

# ЗАСТОСУВАННЯ ТЕОРІЇ ІГОР ПРИ БАНКРУТСТВІ ОРГАНІЗАЦІЇ

Тіщенко А.Є.<sup>1</sup>, Барановська Л.В.<sup>2</sup>

Національний технічний університет України  
«Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна

<sup>1</sup>anna16yarmola@gmail.com,

<sup>2</sup>lesia@baranovsky.org [0000-0003-0024-8180]

**У роботі розглядається використання теорії кооперативних ігор при банкрутстві організації. Обговорюється особливість таких випадків та складності в розподілі ліквідаційної вартості організації між кредиторами, позивачами та заявниками. Представлено рішення ситуації банкрутства певної організації за допомогою побудови динамічної моделі.**

**Ключові слова:** Кооперативні ігри, банкрутство, вектор Шеплі, динамічна модель, характеристична функція.

## 1. ВСТУП

Банкрутство фірми – це довгий і дуже складний процес. Воно настає, коли компанія не може розплатитися за своїми боргами. Цьому сприяють різні причини – підприємство стало збитковим, фінансова криза призвела до того, що власники не в змозі повернути борги банкам та іншим кредиторам.

Успіх діяльності фірми породжується різними зовнішніми та внутрішніми чинниками, і, якщо вона веде цю діяльність неефективно, настає момент, коли її потрібно вивести з ринку. І тому виконується послідовність дій, однією з є оцінка ліквідаційної вартості. Після цього постає питання про розподіл цієї вартості між заявниками, кредиторами та позивачами, що призводить до великої кількості юридичних конфліктів. Ця проблема полягає в тому, що вартості фірми здебільшого недостатньо, щоб виплатити всі борги. Необхідно встановити найприйнятніші правила розподілу.

У літературі представлено ряд таких правил, які використовуються на практиці, але необхідно ці правила якимось порівнювати між собою для визначення переваг та недоліків кожного з них. Найпоширеніше правило - правило пропорційності, у якому кошти між кредиторами розподіляються пропорційно вимогам. Але найчастіше цей поділ супроводжується великою кількістю суперечок між позивачами процедури банкрутства і тоді доводиться вдаватися до перегляду справ.

З іншого боку, мета розподілу – задовольнити потреби кредиторів, позивачів та заявників. А вони, в свою чергу, хочуть максимізувати свою частину виплат. Тож для вирішення цієї проблеми логічно використовувати математичне моделювання засобами теорії кооперативних ігор. Такі ігри моделюють ситуації, в яких учасники гри, об'єднуючись, можуть отримати додатковий прибуток.

Крім того, серйозним кроком у математичному моделюванні банкрутства фірм є розробка динамічних моделей, які враховують тимчасові витрати процесу, що розглядається. Якщо дослідити ліквідацію великих підприємств, можна помітити, що виплати боржникам відбуваються поетапно. Це може бути пов'язано з наявністю дочірніх фірм, дебіторськими заборгованостями, банківськими особливостями та іншими економічними факторами. Тоді необхідно розуміти, яку частину коштів потрібно виплачувати конкретному заявнику в

кожний момент часу. Звідси надходить ідея використання теорії багатокрокових кооперативних ігор.

## 2. ТЕОРІЯ КООПЕРАТИВНИХ ІГОР ТА ЇЇ ЗАСТОСУВАННЯ

Розглянемо моделювання ситуації банкрутства підприємства інструментами теорії кооперативних ігор. Цей метод має значення у різних галузях економічних наук. Він застосовується не тільки як вирішення загальногосподарських завдань, а й у дослідження стратегічних проблем ринків, галузей, підприємств, систем управлінського обліку, і форм стимулювання ефективної діяльності. За допомогою теорії ігор менеджмент підприємства отримує можливість передбачити ходи своїх партнерів та конкурентів.

Перед побудовою моделі необхідно згадати низку базових понять.

Під грою ми розумітимемо процес, у якому беруть участь дві і більше сторін, які ведуть боротьбу реалізації своїх інтересів. Нехай умови гри допускають спільні дії та перерозподіл виграшу. Головне завдання дослідження – це оптимальний розподіл благ між членами об'єднання.

Нехай  $N = \{1, 2, \dots, n\}$  - це множина всіх гравців у рамках цієї моделі. Тоді будь-яку непорожню підмножину  $S \subset N$  ми називатимемо коаліцією.

Під характеристичною функцією  $v$  будемо розуміти функцію, яка для кожної можливої коаліції ставить у відповідність дійсне число. Для будь-яких двох непересічних коаліцій  $T \subset N$  та  $S \subset N$  виконується нерівність:

$$v(T) + v(S) \leq v(T \cup S)$$

Це означає, що коаліція  $T \cup S$  має не менше можливостей, ніж дві коаліції, що не перетинаються,  $T$  і  $S$ , що діють поодиночці.

Тоді кооперативною грою назвемо пару  $(N, v)$  та визначимо її рішення. Найчастіше використовуються принципи оптимальності, такі як  $C$ -ядро,  $NM$ -рішення, вектор Шеплі. Але ми оберемо метод, який підходить для вирішення задачі справедливого розподілу і який гарантує єдиність рішення. Цей принцип вводиться аксіоматично.

Аксіоми Шеплі.

Аксіома 1. Якщо  $S$  - будь-який носій гри  $(N, v)$ , то виконується:

$$\sum_{i \in S} \varphi_i v = v(S).$$

Аксіома 2. Для будь-якої підстановки  $\pi$  і  $\forall i \in N$  вірно:

$$\varphi_{\pi(i)} \pi v = \varphi_i v.$$

Аксіома 3. Якщо  $(N, v)$  и  $(N, u)$  - дві довільні кооперативні ігри, то:

$$\varphi_i u + v = \varphi_i u + \varphi_i v.$$

Нехай  $\varphi$  - це функція, яка ставить у відповідність згідно з аксіомами Шеплі будь-якої гри  $(N, v)$  вектор  $\varphi(v)$ . Тоді цей вектор називатимемо вектором Шеплі гри  $(N, v)$ .

Сформулюємо просте завдання повернення боргів кредиторам. Характеристичну функцію такої гри побудуємо виходячи з вже існуючих правил розподілу. Кожній коаліції поставимо у відповідність поступки кредиторів, які не входять до об'єднання, що розглядається. У разі коли поступка матиме негативне значення, поставимо нульове значення коаліції. Отже, отримуємо наступну характеристичну функцію:

$$v^{d,E} S = \max E - d_i, 0, i \in N \setminus S. \quad (1)$$

### 3. ПРИКЛАД РОЗПОДІЛУ ЛІКВІДАЦІЙНОЇ ВАРТОСТІ ОРГАНІЗАЦІЇ ЗА ДОПОМОГОЮ ПОБУДОВИ ДИНАМІЧНОЇ МОДЕЛІ

Деяка будівельна компанія подала до суду на прийняття її банкрутом. Ліквідаційна вартість її становить 38041000 гривень. Вимоги позивачів, кредиторів та заявників рівні (13 787 000, 18 655 537, 37 530 244). Відомо, що грошова сума надходить на рахунок боржника у два етапи: продаж основного майна та повернення заборгованостей з боку іншої фірми. Таким чином,  $E = (E(t_1), E(t_2)) = (26801000, 11240000)$ . Розрахуємо виплати трьом агентам за допомогою вищезгаданого кооперативного метода.

Крок 1. Розглянемо першу гру  $\Gamma^1 = (N, v^{(d^1, E^1)})$ , де вектор  $d^1 = (13\ 787\ 000, 18\ 655\ 537, 37\ 530\ 244)$ , а  $E^1 = E(t_1) + E(t_2)$ . Будуємо характеристичну функцію (табл. 1).

Таблиця 1. Характеристична функція для кроку 1

Коаліція	Значення
$v(\emptyset)$	0
$v(1)$	0
$v(2)$	0
$v(3)$	5 598 463
$v(1,2)$	510 756
$v(1,3)$	19 385 463
$v(2,3)$	24 254 000
$v(1,2,3)$	38 041 000

Розрахуємо вектор Шеплі

$$Sh_1^1 = \frac{1}{3} \cdot -24254000 + \frac{1}{6} \cdot 510756 - 5598463 + \frac{1}{6} \cdot 19385463 + \frac{1}{3} \cdot 38041000 = 6\ 978\ 626.$$

$$Sh_2^1 = \frac{1}{3} \cdot -19385463 + \frac{1}{6} \cdot 510756 - 5598463 + \frac{1}{6} \cdot 24254000 + \frac{1}{3} \cdot 38041000 = 9412894,5.$$

$$Sh_3^1 = \frac{1}{3} \cdot 510756 - 5598463 + \frac{1}{6} \cdot 19385463 + \frac{1}{6} \cdot 24254000 + \frac{1}{3} \cdot 38041000 = 21\ 649\ 479,5.$$

Вважаємо, що ми зробили виплати кожному кредитору, тобто. отримали вектор виплат  $x^1 = (x_1^1, x_2^1, x_3^1)$ . Чисельне значення вектора ми дізнаємося на наступному кроці. Але при побудові наступної гри врахуємо ці виплати та віднімемо їх із початкових вимог гравців.

Крок 2. Розглянемо гру  $\Gamma_2$ . Зменшуємо розподільну суму  $E_2 = E(t_2)$  та обчислюємо новий вектор вимог

$$d^2 = (13787000 - x_1^1, 18655537 - x_2^1, 37530244 - x_3^1).$$

Будуємо характеристичну функцію (табл. 2) та знаходимо вектор Шеплі:

Таблиця 2. Характеристична функція для кроку 2

Коаліція	Значення
$v(\emptyset)$	0
$v(1)$	0
$v(2)$	0
$v(3)$	0
$v(1,2)$	$-26\,290\,244 + x_3^1$
$v(1,3)$	$-7\,415\,537 + x_2^1$
$v(2,3)$	$-2\,547\,000 + x_1^1$
$v(1,2,3)$	11 240 000

$$Sh_1^2 x^1 = \frac{-2x_1^1 + x_2^1 + x_3^1}{6} - \frac{2043927}{2}.$$

$$Sh_2^2 x^1 = \frac{x_1^1 - 2x_2^1 + x_3^1}{6} - 1412305.$$

$$Sh_3^2 x^1 = \frac{x_1^1 + x_2^1 - 2x_3^1}{6} - \frac{21699317}{2}.$$

Складаємо різницю  $Sh^1 - Sh^2 x^1 = x^1$ , вирішуємо систему з трьох рівнянь із трьома невідомими:

$$x^1 = (7067500, 7067500, 12666000).$$

Отже, отримано виплати для кожного гравця у перший момент надходження.

Користуючись цим же алгоритмом, не складно розрахувати вектор виплат на другому кроці:

$$x^2 = (2239833\frac{1}{3}, 4500083\frac{1}{3}, 4500083\frac{1}{3})$$

Винесемо повне рішення багатокрокової гри та порівняємо його з результатами при однокроковому розподілі:

Таблиця 3. Результати однокрокової та багатокрокової гри

Період виплати	Вектор виплат
$t_1$	(7 067 500, 7 067 500, 12 666 000)
$t_2$	(2 239 833.3, 4 500 083.3, 4 500 083.3)
Сумарна виплата	(9 307 333.3, 11 567 583.3, 17 166 083.3)
Однокрокова гра	(6 978 626, 9 412 894.5, 21 649 479.5)

## ВИСНОВКИ

На підставі проведеного дослідження можна зробити такі висновки.

Проблема розподілу коштів при банкрутстві підприємства є актуальною та активно обговорюється у літературі. У роботі зроблено аналіз чинних правил розподілу фіксованої суми серед позивачів, що висунули свої вимоги. Крім їх перерахування, розглянуто аксіоматичний метод порівняння.

Крім того, були розроблені багатокрокові теоретико-ігрові моделі, що враховують складність надходження коштів на рахунок боржника. Дані моделі ґрунтуються на принципі оптимальності – векторі Шеплі, який гарантує існування та єдиність рішення. На числовому прикладі показано, що такі моделі допомагають справедливо розподілити доступну суму кожному проміжку часу.

## ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. O'Neill A problem of rights arbitration from the Talmud // *Mathematical Social Sciences* , 2011. No 2. P. 345 – 371.
2. Gintis, H. *Game theory Evolving*. — Princenton : Princenton University Press, 2000.
3. Aumann R., Maschler M. E. Game theoretic analysis of a bankruptcy problem from the Talmud // *Journal of Economic Theory*, 1985. Vol. 36, No 1. P. 195 – 213.
4. Herrero C., Villar A. The three musketeers: four classical solutions to bankruptcy problems // *Mathematical Social Sciences*, 2001. Vol. 39, No 3. P. 307 – 328.
5. Петросян Л. А., Зенкевич Н. А., Шевкопляс Е. В. *Теорія ігор*. М.: БХВ-Петербург, 2014.- 423 с.
6. Thomson W. Axiomatic and Game-theoretic Analysis of Bankruptcy and Taxation Problems // *Mathematical Social Sciences*, 2003. Vol. 45, No 3. P. 249 – 280.

# МЕТОДИ І МЕХАНІЗМИ УПРАВЛІННЯ І ПІДТРИМКИ КОМПЛЕКСОМ УПРАВЛІННЯ СКЛАДОМ

Торліна Н.М.<sup>1</sup>, Мухін В.Є.<sup>2</sup>

Національний технічний університет України  
«Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна

<sup>1</sup> nadiia.torlina@gmail.com, <sup>2</sup> v.mukhin@kpi.ua [0000-0002-1206-9131]

**Організація складу є важливим етапом розвитку будь-якої фірми, адже складські операції є однією з найважливіших складових в ціноутворенні товару. Тому мінімізації витрат на шляху товару від виробника до кінцевого споживача можна досягти навіть якщо не оптимально організований процес купівлі-продажу, виробництва, фінансових розрахунків, тощо. Метою роботи є аналіз процесу впровадження та підтримки комплексу управління складом для мінімізації часу та зусиль витрачених на дані процеси на основі методу з використанням механізмів управління. Результатом дослідження є програмний продукт, що автоматизує процеси під час впровадження та підтримки системи управління складом. У роботі було використано емпіричні та теоретичні методи дослідження.**

**Ключові слова:** система управління складом, WMS, механізм управління, впровадження, підтримка, комплекс управління складом.

## 1. ВСТУП

Система управління складом – це система із застосуванням комп'ютеризованих технологій, за допомогою якої оптимізується та автоматизується вся складська діяльність. Скорочена назва цієї системи – WMS (англ. Warehouse Management System – система управління складом) [1].

Економічний сенс впровадження комплексу управління складом є всюди, де може здійснюватись зберігання, переміщення та облік товарів, відправлень, сировини, тощо, в тому числі архівів. Застосування комплексу управління складом має сенс не тільки у великих логістичних центрах або підприємствах чи складах оптової торгівлі, а також у невеликих складах, які мають невеликі операційні обсяги або невелику кількість товару. Адже незалежно від кількості товару, замовлень, розмірів приміщення, автоматизація значно зменшує витрати часу, і внаслідок цього коштів [2].

Однак успіх впровадження системи WMS базується на деяких організаційних та технічних факторах. Наприклад, пошук часу на впровадження, вибір досвідченого керівника проекту, пошук та навчання співробітників, пошук консультантів та команди впровадження, тестування системи.

Крім того, перед початком впровадження комплексу управління складом потрібно переконатись, що процеси загалом підібрані правильно. Якщо ж компанія вже використовує якусь систему автоматизації, наприклад ERP, і бажає інтегрувати її в нову систему, необхідно переконатись, що дані системи сумісні і можуть використовуватись разом без перебоїв. Також важливо строго дотримуватись методів управління проектом. Незважаючи на складність, системи WMS пропонують підприємствам значні переваги. Буде не тільки

скорочено час циклу розміщення та видалення, але й покращиться точність інвентаризації. Це на додаток до збільшення обсягу зберігання, збільшення організованості зберігання матеріалів і більшої гнучкості складських операцій.

## **2. МЕТОДОЛОГІЯ ВПРОВАДЖЕННЯ І ПІДТРИМКИ КОМПЛЕКСУ УПРАВЛІННЯ СКЛАДОМ**

Складність впровадження і підтримки комплексу управління складом залежить від проекту, бізнесу та доступних людських ресурсів. Тому для успішного впровадження системи управління складом потрібно гарантувати, що ключовий персонал, необхідний для впровадження зможе виконувати поставлені задачі швидко та якісно.

Загалом проект з впровадження комплексу управління складом можна поділити на такі ключові етапи [3]:

- Ініціація;
- Планування та дизайн;
- Реалізація;
- Валідація;
- Введення в експлуатацію.

На етапі ініціації проекту відбувається схвалення проекту, обирається команда з впровадження. Зазвичай команда з впровадження складається з керівника проекту, спеціаліста WMS, дизайнера рішень WMS, спеціаліста з інтеграції, спеціаліста з автоматизації. На цьому етапі проводиться огляд процесів та операцій, які мають бути сконфігуровані в системі, попередньо оцінюється складність проекту і ризики. Складність та ризики оцінюються також наприкінці етапу ініціації. Як результат етапу ініціації проект має бути повністю визначений.

Етап планування і дизайну передбачає створення концепції та вимог щодо майбутніх процесів і операцій. На цьому етапі, а саме на початку та в кінці проводиться огляд та оновлення складності і ризиків проекту. Крім того, на даному етапі виявляються та узгоджуються функції, для яких необхідна розробка.

