(i)$, локальні максимуми і локальні мінімуми відповідно. Отримаємо послідовність e_i .
- В послідовності e_i проаналізуємо частоту існування пар символів $(e_i e_{i+1}), i = \overline{1,N-1}$ і побудуємо таблицю ймовірностей виникнення символу $e_{i+1} P_{j+1}(e_{i+1} \mid e_{i-k} ... e_i)$. Тобто обчислюємо ймовірність появи символу e_{i+1} за умови що попередні символи $e_{i-k} ... e_i$.
- За допомогою обчислених ймовірностей можемо зробити імовірнісний прогноз виникнення символу e_{i+1} за умови, що відомі ланцюжки попередніх символів.

Особливий інтерес викликає розрахунок рівноймовірнісних інтервалів.

Нехай часовий ряд має N елементів. Стоїть задача вибору оптимального розміру інтервалу, який фактично задається кількістю елементів m, що до нього потрапляють. Ймовірність потрапляння елементів часового ряду до інтервалу буде дорівнювати $\frac{m}{N}$. Кількість інтервалів буде дорівнювати $K = \frac{N}{m} + 2$. Легко перевірити, що $\sum_{i=1}^{K} P_i = 1$.

РОЗБИВКА ЧАСОВОГО РЯДУ НА ІНТЕРВАЛИ

Одною з проблем побудови хорошої лінгвістичної моделі є розбивка часового ряду на інтервали. Існує два можливих підходи, що виходять з різних припущень дослідника про предметну область:

Відомою величиною є границі інтервалів, невідомою – символ, що відповідає кожному інтервалу. В такому разі, для побудови алфавіту необхідне функція, що задає відображення множини значень, що знаходяться в інтервалі, в символ. Такий підхід краще використовувати, коли є чітка інформація про границі інтервалів, але природа інтервалів невідома і потребує дослідження

Відома функція відображення значень часового ряду в символи, але границі інтервалів невідомі. Цей підхід краще застосувати, якщо відома інформація про природу процесу, але необхідно дослідити його розгортання в часі.

Для одного процесу можна будувати обидва підходи, процедура моделювання передбачає оптимізацію та селекцію моделей.

Можливими функціями дискретизації часових рядів можуть бути:

Розбивка на основі похідної процесу – як першої (перша похідна як напрям розвитку процесу – зростання або спад), так другої (прискорення зміни), і можливо похідних інших порядків.

Розбивка на основі значень – розбивка точок даних на класи (кластеризація) беручи до уваги значення числового ряду в даній точці.

Розбивка на основі повторюваних кривих – розбивка здійснюється на основі співставлення інтервалів процесу і зарані

доступних типових кривих, що описують частини процесу.

Іншою проблемою є верифікація отриманої моделі. Хороша лінгвістична модель має володіти наступними характеристиками:

Включати в себе символи для всіх інтервалів, що об'єктивно існують – тобто бути не простішою, ніж процес, що описується.

Не включати в себе символи, що означають інтервали, не притаманні даному процесу. Імовірними джерелами таких інтервалів і символів, що їм відповідають, можуть бути шуми даних, а також вплив інших пов'язаних процесів.

Побудований набір символів має давати на виході такі послідовності, для яких можливо скласти граматику, придатну для прогнозування. Якщо ж це правило не виконується, то можливе або порушення попередньої характеристики, або ж обраний часовий ряд є випадковим за своєю структурою, і описати його з допомогою лінгвістичного моделювання неможливо.

ПОДАЛЬШІ НАПРЯМКИ РОБОТИ

Для перевірки гіпотези необхідно провести порівняльне дослідження пропонованого методу з існуючими методами аналізу часових рядів: регресія ковзного середнього, метод найменших квадратів, авторегресія.