На етапі реалізації середовище для інтеграції системи має бути готове до налаштування, а також команда впровадження повинна мати список конфігурації, яку треба налаштувати в системі. Прогрес налаштування системи відстежується протягом всього етапу. Коли конфігурація проекту повністю виконана, вона має бути перенесена вручну з середовища для інтеграції в середовище для тестування, на якому відбувається етап валідації проекту. Також в кінці етапу реалізації проводиться демонстрація сконфігурованих процесів для команди з впровадження та операторів системи управління складом для валідації налаштованих процесів та перевірки чи залишилися певні недоліки в процесах.

Етап валідації передбачає тестування всіх процесів, що були налаштовані в системі. Також на цьому етапі спеціаліст WMS переносить вручну налаштування з середовища для тестування до робочого середовища.

На етапі введення в експлуатацію система управління складом відбувається процес останньої перевірки системи, та безпосередньо введення в експлуатацію [3].

Кожен з останніх трьох етапів: реалізації, валідації і введення в експлуатацію, може займати від двох тижнів до декількох місяців, що збільшує вартість проекту.

Після успішного запуску комплексу багато підприємств виявляють, що ресурсів, необхідних для підтримки роботи системи WMS, більше, ніж до впровадження. Насамперед це пов'язано з інтенсивним об'ємом даних програмного забезпечення та тим фактом, що ситуація на складі постійно змінюється: додаються нові елементи, розробляються нові процеси, тощо [3].

Після впровадження проект підтримується операторами складу та спеціалістами WMS. У разі виникнення проблем, спеціалісти WMS мають бути навчені вирішувати їх і якщо першопричину не вдається знайти, звертатись до служби підтримки. Зазвичай технічна підтримка складається з декількох рівнів підтримки для того щоб швидко вирішувати дрібні або прості проблеми, і визначати відповідних спеціалістів та дії для складніших проблем [3]:

- 0 лінія підтримки: самостійна підтримка локальних спеціалістів WMS, використовуючи інформацію з керівництва користувача, поширених запитань, форумів, залучення соціальних контактів, тощо;

- 1 лінія підтримки: базова служба підтримки найбільш частих проблем користувачів та виконання запитів служби підтримки, які потребують участі ІТ;

- 2 лінія підтримки: поглиблена технічна підтримка більш досвідченими спеціалістами системи управління складом проблем користувачів, які не можуть бути вирішені попередньою лінією підтримки;

- 3 лінія підтримки: експертна підтримка продуктів і послуг для вирішення проблем або створення нових функцій залежно від виявлених першопричин;

- 4 лінія підтримки: зовнішня підтримка проблем за контрактом для товарів, що надаються організацією, але не обслуговуються безпосередньо, включаючи підтримку принтерів, підтримку програмного забезпечення постачальників, технічне обслуговування та інші послуги стороннього виконавця.

Компанії можуть змінювати цей шаблон та поєднувати декілька рівнів підтримки для економії часу та людських ресурсів на вирішення проблем. Також може використовуватись часова шкала і пріоритетність для передачі проблеми на наступний рівень. Наприклад, якщо рішення проблеми з високим пріоритетом займає більше 2 годин у рівня 0, вона має бути передана на рівень 1 [4].

В даному дослідженні проаналізовано витрати на впровадження та підтримку комплексу управління складом використовуючи наведену методологію. А також використано підхід на основі методу з використанням технічних механізмів управління та представити, як підприємство може скоротити витрати на впровадження та підтримку комплексу управління складом. У дослідженні використовуються дані по загальним витратам на впровадження системи, витратам на період окремих етапів впровадження та витрати на підтримку WMS.

### **3. РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ**

Для дослідження ефективності програмного продукту було обрано дані, що базуються на даних логістичної компанії, яка займається впровадженням і підтримкою комплексом управління складом. Було обрано 5 проектів, які впроваджувались у період 2020-2022 року. А також представлено дані витрат на підтримку системи управління складом за 2020-2022 роки.

Аналіз існуючих підходів до оцінки логістичних витрат показує, що переважна більшість дослідників [5], [6], [7] при визначенні даного виду витрат зосереджує увагу, здебільшого, на ключових логістичних активностях або ж на конкретному переліку бізнес процесів, що здійснюються у межах підприємства.

У таблиці 1 наведено витрати на впровадження комплексу управління складом на п'ятьох проектах різних за сферою та обсягом без використання технічних механізмів управління.

Таблиця 1. Витрати на впровадження комплексу управління складом без використання технічних механізмів управління

№ проекту	Початок проекту	Кінець проекту	Загальна кількість робочих днів	Вартість послуг за робочий день (дол.)	Вартість комплексу управління складом (дол./місяць)	Загальна вартість проекту (дол.)
1	29.09.2020	01.04.2021	128	1000	2000	136533
2	16.11.2020	26.07.2021	172	1000	2000	183352
3	01.02.2021	15.07.2021	113	1000	2000	120458
4	23.02.2021	14.01.2022	223	1000	2000	237718
5	03.05.2021	05.01.2022	169	1000	2000	180154

Розглянемо кожен проект на етапі реалізації більш детально (Табл. 2).

Таблиця 2. Витрати на етапі реалізації проекту без використання технічних механізмів управління

№ проекту	Початок етапу	Кінець етапу	Загальна кількість робочих днів	Кількість годин на порівняння середовищ	Вартість послуг за робочий день (дол.)	Вартість комплексу управління складом (дол./місяць)	Грошові витрати на порівняння середовищ (дол.)
1	23.10.2020	08.03.2021	93	58	1000	2000	7728,5
2	11.03.2021	22.06.2021	70	50	1000	2000	6662,5
3	12.06.2021	28.06.2021	9	8	1000	2000	1066
4	08.04.2021	30.07.2021	77	56	1000	2000	7462
5	09.06.2021	06.12.2021	125	760	1000	2000	101270

В розробленому програмному продукті порівняння середовищ займає від 1 до 10 хвилин, в залежності від кількості даних з налаштувань системи управління складом. Витрати на етапі реалізації проекту з використанням технічного механізму управління представлені у таблиці 3.

Таблиця 3. Витрати на етапі реалізації проекту з використанням технічних механізмів управління

№ проекту	Загальна кількість днів (робочих)	Кількість годин на порівняння середовищ	Вартість послуг (дол./робочий день)	Вартість комплексу управління складом (дол./місяць)	Грошові витрати на порівняння середовищ (дол.)
1	86	0,16	1000	2000	21,32
2	64	0,16	1000	2000	21,32
3	8	0,16	1000	2000	21,32
4	70	0,16	1000	2000	21,32
5	95	0,16	1000	2000	21,32

В якості кількості годин на порівняння середовищ взято максимальне значення у 10 хвилин. В результаті в середньому грошові витрати за час відведений на порівняння середовищ можна скоротити у більш ніж 100 разів.

У таблиці 4 наведено порівняння витрат на впровадження комплексу управління складом без та з використанням технічних механізмів управління.

Таблиця 4. Порівняння витрат на впровадження комплексу управління складом

	Без використання технічних механізмів управління	З використанням технічних механізмів управління
Середні витрати робочих днів на впровадження комплексу управління складом	161	151
Середні грошові витрати на впровадження комплексу управління складом (дол.)	180421	146805

В результаті використання технічних механізмів управління на етапі реалізації витрати часу зменшились на 6,2%, а грошові витрати на 18,6%.

У таблиці 5 розглядаються витрати на підтримку комплексу управління складом за 2020-2022 роки.

Таблиця 5. Витрати на підтримку комплексу управління складом без використання технічних механізмів управління

Рік	Тип проблеми	Кількість запитів	Мінімальний час на вирішення 1 проблеми (год)	Середній час на вирішення 1 проблеми (год)	Максимальний час на вирішення 1 проблеми (год)	Вартість послуг зовнішньої підтримки (дол/год)
2020	Налаштування	35	0,8	312	2736	1000
2021	Налаштування	44	0,25	192,75	1210	1000
2022	Налаштування	38	1.23	446	3718,5	1000

Розглянуто лише запити з типом проблеми «Налаштування», що передбачає випадки, коли конфігурацію системи було змінено і процес припинив працювати коректно. В середньому за три роки середній час на вирішення 1 проблеми становить 316,9 годин. Також варто зазначити, що час на вирішення запиту залежить від кількості ескалацій проблеми на вищі рівні підтримки.

В таких випадках також можна використати розроблений програмний продукт для порівняння середовищ і виявлення конфігурації, яка була змінена і скорочення часу на пошук проблеми. Зовнішню підтримку в такому випадку можна не залучати до вирішення проблеми. Отже грошові витрати на послуги зовнішньої підтримки для типу проблеми «Налаштування» будуть дорівнювати 0. Що призведе до загального скорочення грошових витрат на зовнішню підтримку комплексу управління складом. Порівняння середовищ, як і раніше, займає від 1 до 10 хвилин. Зміна відповідної конфігурації та перевірка коректності роботи процесу займе в середньому 1 робочий день або 8 годин. Отже, витрати часу на підтримку комплексу управління складом в середньому зменшаться у 40 разів.

## 4. ВИСНОВКИ

На сьогоднішній день автоматизація складських процесів грає важливу роль в розвитку підприємства, адже вона скорочує час на виконання складських операцій, збільшує точність та забезпечує безпеку товару, що зберігається на складі. Впровадження системи управління складом має економічний сенс всюди, де здійснюється зберігання, облік і переміщення будь-яких одиниць зберігання. Організація технічної підтримки також грає важливу роль при впровадженні комплексу управління складськими процесами.

Впровадження і підтримка системи управління складом є складним завданням, що залежить від багатьох факторів, зокрема проекту, бізнесу та доступних людських ресурсів.

В даній роботі було досліджено та проаналізовано витрати на впровадження та підтримку комплексу управління складом, використано підхід на основі методу з використанням технічних механізмів управління та представлено, як підприємство може скоротити витрати на впровадження та підтримку системи управління складом за допомогою автоматизації процесів під час впровадження та підтримки системи управління складом.

### ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Warehouse management system. URL: [https://en.wikipedia.org/wiki/Warehouse\\_management\\_system](https://en.wikipedia.org/wiki/Warehouse_management_system) (дата звернення: 16.11.2022)
2. Дудар Т. Г., Основи логістики: навч. посіб. Тернопіль: Економічна думка, 2006. 163 с.
3. 7 Steps Towards a Successful WMS Implementation. URL: <https://www.linkedin.com/pulse/7-steps-towards-successful-wms-implementation-marcil-msc-> (дата звернення: 16.11.2022)
4. IT Support Levels Clearly Explained: L1, L2, L3 & More. URL: <https://www.bmc.com/blogs/support-levels-level-1-level-2-level-3/> (дата звернення: 16.11.2022)
5. Шевців Л. Ю., Петецький І., Логістичні витрати підприємства: формування та оцінювання, монографія. Львів: НУ «Львівська політехніка», 2011. 244 с.
6. Сумець О. М., Логістичні витрати підприємства: теоретичний аспект, монографія. Харків: КП «Міськдрук», 2013. 224 с.
7. Лифар В. В., Комерційна логістика та методика розрахунку логістичних витрат. Вісник Національного Університету «Львівська політехніка». 2011. Вип. 416. С. 293 – 297.

# СИСТЕМА ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ ДЛЯ КОРОТКОСТРОКОВОГО ПРОГНОЗУВАННЯ НЕЛІНІЙНИХ НЕСТАЦІОНАРНИХ ПРОЦЕСІВ В ЕКОНОМІЦІ ТА ФІНАНСАХ

Харченко Р.А.<sup>1</sup>, Бідюк П.І.<sup>2</sup>

Національний технічний університет України  
«Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна

<sup>1</sup>romankharchenko99@gmail.com,

<sup>2</sup>pbidyuke@gmail.com [0000-0002-7421-3565]

**Метою дослідження є розробка гнучкої системи підтримки прийняття рішень (СППР) для короткострокового прогнозування фінансово-економічних процесів. Побудова моделей та оцінювання прогнозів в рамках СППР передбачає використання гнучкого підходу на базі декомпозиції часового ряду та використанні ряду моделей різних класів. Система надає інструменти для попереднього аналізу даних, автоматизованого підбору структурних параметрів моделей, візуалізації та дозволяє будувати високоточні короткострокові прогнози не вимагаючи від користувачів спеціальних знань в галузі прогнозування.**

**Ключові слова:** СППР, декомпозиція часового ряду, ARIMA, SVR, LSTM.

## 1. ВСТУП

Прогнозування фінансово-економічних процесів є однією з найбільш актуальних задач аналізу даних в цілому. Оцінки прогнозів динаміки розвитку процесів у економіці та фінансах є вхідними даними для вирішення задач планування, прийняття рішень, інвестування, оцінювання ризиків тощо. Універсального рішення, яке буде оптимальним для будь-яких вхідних даних та в кожному окремому контексті вирішення задачі прогнозування цих процесів не існує. Це зумовлено тим, що фінансово-економічним процесам властиві такі характеристики як нестационарність, нелінійність, наявність сезонних та циклічних патернів, залежність від великої кількості зовнішніх, як правило невимірюваних, впливів та збурень. Задачі прогнозування фінансово-економічних процесів та часових рядів у цілому присвячено досить багато літературних джерел, наприклад [1] містить докладні відомості про природу фінансово-економічних процесів у цілому та особливості їх моделювання, у роботах [2, 3] наведено докладні відомості про випадкові процеси, підходи до побудови моделей, прогнозування, критерії якості моделей, надано практичні рекомендації. Згадані вище роботи хоча і надають практичні рекомендації щодо використання методу або групи методів для вирішення задачі прогнозування, не приділяють достатньої уваги безпосередній реалізації методів, підбору гіперпараметрів моделей тощо. Спроектована і реалізована в даній роботі СППР дозволяє використовувати множину моделей різних класів, інструменти для попереднього аналізу даних, автоматизованого підбору структурних параметрів моделей і при цьому не потребує поглиблених знань та навичок в галузі моделювання і прогнозування.

## 2. ПРОГНОЗУВАННЯ ФІНАНСОВО-ЕКОНОМІЧНИХ ЧАСОВИХ РЯДІВ НА БАЗІ ДЕКОМПОЗИЦІЇ

Ідея декомпозиції часового ряду полягає у його розкладенні на набір компонент, які, як правило, є зручнішими та простішими з точки зору моделювання та прогнозування. Виділяються наступні 3 компоненти: компонента тренду  $T$ , сезонна компонента  $S$ , залишкова компонента  $R$ .

Компонента тренду описує загальну тенденцію зміни значень ряду, сезонна компонента описує регулярні патерни, які повторюються через рівні проміжки часу, до залишкової компоненти відносять всі впливи, які не враховані іншими компонентами, наприклад невимірювані впливи та випадкові збурення тощо. Іноді також виділяється циклічна компонента, яка описує досить тривалі патерни, які не обов'язково повторюються через однакові проміжки часу, однак в даній роботі вважаємо, що компонента тренду враховує циклічні зміни.

Використання декомпозиції часового ряду також забезпечує гнучкість процесу моделювання та прогнозування. Наприклад, можна використовувати різні моделі в тому числі з різних класів моделей для прогнозування кожної з компонент. Виокремлення залишкової компоненти дозволить зменшити вплив невимірюваних збурень на якість моделей. На практиці найбільш поширеними є адитивна та мультиплікативна декомпозиції, адитивна декомпозиція полягає у представленні вихідного часового ряду у вигляді суми компонент  $T$ ,  $S$ ,  $R$ , в той час як мультиплікативна у вигляді їх добутку. Вибір того чи іншого підходу залежить від характеру досліджуваних даних, наприклад, існує емпіричне правило, яке стверджує, що у випадку, якщо амплітуда сезонних коливань практично не змінюється в часі, то варто обирати адитивну модель, якщо амплітуда залежить від тренду, то кращим вибором буде мультиплікативна модель [4]. Однак не всі методи декомпозиції передбачають мультиплікативний варіант, в такому випадку можна використовувати операцію логарифмування для представлення мультиплікативної декомпозиції у вигляді адитивної використовуючи таке співвідношення:

$$\log(y) = \log(S * T * R) = \log(S) + \log(T) + \log(R). \quad (1)$$

Надалі за замовчуванням матимемо на увазі адитивну декомпозицію, враховуючи, що за потреби можна отримати мультиплікативний варіант за виразом (1).

Формування остаточної оцінки прогнозу на базі декомпозиції також є досить гнучким і дозволяє використовувати дещо різні підходи. На практиці одним з найбільш поширених є підхід, що передбачає прогнозування компоненти сезонності та вихідного ряду з вилученою сезонною компонентою. Даний підхід може бути оптимальним у випадку якщо залишкова компонента не є цілком випадковою та може містити корисну інформацію про властивості динаміки розвитку процесу. Інший підхід полягає у відкиданні залишкової компоненти та прогнозуванні лише компоненти тренду та сезонної компоненти, в такому випадку фактично використовується фільтрація часового ряду з вилученою сезонною компонентою. Кожен з згаданих вище підходів реалізовано в рамках розробленої СППР.

Незалежно від обраного з наведених вище підходів, остаточна оцінка прогнозу для вихідного ряду формується як сума прогнозованих значень компонент. Прогнозування сезонної компоненти часто зводиться до використання найвісного прогнозу, тобто повторення відповідних значень останнього сезонного циклу. Використання найвісного прогнозу для сезонної компоненти базується на припущенні про незмінність або дуже повільні зміни сезонних впливів, що часто, хоча і не завжди, виконується на практиці, крім того, деякі методи декомпозиції, не передбачають врахування зміни сезонних впливів за побудовою.

В реалізованій системі підтримки прийняття рішень користувач має змогу обрати метод для декомпозиції часового ряду, зокрема імплементовано метод класичної декомпозиції [4] та метод STL [5].

Метод класичної декомпозиції є досить простим та інтуїтивно зрозумілим, невимогливым до обчислювальних ресурсів. Ідея методу базується на використанні ковзних середніх для оцінювання тренду та оцінюванні сезонної компоненти шляхом усереднення значень факторів сезонності, в якості факторів сезонності можуть розглядатися дні тижня, місяці, пори року тощо в залежності від досліджуваної довжини сезонного циклу та дискретизації вимірів ряду. Основним недоліком цього методу є неможливість оцінювання декілька перших та останніх вимірів, що пов'язано з використанням центрованих ковзних середніх, для отримання оцінок останніх вимірів, які є особливо важливими для прогнозування, можна використовувати не центровані ковзні середні, а усереднювати значення лише попередніх вимірів, також можна використовувати методи екстраполяції. Іншими недоліками є надмірне згладжування тренду, неможливість врахування змін сезонних впливів, низька робастність.

Метод STL також базується на процедурі згладжування, проте алгоритм роботи методу значно складніший, ніж у випадку класичної декомпозиції. Цей метод є універсальним та гнучким, при цьому не є надто вимогливим до обчислювальних ресурсів. Основними перевагами методу є можливість врахування сезонних циклів при будь-якому періоді дискретизації вимірів, врахування змін сезонних впливів з часом, невисока у порівнянні з методом класичної декомпозиції чутливість до екстремальних значень та нетипових вимірів даних. Серед недоліків методу можна виділити відсутність автоматичного врахування "календарних ефектів" та відсутність мультиплікативного варіанту декомпозиції. Однак, використовуючи перетворення, наприклад за виразом (1), або в більш загальному випадку перетворення Бокса-Кокса можна отримати результат, подібний до мультиплікативної декомпозиції або навіть деякий проміжний результат між варіантами адитивної та мультиплікативної декомпозицій [7].

В рамках реалізованої СППР користувач має змогу обирати декілька моделей різних класів для побудови оцінок прогнозів компоненти тренду або вихідного ряду з вилученою сезонністю та сезонної компоненти. Зокрема для прогнозування тренду можуть використовуватись класична регресійна модель ARIMA [6] та її різновиди, а саме моделі AR та MA, модель SVR [7], що є модифікацією методу опорних векторів для вирішення задачі регресії та нейронну мережу LSTM [8]. Для прогнозування сезонної компоненти реалізовано модель наївного прогнозу, SVR та LSTM. Відмінності в наборі моделей для прогнозування сезонної компоненти зумовлені тим, що наївний прогноз є оптимальним варіантом за умови припущення про незмінність сезонних впливів, а також може ефективно використовуватись у разі повільної зміни сезонних впливів. Модель ARIMA не була обрана для прогнозування сезонної компоненти, оскільки за умови зміни сезонних впливів з часом, сезонна компонента буде гетероскедастичною, оскільки змінюватиметься амплітуда коливань циклів, що негативно впливатиме на ефективність використання моделі.

### **3. ОПИС РЕАЛІЗОВАНОЇ СППР ТА РЕЗУЛЬТАТИ ОБЧИСЛЮВАЛЬНИХ ЕКСПЕРИМЕНТІВ**

СППР призначена для математичного моделювання та прогнозування нелінійних нестационарних процесів у економіці та фінансах, динаміка яких представлена у вигляді одновимірних часових рядів. Для реалізації СППР було обрано мову програмування Python, взаємодія з користувачем відбувається за допомогою віконного інтерфейсу, розробленого за допомогою фреймворку PyQt. Функціональна схема реалізованої системи наведена на Рис. 1.

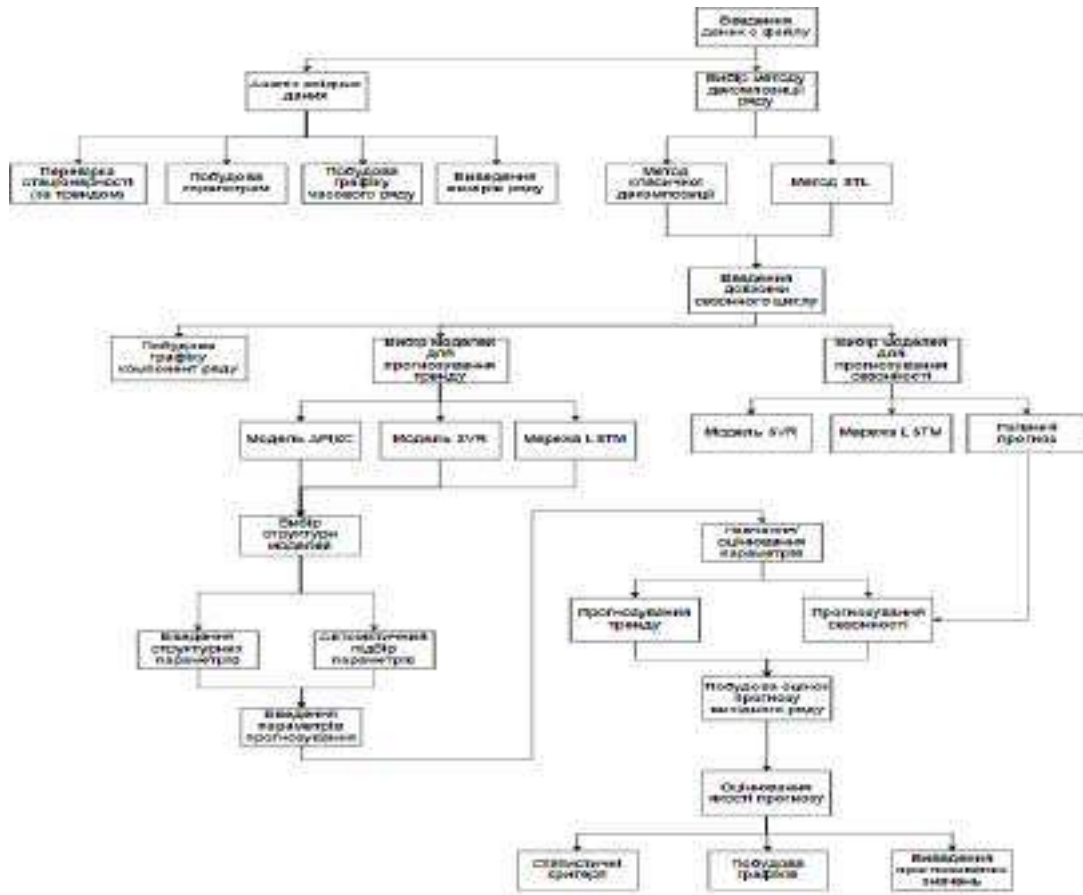


Рисунок 1. Функціональна схема СППР

Окрему увагу слід приділити структурним параметрам моделі, якість підбору яких безпосередньо впливає на якість побудови моделей та оцінок прогнозів. В рамках системи передбачено два варіанти визначення гіперпараметрів — введення користувачем вручну та автоматизований підбір. Для автоматизованого підбору параметрів моделі ARIMA система встановлює порядок тренду шляхом застосування оператора взяття різниць до тих пір, поки в результаті не утвориться стаціонарний за трендом ряд, для підбору порядку авторегресійної частини та ковзного середнього система будує ряд моделей на базі ЧАКФ, якщо при деякому значенні лагу абсолютне значення більше 0.15, то досліджуються моделі авторегресії, ковзного середнього та авторегресії і ковзного середнього, порядок параметрів яких відповідає значенню цього лагу. Для вибору кращої моделі використовується значення інформаційного критерію AIC.

Підбір параметрів мережі LSTM та моделі SVR виконується за допомогою процедури баєсівської оптимізації. В якості функції, яку необхідно мінімізувати, використовується значення метрики MAPE, для моделювання цільової функції в рамках процедури використовується гаусівський процес. В поточній реалізації системи для виконання процедури баєсівської оптимізації використовується програмний модуль для бібліотеки Python sickit-optimize. Для мережі LSTM реалізовано можливість підбору кількості нейронів та довжини вікна вхідних даних, для моделі SVR — тип ядерної функції та довжина вікна вхідних даних.

Наведемо результати роботи СППР на прикладі часового ряду, що описує кількість безробітних у відсотках від працездатних осіб в США, період дискретизації вимірів — місяць. Ряд містить дані на проміжку з 1 січня 2008 року по 1 січня 2020, усього 145 вимірів,

горизонт прогнозування – 5 місяців. Графік ряду та компонент, отриманих в результаті декомпозиції за методом STL з довжиною сезонного циклу 12 наведено на Рис. 2.

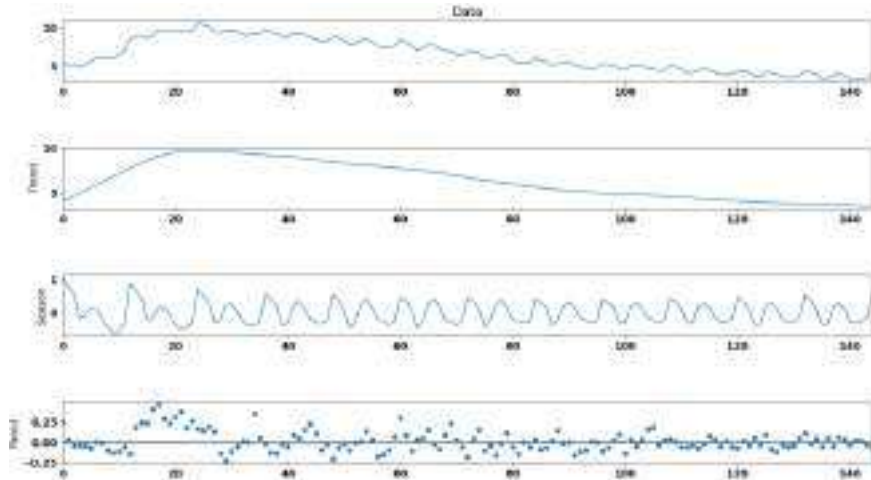


Рисунок 2. Графік досліджуваного ряду та його компонент

Для прикладу виберемо моделі та інші параметри системи, як зображено на Рис. 3.

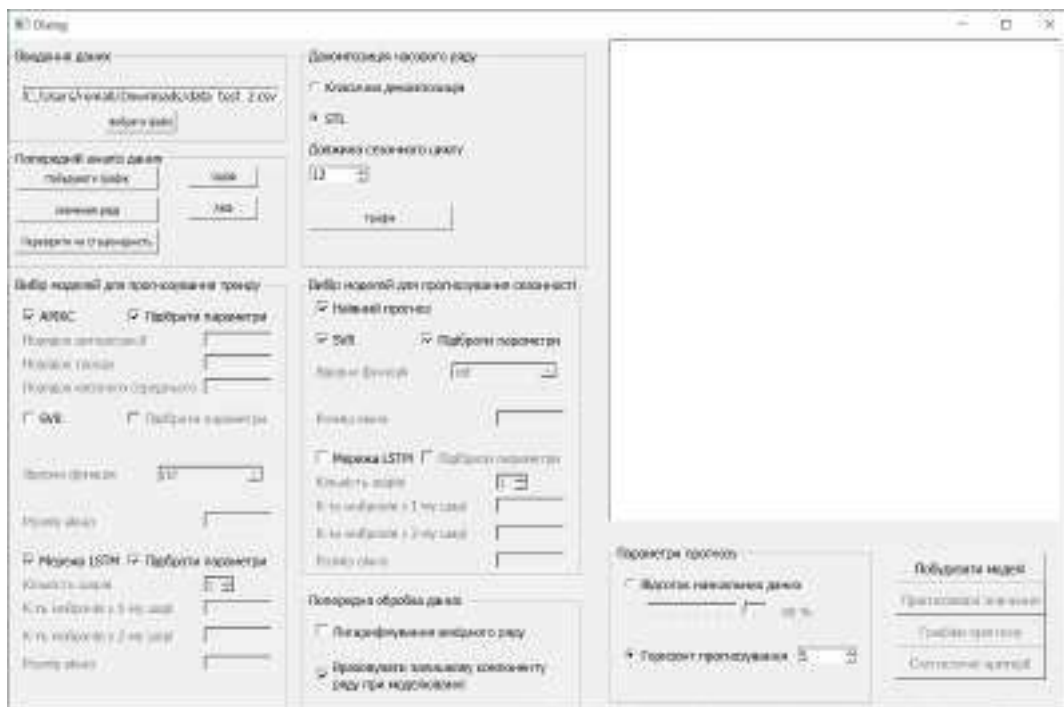


Рисунок 3. Головне вікно програми

На Рис. 4 наведено графіки значень оцінок прогнозів та тестової множини.

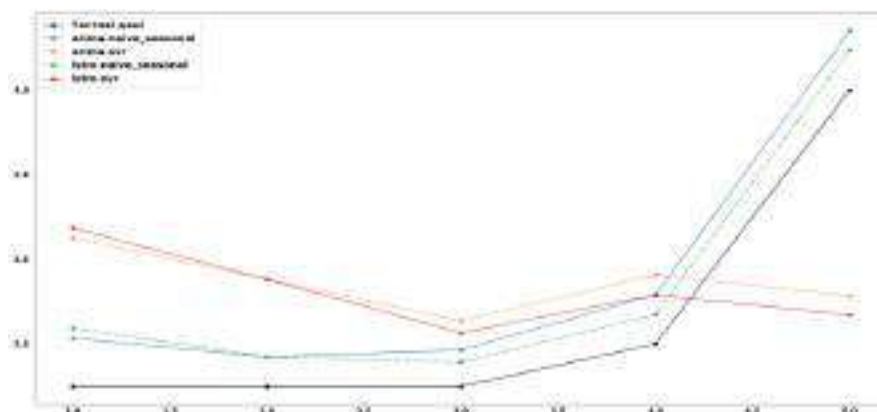


Рисунок 4. Графіки прогнозованих значень тестової множини

Статистичні критерії якості прогнозів наведено у Табл. 1.

Таблиця 1. Статистичні критерії якості прогнозів (перша модель у назві використовується для прогнозування тренду, друга – сезонності)

Метрика \ Модель	MAPE	RSME	ME	MPE	MSE	SSE
ARIMA-naive	3.020509	0.108212	-0.105291	-3.020509	0.011710	0.058549
ARIMA-SVR	7.982132	0.307560	-0.087178	-3.127477	0.094593	0.472965
LSTM-naive	2.488964	0.090508	-0.085868	-2.488964	0.008192	0.040959
LSTM-SVR	7.908984	0.321255	-0.067754	-2.595933	0.103205	0.516025

#### 4. ВИСНОВКИ

В рамках проведеного дослідження було реалізовано систему підтримки прийняття рішень для короткострокового прогнозування нелінійних нестационарних процесів в економіці та фінансах. Підхід на базі декомпозиції часового ряду, імплементований в даній системі, дозволяє отримувати високоточні оцінки прогнозів, про що свідчать статистичні критерії та графік прогнозованих значень, отримані в результаті обчислювального експерименту. Висока якість прогнозів зокрема зумовлена гнучким підходом до побудови моделей. Автоматизовані процедури підбору параметрів дозволяють покращити якість моделей та прогнозів та значно спрощують процес моделювання, надаючи змогу користувачеві використовувати систему без поглиблених знань в області моделювання та прогнозування, однак система дозволяє за бажання користувача самостійно задавати структурні параметри. Також в рамках системи наявні модулі для попереднього аналізу даних, побудови графіків.

Отже, система відповідає поставленим до неї вимогам, для подальшого вдосконалення системи доцільно додати можливість працювати з багатовимірними часовими рядами, розширити набір доступних моделей та вдосконалити процедури підбору параметрів, наприклад розширити множину гіперпараметрів, які можуть бути оптимізовані.

#### ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Cıpra T. Time Series in Economics and Finance. Springer, 2020. 353 с.
2. Бідюк П. І., Романенко В. Д., Тимошук О. Л. Аналіз часових рядів : навч. посіб. Київ, Київ обл. : Політехніка, 2010. 317 с.
3. Montgomery D. C., Jennings C. L., Kulahci M. Introduction to Time Series Analysis and Forecasting. Wiley & Sons, Incorporated, John, 2011. 472 с.

4. Kotu V., Deshpande B. Data Science: Concepts and Practice. Morgan Kaufmann, 2018. 568 с.
5. STL: A Seasonal-Trend Decomposition Procedure Based on Loess / R. B. Cleveland та ін. Journal of Official Statistics. 1990. Т. 6, № 1. С. 3–31. URL: <https://www.scb.se/contentassets/ca21efb41fee47d293bbee5bf7be7fb3/the-usa39s-bicentennial-census-new-directions-for-methodology-in-1990.pdf> (дата звернення: 13.11.2022).
6. Hyndman R. J., Athanasopoulos G. Forecasting: Principles and Practice. 2-ге вид. Melbourne : OTexts, 2018. 382 с. URL: <https://otexts.com/fpp2/index.html> (дата звернення: 13.11.2022). Time Series Forecasting Using LSTM Networks: A Symbolic Approach
7. Awad M., Khanna R. Support Vector Regression. Efficient Learning Machines. Berkeley, CA, 2015. С. 67–80. URL: [https://doi.org/10.1007/978-1-4302-5990-9\\_4](https://doi.org/10.1007/978-1-4302-5990-9_4) (дата звернення: 13.11.2022).
8. Elsworth S., Güttel S. Time Series Forecasting Using LSTM Networks: A Symbolic Approach. arxiv.org. URL: <https://arxiv.org/abs/2003.05672> (дата звернення: 13.11.2022).