Описаний метод найкраще підходитиме для процесів аперіодичних процесів із складною внутрішньою структурою, які мають виражені внутрішні стани – тобто такі, що можуть бути описані прихованими марківськими моделі. Для моделей із структурою, що описуватиметься простими математичними моделями, скоріше за все, лінгвістичне моделювання не матиме подібних переваг

ЗАСТОСУВАННЯ ЛІНГВІСТИЧНОГО МОДЕЛЮВАННЯ

Лінгвістичне моделювання має велику кількість можливих застосувань. Розглянемо тут деякі з них:

АВТЕНТИФІКАЦІЯ КОРИСТУВАЧА

Подається за [11]. В якості завдання стоїть створення елементів захисту інформаційної системи для ідентифікації користувача за динамікою його рухів маніпулятором миша, або пальцем чи стилусом по сенсорному екрану на основі сучасних методів математичного моделювання. Подальший розвиток теорії та практики захисту інформації пов'язаний з розвитком техніки та технології розпізнавання образів – так звані майнові (смарт-карти, електронні ключі) та біометричні методи автентифікації. Існує дві групи методів біометричної автентифікації – статичні та динамічні, які відповідно опираються на статичні ознаки, або ж на характеристики поведінки. Застосування лінгвістичного моделювання в даному контексті – розпізнавання розгортання поведінки користувача в часі, і класифікація паттернів поведінки в залежності від неї. Застосування ж лінгвістичного моделювання бачиться значно ширшим:

Автентифікація користувача при вході, за типом, що вже описаний, однак без використання реперних точок і з урахуванням динаміки руху. Це робить відтворення жесту іншою людиною практично неможливим при достатньому рівні точності.

Інформування і блокування несанкціонованої активності після входу. Це гадається ще більш перспективним напрямком. Активність користувача постійно контролюється, і при появі паттернів, що не належать користувачу (що означатиме несанкціонований доступ), можна вжити відповідних заходів (блокування пристрою до введення секретного коду, інформування користувача іншими засобами зв'язку). Перевагами такого підходу є непомітність для кінцевого користувача (відповідно зручність), складність для атаки, потенційними недоліками – можливість хибного спрацювання (можливо при знаходженні користувача в змінених емоційних станах).

ДІАГНОСТИКА ЗАХВОРЮВАНЬ ДРІБНОЇ МОТОРИКИ

Недоліки рухів можуть бути продіагностовані при роботі людини з електронними пристроями введення, такими як клавіатура, мишка, сенсорний

екран. Знімаючи параметри руху, такі як моментальне прискорення, можна перетворити порядок рухів в часовий ряд і провести його лінгвістизацію згідно схеми вище. Після лінгвістизації багатьох часових рядів, маючи групу де симптом проявляється і де він не проявляється, можна створити базу даних, яка відповідатиме рухами здорової людини, та іншу базу даних з записами рухів людини з діагностованими порушеннями. Маючи ці бази, можна реалізувати алгоритм класифікації рухів, що буде позначати рухи, підозрілі на порушення і сигналізувати про них.

РОЗПІЗНАВАННЯ ЕМОЦІЙНОГО СТАНУ ОПЕРАТОРІВ ВІДПОВІДАЛЬНОГО ОБЛАДНАННЯ

Особливу актуальність має під час використання обладнання, що прямо впливає на життя і здоров'я операторів важкої техніки і транспорту – літаків, кранів, вантажівок, екскаваторів і т.д. В цій техніці є різного роду механізми для вводу – рульові колеса, важелі, педалі. Знімаючи метрики з них, можна будувати часові ряди цих метрик, і застосувавши до них класифікатор, можемо отримати дані про емоційний стан оператора в реальному часі. Ці дані можна використати в кілька способів:

Блокування виконання небезпечних операцій, в разі якщо емоційний стан оператора незадовільний.

Оповіщення диспетчера про критичний стан оператора, і недопущення або зняття оператора з роботи.

Перемикання обладнання в режим автопілота, під час настання критичних обставин.

Принцип класифікації не має суттєво відрізнятися від діагностики рухових порушень, однак має бути більш точним в плані хибних спрацювань, що можуть погіршити зручність роботи з технікою. Скоріше за все, тут виникне необхідність робити компроміс між зручністю користування і безпекою, і вибір тут вже переходить до кінцевого користувача.

АНАЛІЗ ДАНИХ ПЕРЕНОСНИХ ПРИСТРОЇВ

Наразі все більшого і більшого поширення набувають портативні підключені до мережі Інтернет пристрої, в тому числі і такі,що мають датчики різного роду – відеокамери, мікрофони, температурні датчики, датчики наближення, руху, тиску, та ін. Більшість цих даних зберігаються в базах неструктурованими, і їхній аналіз – велике завдання, що гостро стоїть на сьогоднішній день. Лінгвістичний аналіз допоможе знайти закономірності в розгортанні процесів – лінгвістичні ланцюжки, опираючись на які можна впливати на системи. Переваги лінгвістичного аналізу – в простоті побудови моделі – мінімальна кількість припущень (тільки розмір інтервалу та спосіб розбиття), а на виході ми можемо отримати нові знання про систему, тобто лінгвістичне моделювання може застосувати не тільки для моделювання системи, а й для її дослідження.

ДІАГНОСТИКА СКЛАДНИХ ТЕХНІЧНИХ СИСТЕМ

Діагностика складних технічних систем, особливо в процесі їхньої роботи, є актуальною проблемою у зв'язку з поширенням великої кількості техніки. Прикладами такої діагностики можуть бути діагностика двигунів внутрішнього згорання – наприклад сигналізація зносу навантажених частин двигуна за звуком або характером вібрації двигуна. Лінгвістичне моделювання може визначати несправності, що розкинуті в часі, наприклад поломки, що проявляються в декілька фаз (послідовний знос деталі в часі, або знос деталі, що викликає підвищений знос іншої деталі).

ПРОГНОЗУВАННЯ ЕКОЛОГІЧНИХ ЯВИЩ

Проблема прогнозування екологічних часових рядів розглянута в [12]. Наведемо деякі аспекти, що окреслять використання лінгвістичного моделювання в даній області. Більшість методів прогнозування часових рядів довільної природи базується на внутрішній природі поведінки процесу, що виходить з його внутрішніх

закономірностей. Ці методи не дозволяють враховувати так звані “зовнішні впливи”, що можуть приводити до змін поведінки процесу (змін характеристик ряду, тренду, і таке ін.). Тобто за своїми якісними характеристиками поведінку таких часових рядів можна цілком вести на засадах феноменологічних підходів. В екологічному контексті, це можуть бути аномальні явища, що погано описуються за допомогою динаміко-аналітичних підходів, а через відсутність сталих періодів протікання, стає неможливим використання статистичних підходів. Це такі природні явища як цінами, землетруси, виверження вулканів, зсуви та ін. Маючи характеристики сейсмологічної інформації подані в вигляді лінгвістичного ряду, маючи деяку кількість історичних даних, можна прогнозувати настання таких природніх катастроф і завчасно готуватися до них. Недоліком наведеного підходу може бути брак історичної інформації. В разі відсутності достатньої кількості таких даних, необхідно використовувати методи системного аналізу.