# **АНАЛІЗ ВПЛИВУ ВІЙСЬКОВОГО КОНФЛІКТУ НА СОЦІАЛЬНО-ЕКОНОМІЧНІ ПРОЦЕСИ В УКРАЇНІ**

Худіков П.В.<sup>1</sup>, Бідюк П.І.<sup>2</sup>

Національний технічний університет України  
«Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна

<sup>1</sup>pavelkhudikov77@gmail.com

**На даний момент війна в Україні впливає на економічні та соціальні процеси, тому важливо їх досліджувати і спробувати спрогнозувати розуміння подальших дій в цій ситуації. Об'єктом дослідження в роботі є соціально-економічні процеси в Україні, зокрема такі: ВВП, Інфляція, кількість біженців. Метою роботи є розробка і застосування математичних моделей для оцінювання впливу війни на соціально-економічні процеси в Україні та інших розвинутих країнах. Результатом дослідження є побудована система прогнозування, що дає можливість прогнозувати вплив військового конфлікту на соціально-економічні процеси в Україні. У роботі використано теоретичні та емпіричні методи дослідження.**

**Ключові слова: військовий конфлікт, соціально-економічні процеси, задача прогнозування, математичні моделі.**

## **1. ВСТУП**

Війна в Україні стала жорстоким випробуванням на виснаження військових, яка призвела до значної військової підтримки України з боку США та Європи. Однак це лише частина історії. Війна також перетворилася на великий політичний та економічний конфлікт між Заходом і Росією, який зрештою може мати набагато більший вплив на глобальну стабільність і політичну стратегію США та Європи, ніж самі бойові дії. Кінцевим результатом є те, що громадянська сторона війни тепер виходить далеко за межі територій, де відбувалися військові конфлікти та удари по українських містах під час перших етапів війни. Зараз війна змінила міжнародні витрати на енергоносії на глобальній основі, призвела до створення глобального дефіциту продовольства, завдала серйозної шкоди всій російській економіці та допомогла спровокувати значне зростання рівня інфляції на глобальному рівні.

## **2. МЕТОДИ ПРОГНОЗУВАННЯ СОЦІАЛЬНО-ЕКОНОМІЧНИХ ПРОЦЕСІВ**

Соціально-економічні процеси – це зміни в суспільстві та економіці, які відображаються на рівні життя учасників цих процесів, стабільності політичної та економічної ситуації в країні, умовах безпеки та захищеності її населення. Ці процеси обумовлюють соціальні зміни, у яких відображаються соціально-економічні властивості політичних процесів [1].

Для прогнозування цих процесів доречно використовувати регресійні моделі, які будуються на основі статистичних даних.

Лінійна парна регресія намагається змоделювати взаємозв'язок між двома змінними шляхом підгонки лінійного рівняння до спостережуваних даних. Одна змінна вважається пояснювальною, а інша - залежною.

Діаграма розсіювання може бути корисним інструментом у визначенні міцності зв'язку між двома змінними. Якщо, як видається, немає зв'язку між запропонованими пояснювальними та залежними змінними (тобто графік розсіювання не вказує на якісь тенденції до збільшення чи зменшення процесів), то пристосування моделі лінійної регресії до даних, ймовірно, не забезпечить корисної моделі. Цінною числовою мірою асоціації між двома змінними є коефіцієнт кореляції, який приймає значення від -1 до 1, що вказує на силу асоціації спостережуваних даних для двох вибраних змінних.

Логістична регресія – це нелінійна множинна регресія, яка аналізує функціональну залежність між двома або частіше декількома незалежними змінними (регресорами) і залежною змінною. В свою чергу бінарна логістична регресія застосовується для класифікації у тому випадку, коли вихідна змінна може приймати тільки два значення. Найчастіше у задачах кредитування застосовується саме бінарна логістична регресія [2].

Нас цікавить ймовірність появи події у залежності від значень змінних  $x = x_1, x_2, \dots, x_n$ . Виходом є використання логіт-функції. Вона приймає значення від 0 до 1.

Нейронна мережа - це множина алгоритмів, які намагаються розпізнати основні взаємозв'язки у вибірці даних за допомогою процесу, що імітує роботу людського мозку. У цьому сенсі нейронні мережі відносяться до систем нейронів, органічних або штучних за своєю природою. Нейронні мережі можуть адаптуватися до змінних входів; таким чином мережа дає найкращий можливий результат без необхідності переробляти критерії формування висновку. Концепція нейронних мереж, корінням якої є штучний інтелект, стрімко набирає популярність зокрема у розвитку торгових систем.

Нейронні мережі у світі фінансів допомагають розв'язувати такі задачі, як прогнозування часових рядів, прогнозування соціально-економічних процесів, алгоритмічна торгівля, класифікація цінних паперів, моделювання кредитного ризику та оцінювання власних показників і похідних цін.

Метод групового врахування аргументів (МГВА) базується на задаванні правил ускладнення моделі, системі опорних функцій, критерію селекції та методу регуляризації згідно зовнішнім критеріям. ЕОМ проводить генерування моделей-претендентів, селекцію згідно зовнішнім критеріям та відсів моделей, що не пройшли селекцію. В зв'язку з цим основну структуру алгоритму самоорганізації можна навести у такому вигляді:

1. попередня обробка спостережень з урахуванням системи обраних опорних функцій (скорочується кількість претендентів);
2. генерування множини моделей-претендентів;
3. обчислення критеріїв селекції, що є зовнішніми доповненнями, та пошук моделі оптимальної складності.

Головна ідея МГВА полягає в наступному: стверджується, що для задачі однократного прогнозу доцільно знизити точність визначення оцінок коефіцієнтів рівняння регресії, але за рахунок цього придати йому більшу регулярність. Тому нашою метою в цій задачі є не мінімізація похибок на вже відомих вузлах інтерполяції, а мінімізація похибок на нових точках, які ми в момент синтезу рівняння регресії ще не мали відповідних значень [2].

Random forest (англ. випадковий ліс) — ансамблевий метод машинного навчання для класифікації, регресійного аналізу та інших завдань, який працює за допомогою побудови численних дерев прийняття рішень під час тренування моделі й продукує моду для класів (класифікацій) або усереднений прогноз (регресію) на побудованих деревах. Недоліком цього методу є схильність до перенавчання [3].

### 3. РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Для дослідження було взято дані з сайту Світового Банку, Trading Economics, Управління Верховного комісара ООН у справах біженців. В даних було отримано таку інформацію:

- ВВП України (1991-2022 рр.);
- промисловість (включаючи будівництво), додана вартість (% ВВП) (1991-2022 рр.);
- площу земель сільськогосподарського призначення (кв. км) (1991-2022 рр.);
- об'єм виробництва, додана вартість (% ВВП) (1991-2022 рр.);
- кількість українських біженців у різних країнах Європи (станом на березень 2022 р.).

Для коректного проведення регресійного аналізу досліджено наявність можливих лінійних зв'язків між змінними, які в подальшому пояснюватимуть поведінку змінної (табл. 1), та визначити взаємозв'язок ризику з іншими змінними та змінних між собою.

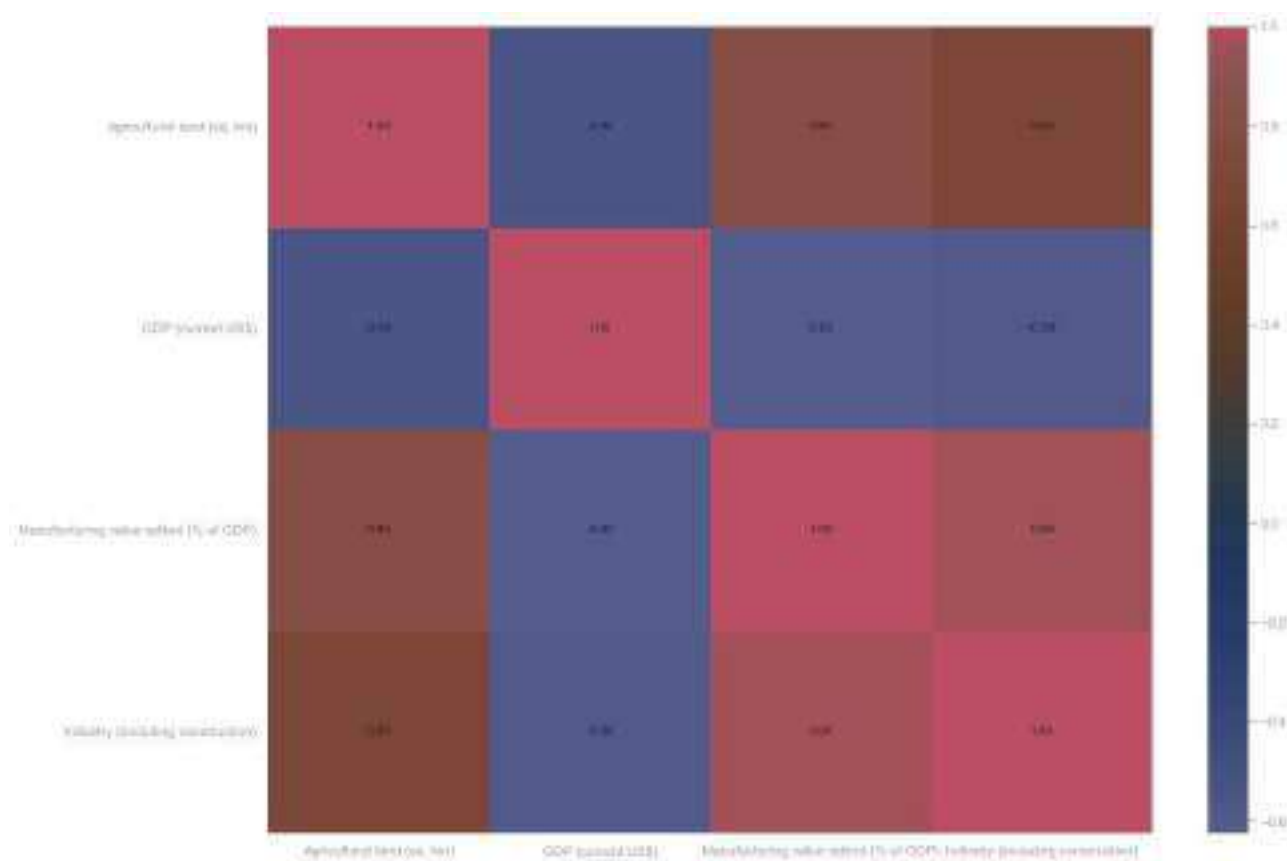


Рисунок 1. Кореляція між змінними

Визначено, що згідно з кореляційними показниками найбільше ВВП залежить від промисловості. Автокореляційний аналіз є важливим кроком у дослідницькому аналізі даних прогнозування часових рядів. Він допомагає виявити шаблони та перевірити на випадковість. Це особливо важливо, коли є намір використовувати модель авторегресії з ковзним середнім (АРКС) для прогнозування, оскільки це допомагає визначити структуру і параметри моделі. Аналіз передбачає перегляд графіків автокореляційної функції (ACF) і часткової автокореляційної функції (PACF).

При оцінюванні спектра параметрів ARMA спочатку оцінюються параметри AR, а потім параметри MA оцінюються на основі цих параметрів AR. Потім отримують спектральні оцінки моделі ARMA. Тому оцінка параметра моделі MA часто розраховується як процес асоціації із спектром параметрів моделі ARMA.

Таблиця 2. Автокореляція та часткова автокореляція

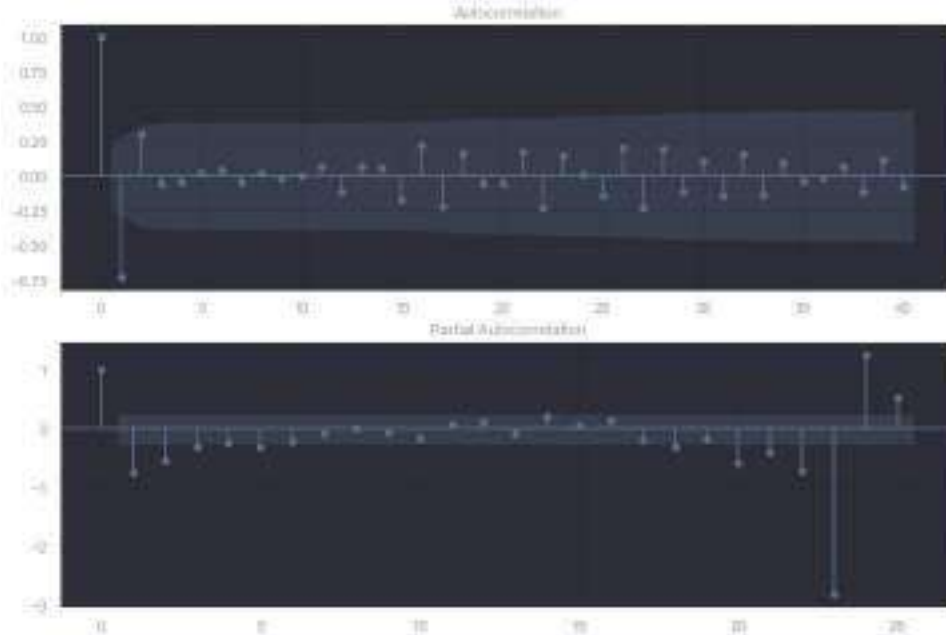


Рисунок 2. Автокореляція та часткова автокореляція

ARMA — це модель прогнозування, у якій методи аналізу авторегресії (AR) і ковзного середнього (MA) застосовуються до даних часових рядів, які добре поведуться. В ARMA припускається, що часовий ряд є стаціонарним і, коли є коливання, то це відбувається рівномірно протягом певного часу.

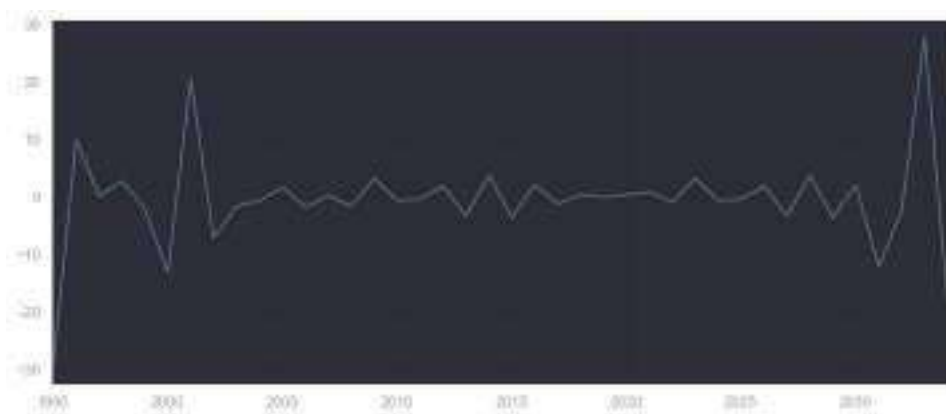


Рисунок 3. Прогноз росту ВВП(%) України

Після очищення даних і дослідницького аналізу даних було проведено побудову наступних регресійних моделей для прогнозування річної зміни відсотка ВВП країн, в які виїхали біженці з України: лінійна регресія; регресія КНН; регресія на деревах рішень; Random Forest; регресія Adaboost; Lasso та Ridge. Базова модель, зміна середньорічного відсотка ВВП, дала середньоквадратичну похибку 2,46%. Середньоквадратична похибка

вказує на те, що отримані похибки приблизно на 2,46% менші від фактичної річної процентної зміни ВВП.

Спрогнозовано річну зміну ВВП у 2022 році для країн, на які сильно вплинули українські біженці. Залишається оцінити, наскільки це точно, оскільки накопичується більше біженців і нові дані використовуються для прогнозів. Для оцінювання прогнозів було використано кількість біженців від 22 березня 2022 року, а також умовні середні або медіанні значення для відсутніх економічних характеристик.

	Country	Refugees under UNHCR's mandate	GDP Annual Change
0	Poland	2083854	3.368897
1	Romania	535461	3.410381
2	Hungary	312120	3.743343
3	Slovakia	250036	3.054826
4	Russian Federation	231764	2.449776
5	Belarus	3765	2.378090

Рисунок 4. Прогноз змін відсотка ВВП країн

Використано дані Організації Об'єднаних Націй, щоб забезпечити початкове дослідження того, як потоки біженців можуть вплинути на економіку приймаючих країн.

Після порівняння та налаштування моделей Random Forest було визнано найкращим, оскільки він дає меншу середньоквадратичну помилку 2,15% для тестових даних. Крім того, модель може пояснити 38,65% дисперсії нашої цілі, що краще, ніж базова модель.

Є впевненість у адекватності побудованої моделі, усвідомлюємо обмеження наших даних і часу. Щоб рухатися вперед із точнішою моделлю, необхідно збирати більше даних і провести глибше занурення. Також необхідно використати моделі часових рядів, щоб дослідити вплив минулих і теперішніх потоків біженців на майбутнє зростання ВВП.

## 4. ВИСНОВКИ

На сьогоднішній день маємо одну з головних проблем України і усього світу - війна. Виникає проблема, що в ході бойових дій страждає економіка, інфраструктура і, що найважливіше, людські життя. Тому виникає потреба дослідити масштаб трагедії і спрогнозувати можливі результати війни хоча б у короткостроковій перспективі.

У ході дослідження було реалізовано такі моделі, які дозволяють виконувати оцінку економічних показників України та інших країн. Для цього було використані дані Світового Банку, Управління Верховного комісара ООН у справах біженців, використані алгоритми регресійного аналізу та машинного навчання.

## ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Прийма С.С., Приймак В.І. Теоретичні основи трактування поняття “Соціально-економічні процеси в ринковій економіці”.
2. Кузнецова Н.В. Системний підхід до аналізу характеристик моделей оцінювання ризиків кредитування.
3. Random Forest. URL: [https://ru.wikipedia.org/wiki/Random\\_forest](https://ru.wikipedia.org/wiki/Random_forest)

# СИСТЕМНИЙ ПІДХІД ЩОДО ПРОЕКТУВАННЯ АРХІТЕКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Шахворостова В.Д.<sup>1</sup>, Тимошук О.Л.<sup>2</sup>

Національний технічний університет України  
«Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна

<sup>1</sup> vdshakh@gmail.com,

<sup>2</sup> o.tymoshchuk@kpi.ua [0000-0003-1863-3095]

**Мікросервісна архітектура** – популярний підхід до розробки програмного забезпечення. Його основна риса полягає у тому, що додаток розбивається на невеликі автономні компоненти (мікросервіси) з чітко визначеними інтерфейсами. Система будується як набір незалежних і слабозв'язаних сервісів, які можна створювати, використовуючи різні мови програмування та технології для збереження даних.

**Ключові слова:** мікросервісна архітектура, мікросервіси, патерни проектування програмного забезпечення, хмарні провайдери.

## 1. ВСТУП

Архітектурний стиль мікросервісів — це підхід, при якому єдиний додаток будується як набір невеликих сервісів, кожен з яких працює у власному процесі та комунікує з іншими, використовуючи прості механізми, як правило HTTP. Ці сервіси побудовані навколо бізнес-потреб і розгортаються незалежно, з використанням повністю автоматизованого середовища. Самі по собі ці сервіси можуть бути написані на різних мовах і використовувати різні технології збереження даних.

Найкраще всього порівняти мікросервісний підхід з монолітним – додатком, що побудований як єдине ціле [1]. Монолітний сервер — досить очевидний спосіб побудови подібних систем [1]. Вся логіка по обробці запитів виконується в єдиному процесі, при цьому є можливість використовувати переваги обраної мови програмування. Монолітні додатки можуть бути успішними, однак все більше людей розчаровуються у них, особливо враховуючи те, що все більше додатків розгортаються у хмарі. Будь-які зміни, навіть найменші, вимагають передеплою та розгортання всього моноліту. З часом зберігання якісної модульної структури стає все складнішою задачею: зміна логіки одного модулю мають тенденцію впливу на код інших. Тож доводиться масштабувати увесь додаток цілком, навіть за умови, що це потрібно лише для одного модулю цього додатку.

## 2. ПІДХОДИ ДО ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Шаблони проектування програмного забезпечення — це загальні багаторазові рішення поширених проблем у проектуванні програмного забезпечення. Шаблони дизайну допомагають ділитися спільним словником та використовувати перевірене на практиці рішення замість того, щоб щоразу експериментувати та ризикувати.

Патерном проектування називається певний звичний спосіб вирішення конкретної проблеми чи виклику, які регулярно можна зустріти при проектуванні архітектури програмного забезпечення [2].

У рамках аналізу підходів до проектування програмного забезпечення, вдалось виділити наступні шаблони:

1. Агрегатор.
2. Посередник.
3. Ланцюжок.
4. Гілка.
5. Дані, що розділяються.
6. Асинхронні повідомлення.

Для подальшої роботи було прийнято рішення орієнтуватись на патерни агрегатор, ланцюжок та дані, що розділяються. Надалі походу реалізації програмного забезпечення було остаточно визначено з яким саме із наведених патернів була продовжена робота, крім того, також розглядався варіант комбінації підходів. Прийняття рішення базувалось на тому який варіант був найефективнішим – чиста реалізація конкретного патерну чи їх комбінація.

### 3. ХМАРНА ІНФРАСТРУКТУРА

Хмарні послуги – це інфраструктура, платформи або програмне забезпечення, які розміщуються сторонніми постачальниками та надаються користувачам через Інтернет [3].

На сьогоднішній день на глобальному ринку хмарних технологій домінують три основних постачальника, які ділять 64% ринку [4]:

1. Amazon Web Services або AWS.
2. Microsoft Azure.
3. Google Cloud Platform (GCP).

Згідно з Synergy Research Group ситуація на ринку нині має вигляд, що проілюстрований на рисунку 1.

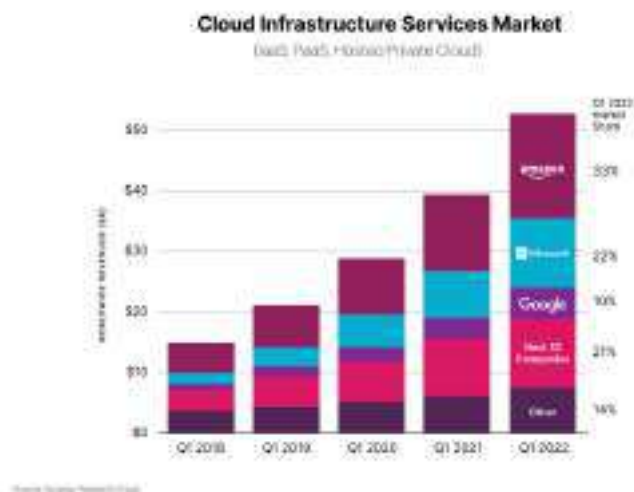


Рисунок 1. Динаміка популярності різних хмарних сервісів впродовж останніх п'ятьох років

Amazon Web Services (AWS) пропонує комп'ютерні ресурси та послуги, які можуть створювати програми за лічені хвилини з оплатою за використання [4]. Наприклад, є можливість орендувати сервер на AWS для підключення, налаштування, захисту та роботи так само, як фізичний сервер. Відмінність полягає в тому, що віртуальний сервер працює поверх мережі планетарного масштабу, керованої AWS.

Microsoft Azure — це загальнодоступна хмарна платформа, яка надає рішення інфраструктури як послуги (IaaS), платформи як послуги (PaaS) і програмного забезпечення як послуги (SaaS) для аналітики, віртуальних обчислень, зберігання, мереж та інших служб. Він може покращити або замінити локальні сервери [4].

Google Cloud, спочатку App Engine, — це набір хмарних сервісів, створений Google у 2008 році [4]. GCP пропонує підприємствам у всьому світі інфраструктуру як послугу (IaaS), платформу як послугу (PaaS) і програмне забезпечення як послугу (SaaS). . Наприклад, GCP — це в першу чергу служба для розробки та підтримки оригінальних програм, які потім можна публікувати з його гіпермасштабованих центрів обробки даних.

#### 4. АНАЛІЗ РЕЗУЛЬТАТІВ ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

На початку розробки було свідомо прийняте рішення щодо відмови від графічного інтерфейсу. Під час імплементатії перевірка роботоспроможності написаного перевірялась у різних форматах та у кілька етапів.

Спершу коректна робота написаного коду перевірялась локально – запуск програми у Терміналі через команду «go run main.go» та співставлення очікуваного результату з фактичним: зміни у базі, зміна у поведінці на фронті відповідно до нового стану бази.

Наступний етап полягав у деплої нового функціоналу на проект у Google Cloud та у подальшій перевірці в умовах, що найближчі до оточення продакшену.

Після успішного деплою проводилось тестування функціоналу у нових умовах. У нагоді стала програма Postman, адже саме її можливості дозволяють тестувати API. Отже, вказавши URL для запиту, тіло самого запиту та пройшовши авторизацію, вдалось отримати відповідь та статус 200OK, що проілюстровано на рисунку 2. Перевіривши зміни у базі, вдалось переконатись що частина написаного функціоналу справді відпрацював коректно, тож можна рухатись далі та розробляти наступні елементи для повноцінного функціонування програмного забезпечення.

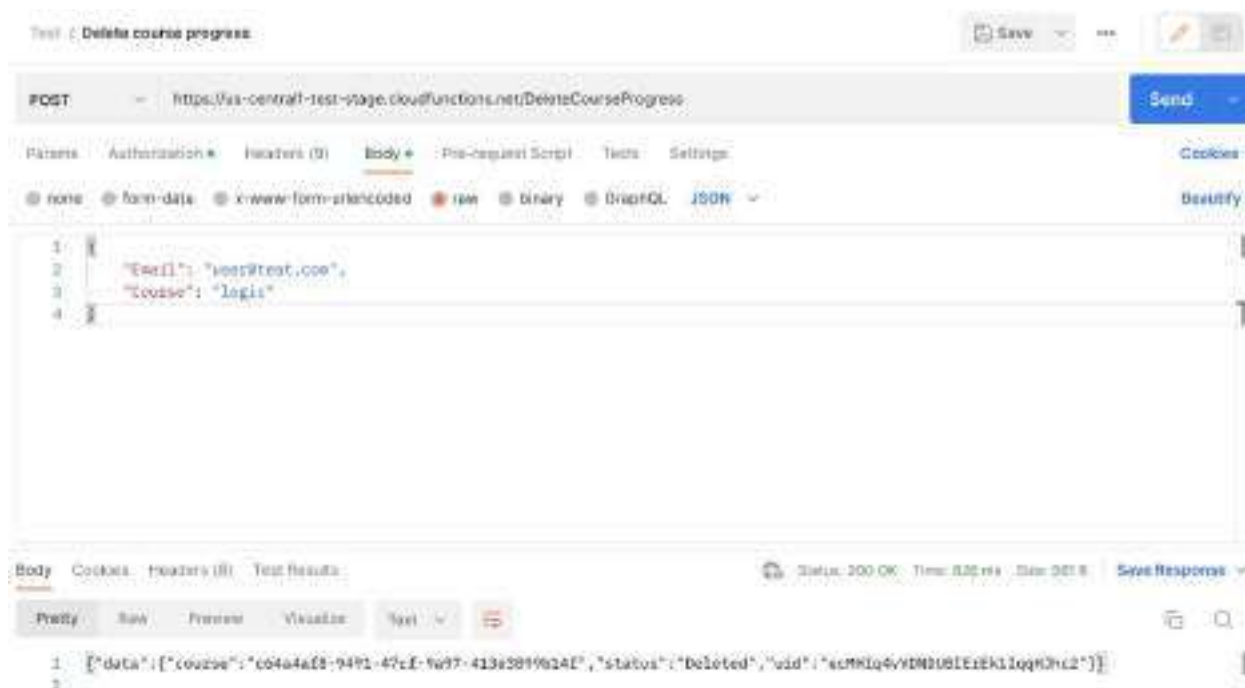
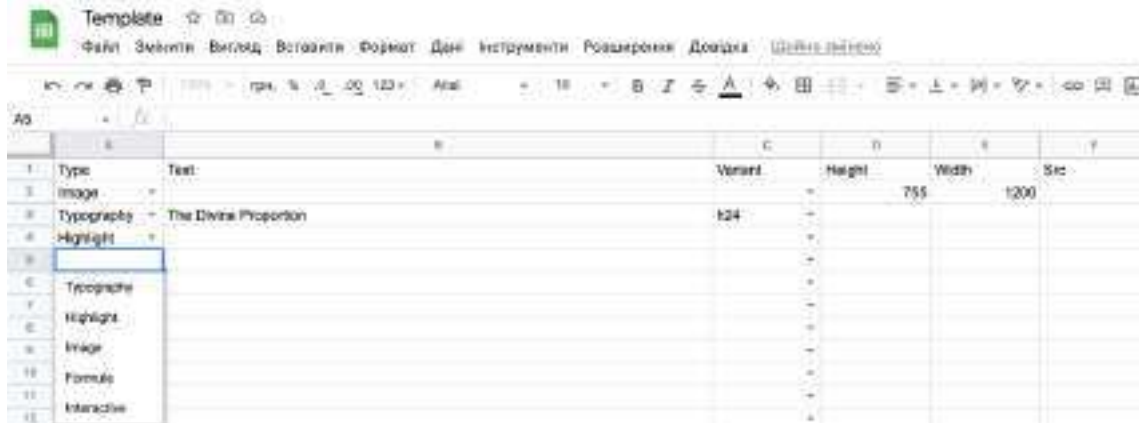


Рисунок 2. Тестування хмарної функції через Postman

Також на кожному етапі додатково перевірялись логи роботи функції у Google Cloud, щоб вкотре пересвідчитись, що все працює правильно.

Крім наведених особливостей розробки та тестування, також додатково варто додати з яким підходом реалізовувались складніші задачі, наприклад, такі як створення нової статті, що може складатись з різних інтерактивних елементів.

Введення великої кількості інформації у тіло запиту може бути проблематичним як для користувача, так і для обробки програми. Тож було прийнято рішення про заповнення файлу у Google Spreadsheet (приклад шаблону для заповнення проілюстровано на рисунку 3), прикріплення заповненого файлу до запиту у Postman та його відправка на сервер, де буде проводитись подальша обробка інформації.



The image shows a Google Spreadsheet template with columns for 'Type', 'Text', 'Variant', 'Height', 'Width', and 'Src'. The 'Text' column contains the title 'The Divine Proportion'. The 'Height' column has the value '124', 'Width' has '755', and 'Src' has '1200'. A dropdown menu is open over the 'Type' column, showing options: Text, Image, Typographic, Highlight, Image, Formula, and Interactive.

Type	Text	Variant	Height	Width	Src
Image	The Divine Proportion	124	755	1200	
Typographic					
Highlight					
Image					
Formula					
Interactive					

Рисунок 3. Шаблон для заповнення статті у Google Spreadsheet

У подальшому заповнений файл прикріплюється до запиту у Postman та відбувається відправка на сервер, де буде проводитись подальша обробка інформації.

## 5. ВИСНОВКИ

Популярність мікросервісів останнім часом зростає, оскільки вони можуть вирішити багато поточних ІТ-завдань, таких як збільшення швидкості, масштабованість додатків і процеси швидкого тестування. Більше того, мікросервісна архітектура впевнено та поступово стає лідером у підході до розробки систем, замінюючи монолітні рішення, які є застарілими та менш гнучкими.

У ході дослідження вдалось проаналізувати системний підхід щодо проектування архітектури програмного забезпечення: існуючі патерни та ринок хмарних провайдерів. Окрім того, на основі дослідження вдалось побудувати платформу для взаємодії з великою кількістю контенту різного роду.

## ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Dataversity [Електронний ресурс] : A Brief History of Microservices / Keith D.Foote – Режим доступу : [dataversity.net/a-brief-history-of-microservices/](https://dataversity.net/a-brief-history-of-microservices/) (дата звернення 13.11.2022).
2. Refactoring Guru [Електронний ресурс] : What's a design pattern? – Режим доступу : <https://refactoring.guru/design-patterns/what-is-pattern> (дата звернення 13.11.2022).
3. RedHat [Електронний ресурс] : What are cloud services? – Режим доступу : <https://www.redhat.com/en/topics/cloud-computing/what-are-cloud-services> (дата звернення 13.11.2022).
4. Brocoders [Електронний ресурс] : AWS vs. Azure vs. GCP. Comparing the TOP 3 Cloud Providers – Режим доступу : <https://brocoders.com/blog/aws-vs-azure-vs-gcp/> (дата звернення 13.11.2022).

# МОДЕЛЮВАННЯ ЗАЛЕЖНОСТІ ФАКТОРІВ РИЗИКУ ЗА ДОПОМОГОЮ КОПУЛ

Шепель І.О.<sup>1</sup>, Бідюк П.І.<sup>2</sup>

Національний технічний університет України  
«Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна

<sup>1</sup>irene.shepel@gmail.com

**В останні роки управління ризиками стало одним із найважливіших напрямів досліджень у фінансовій та економічній сферах. Одним з основних інтересів в цих сферах є моделювання залежностей факторів ризику, що дозволяють зробити апарат спеціальних функцій – копул і статистичний аналіз. Копула об'єднує одновимірні маргінальні розподіли разом, щоб сформувати багатовимірний розподіл, і у результаті виходить спільна функція розподілу стандартних рівномірних або нормальних випадкових величин. Метою роботи є аналіз методів побудови копул та моделювання функції спільного розподілу. Результатом дослідження є змодельовані багатовимірні розподіли та оцінки міри ризику.**

**Ключові слова:** фінансові ризики, копули, аналіз ризиків, залежність, VaR.

## 1. ВСТУП

Останнім часом дослідження, присвячені моделюванню внутрішньої залежності системи показників, використовують апарат спеціальних функцій – копул. Фундаментальна ідея копул полягає у поданні спільного розподілу множини показників у вигляді двох компонентів: перша відповідає за поведінку цих показників, а друга визначає характер залежності між ними.

Довгий час статистичне моделювання в сфері фінансів та економіки здебільшого ґрунтувалося на спрощених припущеннях. Нормальний розподіл домінував у дослідженнях багатовимірних розподілів: його часто припускали, але рідко заперечували [1]. Багатовимірні нормальні розподіли привабливі через їхню легку математичну обробку та керованість. У цьому випадку зв'язок між двома випадковими наслідками можна повністю описати, знаючи лише маргінальні розподіли та один додатковий параметр — коефіцієнт кореляції [2]. З роками статистики почали розуміти необхідність вивчення альтернатив нормальному розподілу. Істотною перевагою копул порівняно з іншими методами, що використовуються в моделюванні та аналізі, є його гнучкість – можливість побудови моделей різноманітних нелінійних структур залежності між досліджуваними рядами даних [3].

## 2. КОПУЛИ ТА МЕТОДИ ОЦІНЮВАННЯ ЇХ ПАРАМЕТРІВ

У теорії ймовірностей відправною точкою є поняття ймовірнісного простору  $(\Omega, \mathfrak{F}, Pr)$ , де  $\Omega$  – область визначення відповідних випадкових змінних;  $\mathfrak{F}$  – сігма-алгебра, тобто сімейство випадкових подій, які є підмножинами  $\Omega$ ;  $Pr$  – ймовірнісна міра. При цьому сімейство підмножин  $\mathfrak{F}$  в області визначення  $\Omega$  називають сігма-алгеброю, якщо воно

задовольняє таким властивостям: (1)  $\Omega \in \mathfrak{F}$ ; (2)  $A \in \mathfrak{F} \Rightarrow A \in \mathfrak{F}$ ;  $A_1, A_2, A_3, \dots \in \mathfrak{F} \Rightarrow \bigcap_{i \geq 1} A_i \in \mathfrak{F}$ .