СПИСОК ЛІТЕРАТУРИ

1. Fu K. S. A step towards unification of syntactic and statistical pattern recognition / K. S. Fu // IEEE Transactions on pattern analysis and machine intelligence. – 1986. – Vol. PAMI-8, № 3 (May). – P. 398–404.
2. Баклан І. В. Лінгвістичне моделювання: основи, методи, деякі прикладні аспекти / І. В. Баклан. // Системные технологии. – 2011. – № 3 (74). – С. 10–19.
3. Баклан І. В. Інтервальний підхід до побудови лінгвістичної моделі / І. В. Баклан. // Системные технологии. – 2013. – № 3 (86). – С. 3–8.
4. Шулькевич Т.В., Халимон А.Ю., Селін Ю.М., Недашківський Є.А., Баклан І.В. ПРОЦЕДУРА ЛІНГВІСТИЧНОГО МОДЕЛЮВАННЯ ДИНАМІЧНИХ ПРОЦЕСІВ РІЗНОЇ ПРИРОДИ // Інтелектуальні системи прийняття рішень і проблеми обчислювального інтелекту: Матеріали міжнародної наукової конференції. – Херсон: Видавництво ПП Вишемирський В. С., 2016. –С. 159-161.
5. Канторович Л. В. О некоторых новых подходах к вычислительным методам и обработке наблюдений / Л. В. Канторович. // Сибирский математический журналю – 1962. – Том III, № 5 (Сентябрь – Октябрь). – С. 701–709.
6. Добронец Б. С. Численные операции над случайными величинами и их приложения / Б. С. Добронец, О. А. Попова. // Journal of Siberian Federal University. Mathematics Physics. – 2011. – № 4 (2). – С. 229–239.
7. Ревенко Д. С. Статистическая оценка динамических процессов с неопределенными данными / Д. С. Ревенко, В. М. Вартанян, Ю. А. Романенков. // Економіка та управління підприємствами машинобудівної галузі: проблеми теорії та практики. – 2008. – № 4 (4). – С. 53–64.
8. Бідюк П. І. Системний підхід до прогнозування на основі моделей часових рядів / П. І. Бідюк. // Системні дослідження та інформаційні технології. – 2003. – № 3. – С. 88–110.
9. Баклан І. В. Ймовірнісні моделі для аналізу та прогнозування часових рядів / І. В. Баклан, Г. А. Степанкова. // Штучний інтелект. – 2008. – № 3. – С. 505–515.
10. Morton K.W. Scaling neutron tracks in Monte Carlo shielding calculations / Morton K.W. - J. Nuclear Energy. - 1957. - №3/4 - С.320-324.
11. Баклан І.В. Застосування лінгвістичного моделювання для автентифікації за динамікою рухів користувача /Баклан І.В., Селін Ю.М., Трохименко Ю.А. // Весник Херсонского национального университета. Вып. 3(50). - Херсон: ХНТУ, 2014. - С.117-121.
12. Селін Ю.М., Баклан І.В. Математичний апарат для прогнозування часових рядів економічного та екологічного типів, що можуть бути піддані зовнішнім впливам // Вестн. Херсонского нац. ун-та.–2013.– №2(47).–С.315-318.

СКЛАД

Організаційного комітету з проведення на ФІОТ наукової конференції студентів «Інформатика та обчислювальна техніка – ІОТ-2017»

Голова організаційного комітету: О.А. Павлов – декан.
Заст. голови організаційного комітету: О.М. Долголенко.
Співголови організаційного комітету:
Г.М. Луцький – професор кафедри ОТ;
О.М. Жданова – доцент кафедри АСОІУ.

Члени організаційного комітету: П.І.П.	Посада
Ю.О. Кулаков	Професор кафедри ОТ
В.І. Жабін	Професор кафедри ОТ
В.П. Симоненко	Професор кафедри ОТ
С.Г. Стіренко	Професор кафедри ОТ
В.М. Томашевський	Професор кафедри АСОІУ
І.В. Стеценко	Професор кафедри АСОІУ
О.В. Гавриленко	Доцент кафедри АСОІУ
Ю.О. Олійник	Старший викладач кафедри АСОІУ
М.О. Сперкач	Старший викладач кафедри АСОІУ
О.А. Халус	Старший викладач кафедри АСОІУ
К.Ю. Ларіна	Старший викладач кафедри АСОІУ

Секретарі конференції:	Посада
Н.Є. Куц	Пров. інженер кафедри ОТ
С. П. Якуніна	Інженер кафедри АСОІУ

Публікується у відповідності до розпорядження декана факультету ІОТ Павлова О.А. № 22 від «_29_»__03_2017 р.

Наукова конференція студентів, магістрантів та аспірантів «Інформатика та обчислювальна техніка – ІОТ-2017». Секція кафедри автоматизованих систем обробки інформації і управління. Матеріали конференції. – Київ. – 2017. – 25-27 квітня 2017р. – 263 с.

У збірник включені тези доповідей, які були представлені на конференції “Інформатика та обчислювальна техніка» – ІОТ-2017” в секції кафедри автоматизованих систем обробки інформації і управління. В доповідях розглянуті наукові та методичні питання щодо сучасних аспектів інформатики та обчислювальної техніки.

Редакційна колегія:
Жданова О.Г., к.т.н., доцент, кафедра АСОІУ НТУУ «КПІ ім. Ігоря Сікорського» – співголова
Олійник Ю.О., кафедра АСОІУ НТУУ «КПІ ім. Ігоря Сікорського»
Гавриленко О.В., к.ф.-м.н., кафедра АСОІУ НТУУ «КПІ ім. Ігоря Сікорського»