Надалі під ризиком будемо розуміти випадкову подію, яка може призвести до фінансових втрат. Формальним означенням ризику може бути таке: ризик  $X$  – це випадкова подія, яка характеризується двома невід’ємними величинами – рівнем випадкових виплат або втрат та ймовірністю цієї події. Спільний розподіл ризиків розглянемо у вигляді:

$$H(x_1, \dots, x_n) = P(X_1 \leq x_1, \dots, X_n \leq x_n) = C(F_1(x_1), \dots, F_n(x_n)),$$

де  $F_1, \dots, F_n$  – маргінальні функції розподілу окремих ризиків;  $C$  –  $n$ -копула, що характеризує структуру залежності між ризиками.

**Теорема 1** (Скляра) [5]: Нехай  $H$  –  $n$ -вимірна функція спільного розподілу з маргінальними розподілами  $F_1, \dots, F_n$ . Тоді існує така  $n$ -копула  $C$ , що

$$H(x_1, \dots, x_n) = C(F_1(x_1), \dots, F_n(x_n)) \text{ для всіх } x \in R^n.$$

Якщо  $F_1, \dots, F_n$ , є неперервними, тоді  $C$  унікальна; в іншому випадку функції  $C$  є унікально визначеними на  $\text{Rng } F_1 \times \dots \times \text{Rng } F_n$ . І навпаки, якщо  $F_1, \dots, F_n$  є розподілами, а  $C$  є  $n$ -копулою, тоді  $H(x_1, \dots, x_n)$  –  $n$ -вимірна функція спільного розподілу з маргінальними розподілами  $F_1, \dots, F_n$ .

**Побудова сімейства копул.** Теорема Скляра гарантує існування копули та її унікальність за певних умов, але не надає методу її знаходження. Розглянемо методи побудови сімейств копул.

### 1. Метод оберненої функції

Ідея цього методу полягає в тому, що з теореми Скляра для спільної функції розподілу  $H$  та неперервних маргінальних розподілів  $F_1, \dots, F_n$  копула  $C$  визначається так:

$$C(u_1, \dots, u_n) = H(F_1^{(-1)}(u_1), \dots, F_n^{(-1)}(u_n)),$$

де  $F_i^{(-1)}(u_i)$  – обернена функція до функції маргінального розподілу  $F_i$ .

Найбільш поширеними у моделюванні фінансових випадкових величин є еліптичні розподіли і зокрема багатовимірний нормальний розподіл. При застосуванні до нього зворотного методу отримуємо багатовимірну нормальну копулу або як її ще називають – Гаусову копулу.

**Означення 1:** Нехай  $\rho$  – симетрична, додатно визначена матриця з діагоналлю 1. Гаусовою багатовимірною копулою називають функцію:

$$C(F_1, F_2, \dots, F_n, \rho) = \Phi_\rho(\Phi^{-1}(F_1), \Phi^{-1}(F_2), \dots, \Phi^{-1}(F_n)),$$

де  $\Phi$  – функція стандартного одновимірного нормального розподілу;  $\Phi_\rho$  – функція багатовимірного стандартного нормального розподілу з кореляційною матрицею  $\rho$ .

Щільністю багатовимірної Гаусової копули є:

$$c(F_1, F_2, \dots, F_n, \rho) = \frac{1}{\rho} \exp \left\{ -\frac{1}{2} \Phi^{-1}(F_1), \dots, \Phi^{-1}(F_n) \times \rho^{-1} - I \times \begin{matrix} \Phi^{-1}(F_1) \\ \vdots \\ \Phi^{-1}(F_n) \end{matrix} \right\}.$$

Тобто Гаусова копула повністю визначається кореляційною матрицею  $\rho$ , а тому її параметри досить легко оцінити.

### 2. Архімедові копули

**Означення 2:** Нехай  $\phi$  – неперервна строго зростаюча функція з  $I$  в  $[0, \infty]$  такою що  $\phi(1) = 0$ . Псевдо-інверсією для  $\phi$  називається така функція  $\phi^{[-1]}$  з  $\text{Dom } \phi = [0, \infty]$  в  $I$ , що

$$\phi^{[-1]}(t) = \begin{cases} \phi^{-1}(t), & 0 \leq t \leq \phi(0), \\ 0, & \phi(0) \leq t \leq \infty. \end{cases}$$

Відзначимо, що  $\phi^{[-1]}$  є неперервною та неспадаючою на  $[0, \infty]$ , та строго спадаючою на  $[0, \phi(0)]$ . Більш того,  $\phi^{[-1]}(\phi(u)) = u$  на  $I$ , та

$$\phi(\phi^{[-1]}(t)) = \begin{cases} t, & 0 \leq t \leq \phi(0), \\ \phi(0), & \phi(0) \leq t \leq \infty. \end{cases}$$

Якщо  $\phi(0) = \infty$ , то  $\phi^{[-1]} = \phi^{-1}$ .

**Лема 1:** Нехай  $\phi$  є неперервною строго спадаючою функцією з  $I$  в  $[0, \infty]$  такою, що  $\phi(1) = 0$ , та нехай  $\phi^{[-1]}$  буде псевдо-інверсією для  $\phi$  і функція  $C$  з  $I^2$  в  $I$  задається

$$C(u, v) = \phi^{[-1]}(\phi(u) + \phi(v)). \quad (1)$$

Тоді  $C$  задовольняє умовам обмеження для копули.

**Доведення:**  $C(u, 0) = \phi^{[-1]}(\phi(u) + \phi(0)) = 0$  та  $C(u, 1) = \phi^{[-1]}(\phi(u) + \phi(1)) = \phi^{[-1]}(\phi(u)) = u$ . Аналогічно для  $C(0, v) = 0$  та  $C(1, v) = v$ . Якщо  $\phi(0) = \infty$  тоді  $\phi$  називають строгим генератором. В цьому випадку  $\phi^{[-1]} = \phi^{-1}$  та

$$C(u, v) = \phi^{-1}(\phi(u) + \phi(v)) \quad (2)$$

називається строгою архімедовою копулою.

Всі копули, що можуть бути представлені у вигляді (1), називають архімедовими. Функція  $\phi$  називається генератором копули. Цей клас копул є одним з найбільш вживаних, тому що до нього належить значна кількість параметричних родин копул, що дає можливість відобразити значну різноманітність структур залежності. До того ж, побудова копул цього класу досить легка.

З (2) маємо:

$$C_{\theta}(u, v) = \phi^{-1}(\phi(u) + \phi(v)) = \exp\left(-\left[(-\ln u)^{\theta} + (-\ln v)^{\theta}\right]^{\frac{1}{\theta}}\right).$$

Сімейство копул називають копулами Гумбела у випадку двох змінних. До архімедових також належать сімейства копул Франка у випадку двох змінних

$$C(F_1, F_2) = -\frac{1}{\beta} \ln \left[ 1 + \frac{(e^{-\beta F_1} - 1)(e^{-\beta F_2} - 1)}{e^{-\beta} - 1} \right], \quad \beta \in \mathbb{R} \setminus \{0\};$$

### 3. РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Для оцінювання можливих ринкових втрат при виконанні валютних операцій часто застосовують відому модель VaR (Value-at-Risk). Ця модель використана для аналізу обмінних курсів HRK, PLN, CZK відносно EUR. Для оцінювання параметрів моделі взяті щоденні курси з 04.2003 по 01.2008, що дало вибірку розміром 1598 спостереження. Змодельовано функції спільного розподілу (Рис. 1–4) та оцінено параметри одновимірних маргінальних розподілів для кожного з курсів та параметри копул за методом максимальної правдоподібності (Табл. 1). Результати обчислювальних експериментів отримано за допомогою спеціалізованої системи підтримки прийняття рішень для моделювання і прогнозування фінансових процесів.

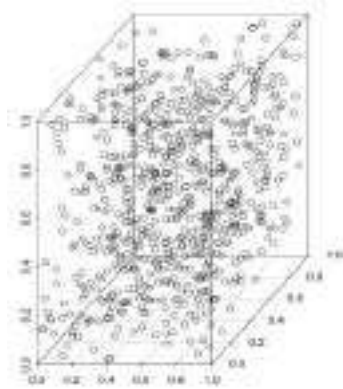


Рисунок 1. Спільний розподіл на основі копули Гумбела

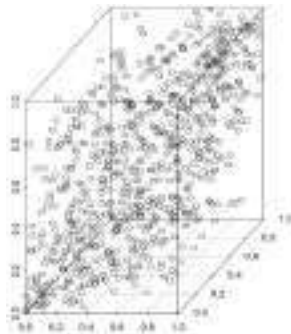


Рисунок 2. Спільний розподіл на основі нормальної копули

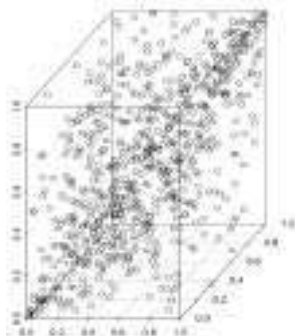


Рисунок 3. Спільний розподіл на основі копули Франка

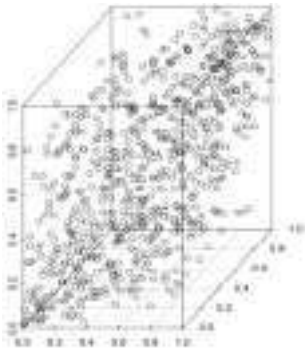


Рисунок 4. Емпіричний спільний розподіл курсів валют

Далі наведені результати оцінювання параметрів копул.

Таблиця 1. Результати оцінювання параметрів копул

Копула	Параметр	Значення	Середньоквадратична похибка
Гумбела	$\theta$	1.8736	0.0155
Нормальна	$\rho_1$	0.6617	0.0119
	$\rho_2$	0.3541	0.0138
	$\rho_3$	0.6038	0.0117
	$\rho_4$	0.8313	0.0055
	$\rho_5$	0.8672	0.0052
	$\rho_6$	0.8078	0.006
Франка	$\beta$	4.8546	0.0923

Емпірична оцінка міри ризику VaR для квантиля 0,05, тобто для 47 спостережень, що перевищують поріг, дорівнює 3,2412; для квантиля 0,01, відповідно для 15 спостережень, що перевищують поріг, оцінка міри ризику складає 3,7045. Наявних спостережень для квантиля 0,03 достатньо для отримання можливості використання цієї емпіричної оцінки на практиці, а для квантиля 0,01 вибірка є замалою.

#### 4. ВИСНОВКИ

Виконано аналіз можливості застосування класу спеціальних функцій – копул до опису багатовимірних розподілів у задачах менеджменту ризиків. Запропоновано метод побудови комбінованих маргінальних розподілів, який дозволяє враховувати важкі хвости одновимірних розподілів ризиків. Маргінальні розподіли поєднані у спільні розподіли ризиків через структуру залежності, що характеризується копулами.

На основі аналізу методів побудови сімейств копул запропоновано використовувати для моделювання ризиків кілька сімейств копул із корисними для менеджменту ризиків властивостями. В експерименті над тривимірними розподілами змін курсів валют продемонстрована можливість застосування запропонованого в роботі методу до моделювання багатовимірного розподілу через комбіновані маргінальні розподіли і структуру залежності між ними. Також наведено приклад застосування методу максимальної правдоподібності до оцінювання параметрів функції розподілу з їх подальшим використанням для оцінювання міри ризику.

#### ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Embrechts P., McNeil A. J., Straumann D. Correlation and Dependence in Risk Management: Properties and Pitfalls. Risk Management. 2002. С. 176–223.
2. Frees E. W., Valdez E. A. Understanding Relationships Using Copulas. North American Actuarial Journal. 1998. Т. 2, № 1. С. 1–25.
3. Bouchaud J.-P. Theory of financial risk and derivative pricing: from statistical physics to risk management. 2-ге вид. Cambridge : Cambridge University Press, 2003. 379 с..
4. Cherubini U., Luciano E., Vecchiato W. Copula Methods in Finance. Wiley & Sons, Incorporated, John, 2007. 310 с.
5. Extreme Value Theory and Applications, Vol. 3 // International conference NIST, 1993, 2-7 May. – Gaithersburg, 1993. – 230 с.

# СТОХАСТИЧНА МОДЕЛЬ В БІОЛОГІЇ ТА ВІРУСОЛОГІЇ

Яблунівський О.В.<sup>1</sup>, Мальцев А.Ю.<sup>2</sup>

Національний технічний університет України  
«Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна

<sup>1</sup>alexandryablunovskiy@gmail.com

**Віруси є досить великою та повсякденною загрозою, саме тому досить важливо слідкувати за ними та прогнозувати якомога точніше наслідки до яких вони можуть призвести. Саме тому для даної статті я обрав тему математики в вірусології оскільки вченні невпинно б'ються з даною проблемою та намагаються вирішити її якомога швидше. Результатом дослідження є модифікована модель SIRI, яка може описувати поведінку вірусів в наш час якомога краще.**

**Ключові слова: модель SIRI, стохастична модель в вірусології.**

## 1. ВСТУП

Наше сьогодні показує нам, що проблема епідемій та пандемій у світі є високою, як ніколи. Людство нещодавно пережило досить важкі часи та навіть зараз, ця проблема не зникла повністю. Науковці та лікарі невпинно б'ються з пандемією коронавірусу у світі, проте ми бачимо, що це не приносить повноцінного його зникнення. Тому людство повинно розуміти, що цю проблему треба вирішити як можна швидше.

Проблема широкомасштабних епідемій та пандемій походить з досить давнього часу. Людство стикається з цими проблемами кожне сторіччя і доля зводила нас з такими хворобами як чума, віспа, багато різних штамів вірусу грипу та досить багато всього іншого. Проте людство невпинно та досить швидко намагалось вирішувати ці проблеми і вдавалось це досить добре та ми робили досить вагомні внески в розвиток медицини. Проте науковці математики невпинно боролись та створювали математичні моделі розповсюдження вірусів, щоб мати змогу прогнозувати, як буде ширитись світом ця чи інша хвороба та що ми можемо зробити аби зменшити наслідки від неї для всього людства. Одна з досить вагомих та відомих стохастичних моделей є модель SIRI. Вона показує не тільки динаміку переходу населення з однієї категорії в іншу, аде і враховує зміни та фактори навколишнього середовища.

## 2. ОПИС ІНСУЮЧОЇ МОДЕЛІ

Отже, що з себе представляє модель SIRI. Вона побудована на принципі того, що з плином часу людство може переходити з однієї категорії в іншу, а саме:

1. S – це категорія сприйнятливих, тобто тих людей, які є в небезпеці та можуть бути інфікованими
2. I – це категорія інфікованих людей, тобто ті у яких уже почався перебіг хвороби та вони наражають на небезпеку людей з категорії сприйнятливих, оскільки вони можуть заражати їх.
3. R – це категорія, скажемо так, видалених, тобто це ті люди які вже не несуть загрозу іншим, тобто вони або одужали, або були ізольовані або померли від вірусу.

Дана модель, як і класична модель SIR фундаментально побудована на декількох принципово важливих припущеннях, які трохи згладжують наш світ, оскільки виразити всі аспекти в світі майже неможливо. Отже:

- Перше наше припущення, це те що ми вважаємо, що наше населення не є сталим та може змінюватись від загального коефіцієнту поповнення за рахунок народження чи імміграції, та те що він дорівнює коефіцієнту смертності;

- Друге припущення – це є той факт, що люди інфікуються з якоюсь швидкістю і вона не є сталою, а може змінюватись з часом та/або з урахуванням різних факторів, які можуть мати вплив на неї;
- Третє припущення – це те що люди переходять з категорії інфікованих в категорію видалених теж з якоюсь швидкістю, проте ця швидкість також не є сталою та може бути залежною від багатьох факторів, таких як час, навколишнє середовище та багато всього іншого;
- Четвертим є припущення, що люди з категорії видалених можуть і переходять в категорію інфікованих.

Отже перейдемо до самих рівнянь. Перше рівняння описує зміну кількості людей в категорії сприйнятливих до захворювання.

$$\frac{dS}{dt} = \mu - \mu S(t) - \frac{\beta S(t)I(t)}{1 + \alpha I(t)}$$

Де  $\mu$  - це загальний коефіцієнт поповнення за рахунок імміграції або народження він дорівнює природній смертності, він має бути додатнім та більше нуля;

$\frac{\beta I}{1 + \alpha I}$  – це швидкість передачі захворювання, вона є додатною;

$\alpha$  – це константа напівнасичення.

Друге рівняння описує зміну кількості людей, які були інфікованими та будується з декількох факторів, які є досить важливими при зміні стану, а саме зі швидкості передачі захворювання, коефіцієнта поповнення за рахунок народження та імміграції, швидкості з якою неінфекційні люди повертаються до заразного стану та коефіцієнту одужання. Отже, маємо що:

$$\frac{dI}{dt} = \frac{\beta S(t)I(t)}{1 + \alpha I(t)} - (\mu + e)I(t) + \frac{\gamma R(t)}{1 + kR(t)}$$

Де  $\frac{\beta I}{1 + \alpha I}$  – це швидкість передачі захворювання, вона є додатною;

$\mu$  - це загальний коефіцієнт поповнення за рахунок імміграції або народження він дорівнює природній смертності, він має бути додатнім та більше нуля;

$e$  – це коефіцієнт одужання;

$\frac{\gamma}{1 + kR(t)}$  – це швидкість з якою неінфіковані люди повертаються до заразного стану, та можуть наражати на небезпеку іншу частину населення;

$k$  – це константа напівнасичення.

Третє рівняння має показувати нам з якою швидкістю будемо мати поповнення даної категорії людей та швидкості з якою вони будуть вибувати з неї та переходити у стан, коли вони знову можуть наражати на небезпеку інших людей.

$$\frac{dR}{dt} = eI(t) - \mu R(t) - \frac{\gamma R(t)}{1 + kR(t)}$$

Де  $e$  – це коефіцієнт одужання;

$\mu$  - це загальний коефіцієнт поповнення за рахунок імміграції або народження він дорівнює природній смертності, він має бути додатнім та більше нуля;

$\frac{\gamma}{1 + kR(t)}$  – це швидкість з якою неінфіковані люди повертаються до заразного стану, та можуть наражати на небезпеку іншу частину населення;

Ці три рівняння в цілому створюють модель SIRI яка показує нам плинність епідемії чи пандемії з плином часу. Тепер поглибимось в деталі. Маємо що для того щоб порахувати кількість людей в населенні, то ми маємо таке рівняння, що в момент часу  $t$  населення буде складати:

$$N(t) = S(t) + R(t) + I(t)$$

Де  $S$  – це кількість сприйнятливих в момент часу  $t$ ;

$R$  – це кількість видалених в момент часу  $t$ ;

$I$  – це кількість інфікованих в момент часу  $t$ .

Також ми маємо коефіцієнт який буде показувати нам базову репродукційну кількість. Це означає, що при  $R_0 > 1$  епідемія відбудеться, якщо ж він буде менше ніж одиниця, то ми матимемо ситуацію коли епідемії не буде.

$$R_0 = \frac{\beta}{\mu + e + \frac{e\gamma}{\mu + \gamma}}$$

Де  $\beta$  - це швидкість передачі захворювання  
 $\mu$  - це загальний коефіцієнт поповнення за рахунок імміграції або народження він дорівнює природній смертності, він має бути додатним та більше нуля;  
 $e$  - це коефіцієнт одужання;  
 $\gamma$  - це швидкість з якою неінфіковані люди повертаються до заразного стану, та можуть наражати на небезпеку іншу частину населення;

Ми припускаємо, що коефіцієнти швидкості передачі захворювання та швидкості з якою неінфіковані люди повертаються до заразного стану, та можуть наражати на небезпеку іншу частину населення перебувають під впливом навколишнього середовища, а саме:

$$\beta \rightarrow \beta + \sigma_1 B_1(\dot{t}), \gamma \rightarrow \gamma + \sigma_2 B_2(\dot{t})$$

$\dot{B}_i(t) (i = 1, 2)$  - позначає білий шум, тобто ми маємо, що  $B_i(t)$  - це стандартний Броунівський рух, який є визначений на повному ймовірному просторі  $(\Omega, \mathcal{F}, P)$ .  $B_1(0) = 0$  та  $B_2(0) = 0$  - є стандартним відхиленням білого шуму та  $\sigma_i^2 > 0$  - представляє інтенсивність білих шумів. Отже в фіналі матимемо таку результуючу модель:

$$dS = \left[ \mu - \mu S(t) - \frac{\beta S(t)I(t)}{1 + \alpha I(t)} \right] dt - \frac{\sigma_1 S(t)I(t)}{1 + \alpha I(t)} dB_1(t)$$

$$dI = \left[ \frac{\beta S(t)I(t)}{1 + \alpha I(t)} - (\mu + e)I(t) + \frac{\gamma R(t)}{1 + kR(t)} \right] dt + \frac{\sigma_1 S(t)I(t)}{1 + \alpha I(t)} dB_1(t) + \frac{\sigma_2 R(t)}{1 + kR(t)} dB_2(t)$$

$$dR = \left[ eI(t) - \mu R(t) - \frac{\gamma R(t)}{1 + kR(t)} \right] dt - \frac{\sigma_2 R(t)}{1 + kR(t)} dB_2(t)$$

Коливання середовища є важливою частиною екосистеми в природному світі, включаючи температуру, імунологічний стан господаря, інкубаційні періоди тощо. Іноді невеликий шум може придушити вибухи в динаміці популяції. Теорія стохастичних диференціальних рівнянь (SDE) є кращим інструментом для представлення таких факторів впливу, оскільки SDE може забезпечити деякий додатковий ступінь реалізму порівняно з детермінованими системами. Багато вчених вводять ефекти стохастичних збурень у свої моделі як з біологічної, так і з математичної точки зору.

### 3. ЗАПРОПОНОВАНА ЗМІНА В МОДЕЛЬ

За основу мною була взята та сама модель SIRI, яка була описана вище. Нагадаю, що основа полягає в поділенні людей на три основні категорії між котрими вони можуть переміщуватись, отже категорії:

- S - сприйнятливі;
- I - інфіковані;
- R - видалені.

Перша зміна про яку я хочу поговорити - це умова про те що населення для котрого ми будемо дану модель можна поділити на чотири рівні категорії за віком. Ця зміна може бути зумовлені тим що, кожна категорія людей хворіє по різному та для кожної категорії населення ми матимемо свій статистичний коефіцієнт смертності. Отже, маємо:

- Перша категорія - це люди молодого віку, приблизно від нуля до двадцяти років, це молоді люди, які мають найнижчий коефіцієнт смертності та найлегший перебіг хвороби COVID-19;
- Друга категорія - це люди віком від двадцяти до сорока років, ми знаємо, що для даної категорії статистичний коефіцієнт смертності буде вищим за першу категорію та коефіцієнт швидкості перебігу хвороби буде трохи інший, оскільки люди цього віку хворіють трошки інакше;
- Третя категорія - це люди віком від сорока до шістдесяти років, вони матимуть ще вищий коефіцієнт смертності ніж друга категорія та будуть мати більший коефіцієнт швидкості перебігу хвороби, оскільки вони хворітимуть довше;

- Четверта категорія - це люди віком від шістдесяти років вони мають найвищий статистичний коефіцієнт смертності та найважчий перебіг хвороби, що являє собою найвищий коефіцієнт швидкості перебігу хвороби.

З усього вище сказаного, ми можемо сказати, що такий розподіл буде показувати стан речей стосовно хвороби найкраще, та допоможе покращити модель. Але, як відомо коефіцієнт смертності відрізняється не дуже сильно то ми будемо брати середній з усіх категорій.

Для даної зміни ми маємо додати ще декілька припущень про які, ми говорили раніше, а саме:

- Ми можемо поділити населення на чотири рівні категорії за віком;
- кількість інфікованих розподіляється на чотири категорії однаково, тобто мається на увазі, що якщо чотири інфікованих за один день, то до кожної з категорій перейде по одному інфікованому;

Перейдемо до математичних аспектів даної зміни. Оскільки ми маємо тепер чотири категорії населення, то для кожної групи населення будемо мати, що:

$$\frac{dS}{dt} = \mu - \mu S(t) - \frac{\beta}{1 + \alpha(I_1(t) + I_2(t) + I_3(t) + I_4(t))} (I_1(t) + I_2(t) + I_3(t) + I_4(t))(S_1(t) + S_2(t) + S_3(t) + S_4(t))$$

Де  $\mu$  - це загальний коефіцієнт поповнення за рахунок імміграції або народження він дорівнює природній смертності, він має бути додатнім та більше нуля;

$\frac{\beta I}{1 + \alpha I}$  - це швидкість передачі захворювання, вона є додатною;

$\alpha$  - це константа напівнасичення.

Рівняння для інфікованих 1 група:

$$\frac{dI}{dt} = 0,25 * \frac{\beta(S_1(t) + S_2(t) + S_3(t) + S_4(t))(I_1(t) + I_2(t) + I_3(t) + I_4(t))}{1 + \alpha(I_1(t) + I_2(t) + I_3(t) + I_4(t))} - (\mu + e_1)I(t) + \frac{\gamma_1(R_1(t) + R_2(t) + R_3(t) + R_4(t))}{1 + k(R_1(t) + R_2(t) + R_3(t) + R_4(t))}$$

Де  $\frac{\beta I}{1 + \alpha I}$  - це швидкість передачі захворювання, вона є додатною;

$\mu$  - це загальний коефіцієнт поповнення за рахунок імміграції або народження він дорівнює природній смертності, він має бути додатнім та більше нуля;

$e_1$  - це коефіцієнт одужання;

$\frac{\gamma}{1 + kR(t)}$  - це швидкість з якою неінфіковані люди повертаються до заразного стану, та можуть наражати на небезпеку іншу частину населення;

$k$  - це константа напівнасичення.

Для усіх інших категорій таке саме рівняння, проте з іншими коефіцієнтами одужання, швидкості з якою неінфіковані люди повертаються до заразного стану, та можуть наражати на небезпеку іншу частину населення

Рівняння для категорії видалених 1 категорія населення:

$$\frac{dR}{dt} = e_1(I_1(t) + I_2(t) + I_3(t) + I_4(t)) - \mu(R_1(t) + R_2(t) + R_3(t) + R_4(t)) - \frac{\gamma_1(R_1(t) + R_2(t) + R_3(t) + R_4(t))}{1 + k(R_1(t) + R_2(t) + R_3(t) + R_4(t))}$$

Де  $e$  - це коефіцієнт одужання;

$\mu$  - це загальний коефіцієнт поповнення за рахунок імміграції або народження він дорівнює природній смертності, він має бути додатнім та більше нуля;

$\frac{\gamma}{1 + kR(t)}$  - це швидкість з якою неінфіковані люди повертаються до заразного стану, та можуть наражати на небезпеку іншу частину населення.

## 4. ВИСНОВКИ

В даній частині дипломної роботи, мною було надане біологічне та математичне обґрунтування зміни, яку я вводив на основі базової моделі. За базову модель мною було вирішено взяти модель SIRS.

Я вважаю, що в наш час ця модель буде видавати набагато реальнішу картину світу, бо в класичній моделі немає різниці від віку. В новій моделі – це все враховано, тому я вважаю, що в наш час, потрібна саме ця модель.

### ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Anderson, R., May, R.: Population biology of infectious diseases: Part I. *Nature* 280, 361–367 (1979)
2. Ruan, S., Wang, W.: Dynamical behavior of an epidemic model with a nonlinear incidence rate. *J. Differ. Equ.* 188, 135–163 (2003)
3. Meng, X., Chen, L., Wu, B.: A delay SIR epidemic model with pulse vaccination and incubation times. *Nonlinear Anal. Real.* 11(1), 88–98 (2010)
4. Driessche, P.V.D., Zou, X.: Modeling relapse in infectious diseases. *Math. Biosci.* 207, 89–103 (2007)
5. CAMBRIDGE STUDIES IN MATHEMATICAL BIOLOGY: 15 Editors C. CANNINGS University of Sheffield, UK F. C. HOPPENSTEADT Arizona State University, Tempe, USA L. A. SEGEL Weizmann Institute of Science, Rehovot, Israel
6. Fatini, M., Lahrouz, A., Pettersson, R., Settati, A., Taki, R.: Stochastic stability and instability of an epidemic model with relapse. *Appl. Math. Comput.* 316, 326–341 (2018)
7. Capasso, V., Serio, G.: A generalization of the Kermack–McKendrick deterministic epidemic model. *Math. Bios.* 42, 41–61 (1978)

# МЕТОДИ АНАЛІЗУ ТА ПРОГНОЗУВАННЯ ФІНАНСОВОГО СТАНУ КОРПОРАЦІЇ НА ПРИКЛАДІ КОМПАНІЇ “PHILIP MORRIS INTERNATIONAL”

Ярмола А.О.<sup>1</sup>, Зайченко О.Ю.<sup>2</sup>

Національний технічний університет України  
«Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна

<sup>1</sup> anna16yarmola@gmail.com,

<sup>2</sup> zaichenko.helen@lil.kpi.ua [0000-0001-9662-3269]

Одним з найважливіших етапів в управлінні підприємством, незважаючи на розміри компанії та виду її комерційної діяльності є аналіз фінансового стану. Правильний підхід до вирішення даного завдання надає змогу прийняти правильні рішення в управлінні підприємницькою діяльністю, виявити сильні та слабкі сторони в роботі, а також прийняти оптимізаційні рішення для досягнення кращих результатів та оцінити перспективи майбутнього розвитку. За допомогою аналізу, а також і прогнозування майбутнього фінансового стану, компанія отримує наглядну картину для розуміння фінансової ситуації та прийняти обґрунтоване та документально підтвержену інформацію, щодо ефективності та стабільності її функціонування.

**Ключові слова:** аналіз фінансового стану, прогнозування, ліквідність, конкурентоспроможність, платоспроможність, Philip Morris International

## 1. ВСТУП

Одним з найпоширеніших фактів на сьогодні є те, що вся економіка у світу існує та розвивається завдяки підприємствам та корпораціям, в основі яких і лежить економічна та фінансова діяльність. Саме завдяки компаніям, які в свою чергу не тільки перетворюють процеси та ресурси на прибуток, а й задовольняють потреби суспільства. Поняття «фінансовий стан» не має одного визначення, багато науковців трактують це поняття по різному, але найбільш точним визначенням, що:

«Фінансовий стан- це складова аналітичного економічного дослідження щодо вивчення рівня, змін і динаміки фінансових показників у їх взаємозв'язку і взаємозумовленості, з метою прийняття ефективних управлінських рішень для забезпечення внутрішньої і зовнішньої діяльності господарюючих систем, сталого економічного розвитку і стійкого фінансового стану.»

При аналізі фінансового стану найчастіше розглядають такі елементи, як: ліквідність, рентабельність, платоспроможність та стабільність.

## 2. МЕТОД АНАЛІЗУ ТА ПРОГНОЗУВАННЯ ФІНАНСОВОГО СТАНУ

Існує безліч методів аналізу фінансового стану, серед яких можна виокремити такі, як: горизонтальний (часовий) аналіз, вертикальний, трендовий аналіз, аналіз відносних показників, порівняльний та факторний аналіз. Також в першу чергу варто зазначити, що кожен з представлених методів базується на джерелах інформації про діяльність компанії: Баланс, Звіт про фінансові результати, Звіт про рух грошових коштів та Звіт про власний

капітал. Основу для якісного проведення аналізу фінансового стану є забезпеченість та повнота інформації, необхідної для дослідження.

Існують різні формули — значення, різні коефіцієнти — які можна використовувати відповідно до того, який фінансовий звіт ви аналізуєте. Щоб здійснити базовий аналіз балансу буде достатньо розрахунок лиш є таких показників:

- Коефіцієнт поточної ліквідності - вимірює вашу ліквідність, наскільки легко ваші поточні активи можна конвертувати в готівку, щоб покрити ваші короткострокові зобов'язання. Чим вищий коефіцієнт, тим більш ліквідні ваші активи.

- Коефіцієнт швидкої ліквідності - він вимірює, наскільки добре ваш бізнес може погасити свої борги

- Коефіцієнт співвідношення боргу до власного капіталу - показує, наскільки ваш бізнес залежить від власного капіталу порівняно з позиченими коштами.

Фінансова стійкість формується з ефективного формування, використання та використання фінансових ресурсів, а як результатом правильного використання впливає і платоспроможність компанії.

Коефіцієнт платоспроможності та ліквідності – є двома самостійними показниками, але пов'язані між собою, значення яких в свою чергу фінансовий стан підприємства. Поняття платоспроможності показує здатність компанії виплачувати свої зобов'язання.

По іншому можна описати платоспроможність, як стан грошовим забезпечення виробничої та інвестиційної діяльності.

А поняття ліквідності показує можливість будь-якого активу перетворитися на гроші, також є засобом забезпечити підприємство платоспроможністю.

Але також в компанії можуть виникати ситуації, коли воно є ліквідним, але неплатоспроможним: це наявність майна, якого достатньо для виплати всіх зобов'язань, але не вистачає для покриття поточних боргів, коли компанія по якійсь причині не може продати майно.

Так и навпаки, є платоспроможним, але не є ліквідним: має в наявності засоби для виплат поточних боргів, але недостатньо для всіх зобов'язань.

Для визначення ділової активності компанії проводиться на основі коефіцієнтів оборотності.

Абсолютні показники ділової активності підприємства характеризують співвідношення "витрати – прибуток", що є основою оцінювання ефективності діяльності. Витрати подані вкладеним в активи капіталом, прибуток становить різницю між виручкою від реалізації і витратами.

При підвищенні ділової активності компанії прибуток має збільшуватися більшим темпом від виручки реалізації та активи. Навіть такі випадки можуть виявлятися і успішних корпорація, що це відношення порушується. Компанія буде мати достатній рівень ділової активності, нарощуючи кількість об'єктів діяльності, реалізує весь об'єм покупцям та отримує оплату, яка вказана в умовах договору.

Важливість прогнозування фінансового стану підприємств в сучасних умовах полягає у необхідності дослідження впливу різних ризиків на їх функціонування та здійснення при потребі відповідних заходів щодо їх мінімізації з метою запобігання виникненню кризових процесів на підприємстві, погіршенню ліквідності чи фінансової стійкості

Прогнозування фінансового стану підприємства здійснюється у декілька етапів, які включають [1]:

1. Визначення об'єктів прогнозу, мети і задач.

2. Відбір об'єктів, які прогнозуються, встановлення ієрархічності, взаємозв'язків між ними.

3. Визначення часових горизонтів прогнозу.
4. Формування інформаційної бази даних для прогнозування.
5. Вибір методів прогнозування, їх обґрунтування, оцінювання точності прогнозу.
6. Складання прогнозу.
7. Відстеження результатів

### 3. РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

#### 3.1 Аналіз фінансового стану

Для дослідження фінансового стану та прогнозування було обрано американську компанію «Philip Morris International». Вона є однією з найбільших виробників тютюнових виробів у світі. Свою продукцію компанії вдалося реалізувати в близько 180 країнах.

Мета компанії «Philip Morris International»: «Наша мета – замінити сигарети на бездимні продукти в інтересах повнолітніх споживачів, які інакше продовжували б курити, суспільства, компанії та своїх акціонерів.» [2]

Для аналізу було зібрану повну інформацію у вигляді фінансових звітів компанії за період 2019-2021 роки. Після перевірки повноти представлених звітів перейдемо до оцінки фінансового стану. Зі звітності було досліджено основні показники для визначення та оцінки фінансового стану на теперешній час. Після проведення дослідження можна зробити ряд висновків, які характеризують активи, капітал та прибуток компанії. Завдяки експортному потенціалу компанії, у 2021 році чистий дохід від реалізації продукції склав близько 30 мільярдів гривень, також у 2021 році РМІ демонструє стабільно високі показники фінансової діяльності. Прибуток компанії становить 3.3 мільярда гривень у 2021 році. Активи компанії на кінець 2021 року склали 11 мільярда гривень, в тому числі оборотні активи – 10.6 мільярда гривень – та необоротні – 0.4 мільярда гривень. Пасиви на кінець 2021 року переважно сформовані із поточних зобов'язань в сумі 9.3 мільярда гривень і власного капіталу – 1.5 мільярда гривень. Для кращого розуміння є необхідність продемонструвати динаміку змін фінансово-економічних показників у вигляді графіків.

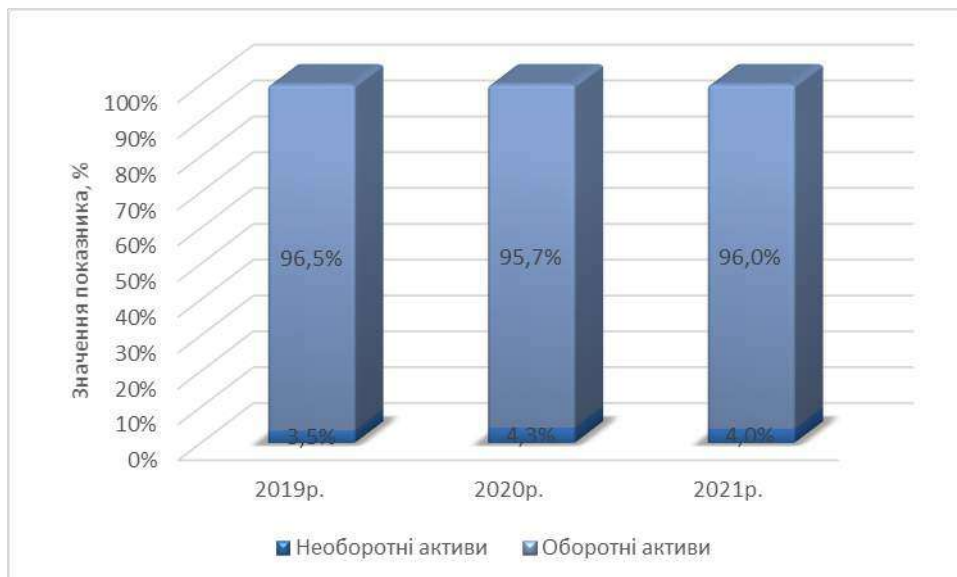


Рисунок 1. Динаміка активів РМІ у 2019-2021 рр.

В результаті бачимо стабільне співвідношення суми активів, а також у компанії значно більше переважають оборотні активи над необоротними, що каже про неоптимальне їх співвідношення.



Рисунок 2. Динаміка структури капіталу РМІ за 2019-2021 рр.

Мета компанії при управлінні капіталом полягає у забезпеченні подальшої роботи як безперервно функціонуючого підприємства, щоб приносити прибуток учасникам товариства та вигоди іншим зацікавленим сторонам, а також підтримувати оптимальну структуру капіталу. Сума капіталу, управління яким компанія здійснює станом на 31 грудня 2021 року, складає 1 155 550 тис. гривень. Компанія контролює величину капіталу на основі співвідношення власних та позикових коштів.

Також продемонструємо динаміку чистого доходу від реалізації та чистого прибутку РМІ у 2019-2021 рр.

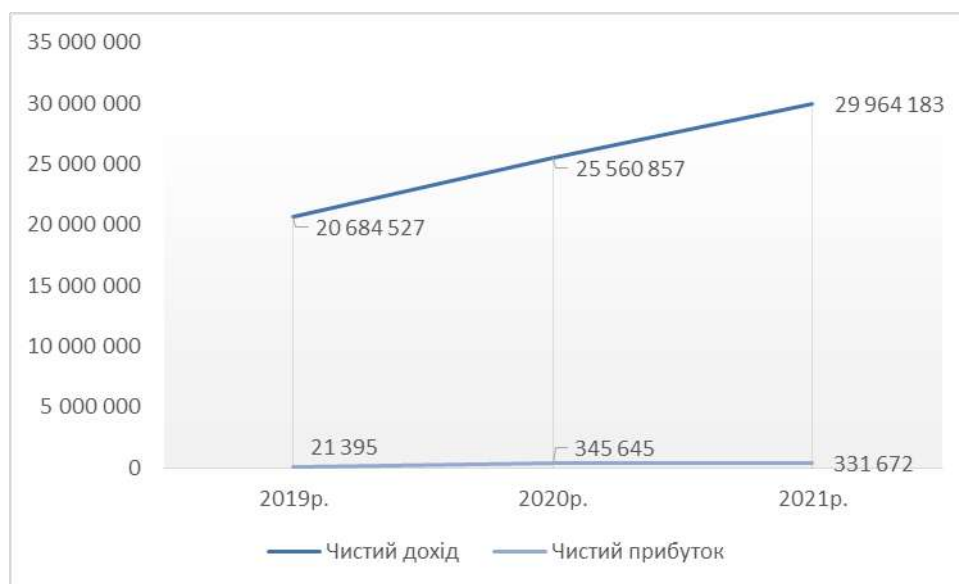


Рисунок 3. Динаміка чистого доходу від реалізації та чистого прибутку РМІ у 2019-2021 рр.

Для визначення ділової активності компанії проводиться на основі коефіцієнтів оборотності. Абсолютні показники ділової активності підприємства характеризують співвідношення "витрати – прибуток", що є основою оцінювання ефективності діяльності.

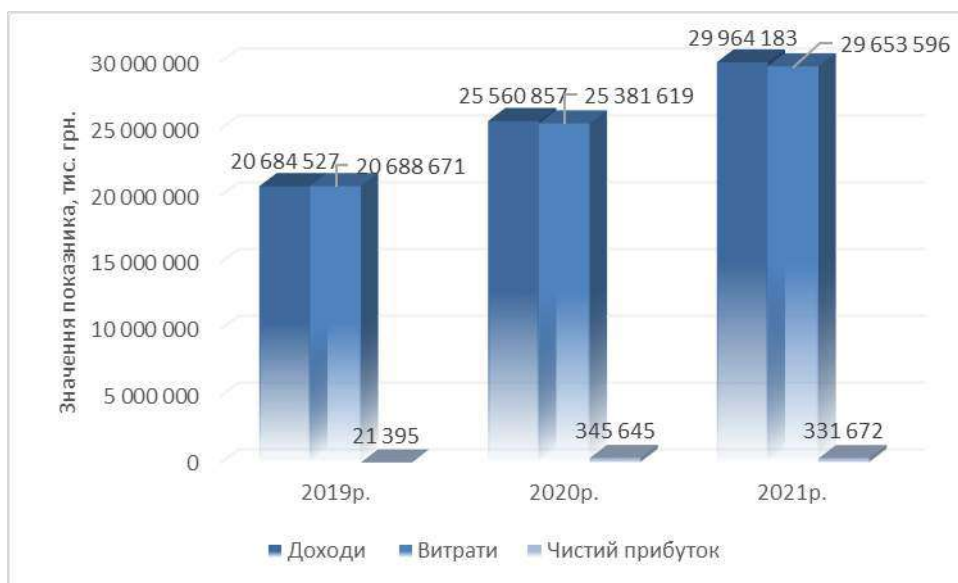


Рисунок 4. Динаміка обсягів доходів, витрат та чистого прибутку РМІ за 2019-2021 роки

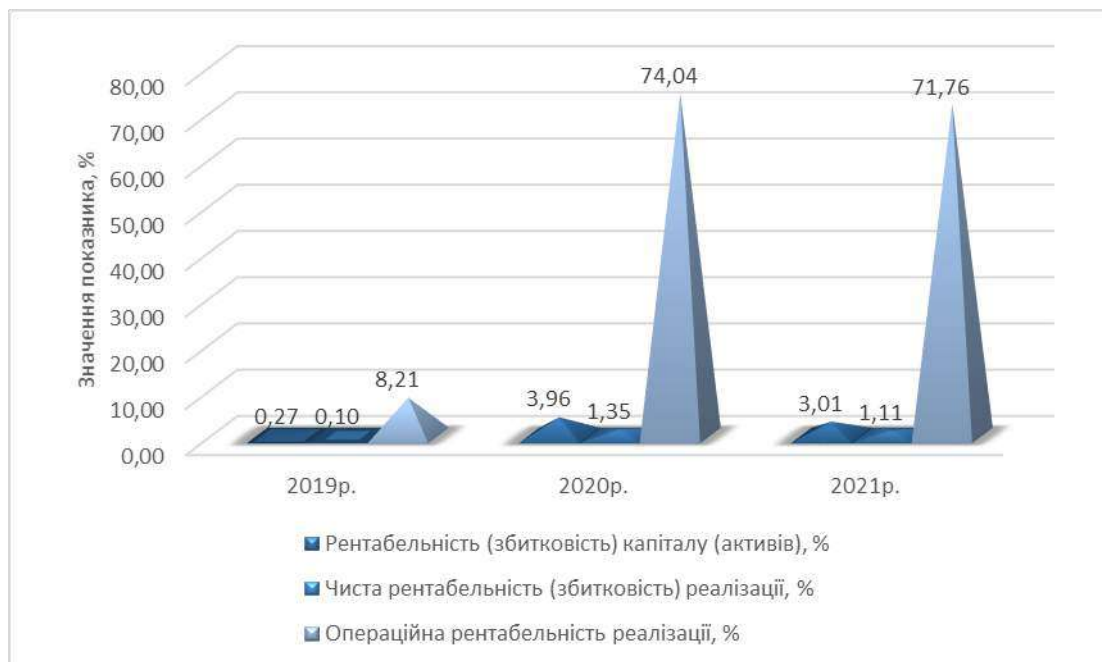


Рисунок 5. Динаміка показників рентабельності (збитків) капіталу, операційної та чистої реалізації продукції РМІ за 2019-2021 рр.

Ми змогли побачити справедливу величину прибутку компанії порівняно з сумою її витрат, капіталу та ресурсів. За результатом дослідження компанія має високий рівень платоспроможності, оскільки коефіцієнти абсолютної, швидкої та загальної ліквідності перевищують нормативні значення.

Також побудуємо динаміку коефіцієнтів термінової, мобільності активів та загальної ліквідності.

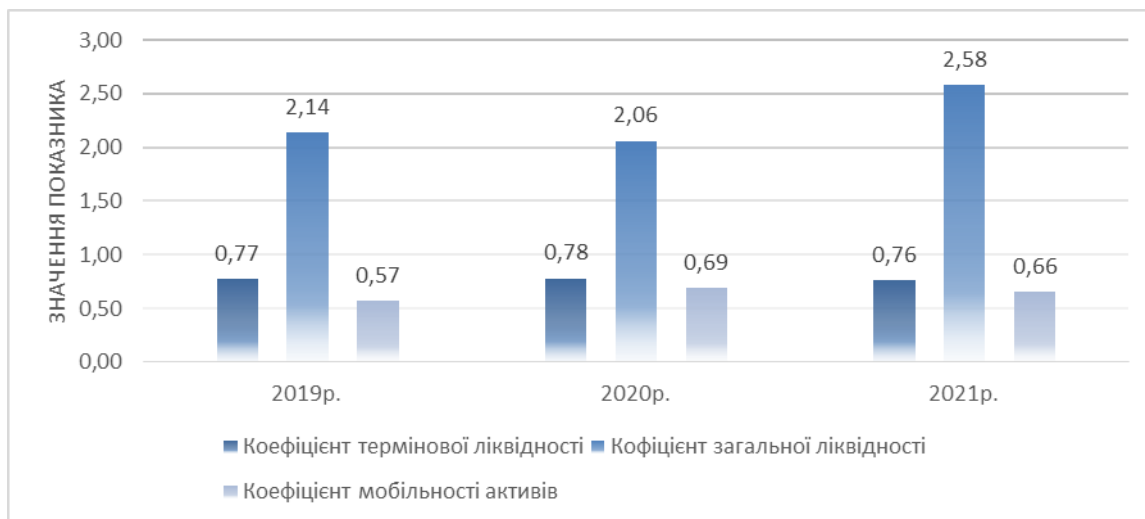


Рисунок 6. Динаміка коефіцієнтів ліквідності РМІ за 2019-2021 рр.

Отже, розглянувши деякі фінансові показники можна зробити ряд висновків, щодо фінансового стану компанії РМІ:

1. Компанія протягом року демонструє стабільний стан ліквідності. Коефіцієнт загальної ліквідності станом на кінець 2021 року становить 2.8.

2. Активи на кінець 2021 року склали 11.5 мільярда гривень, в тому числі оборотні активи – 9.9 мільярда гривень, необоротні активи – 1.6 мільярда гривень.

3. Пасиви сформовані із поточних зобов'язань в сумі 3.5 мільярдів гривень і власного капіталу – 7.9 мільярда гривень.

4. Внаслідок використання компанією фінансових інструментів компанія наражається на наступні ризики:

- Кредитний ризик

Станом на кінець 2021 року у компанії був один дебітор, розмір заборгованості якого складала приблизно 75%-85% всієї дебіторської заборгованості.

- Ринковий ризик

Кожного року компанія поновлює свою фінансову політику, яка встановлює ліміти для зберігання коштів на поточних, депозитних та кредитних рахунках компанією окремо по кожному банку, з якими співпрацює.

- Валютний ризик

Ризик розраховується лише для грошових залишків у валютах, інших ніж функціональна валюта компанії. Валютний ризик на кінець 2021 року не відображає типовий рівень ризику протягом року.

- Ризик ліквідності

Для того, щоб звести до мінімуму ризик ліквідності компанія підтримує стабільну базу фінансування, заключаючи кредитні угоди з міжнародними банками.

5. Також основним ризиком для діяльності компанії є нестабільність та непередбачуваність змін у податковому законодавстві.

### 3.2 Прогнозування фінансового стану підприємства

Для нашої задачі в прогнозуванні фінансового стану підприємства доцільно використовувати метод прогнозування екстраполяції тренду, він базується на зміні тенденцій характеристик підприємства. Та є одним з найпоширеніших методів прогнозування динаміки. Моє рішення було прийнято у виборі цього методу шляхом проведеного

дослідження та було зроблено висновок, що компанія має стабільний результат ліквідності та платоспроможності, та має прогресивний ріст її основних запасів та прибутку, що в свою чергу свідчить про постійний розвиток компанії, залучення більшого об'єму фінансування та розширення асортименту свого продукту.

Мета даного методу полягає в тому щоб показати до яких результатів можна дійти в майбутньому, якщо діяльність компанії буде розвиватися з таким же темпом та прискоренням, як в минулі періоди.

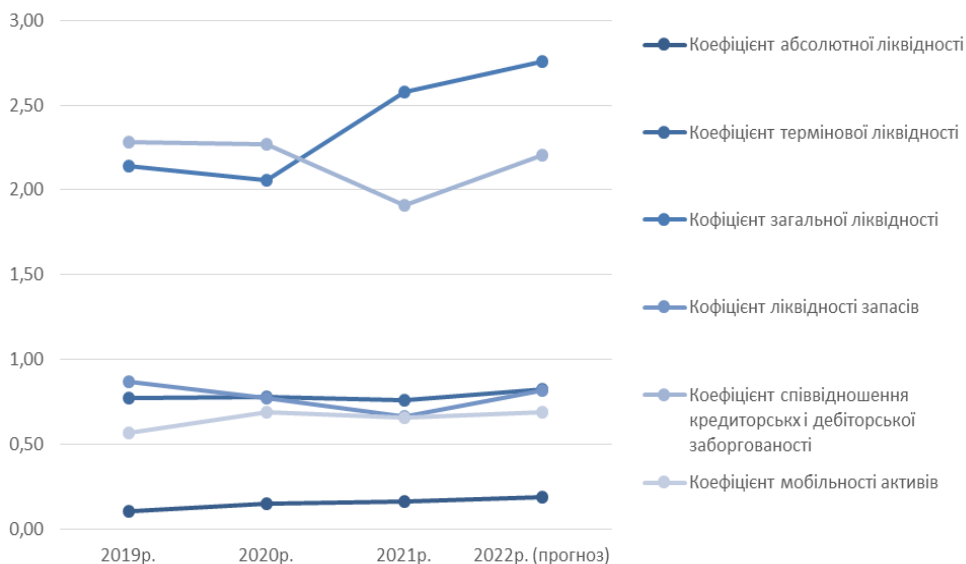


Рисунок 7. Динаміка зміни показників на прогностичний період

Отже, наведений прогноз демонструє, більшість показників мають стабільний темп росту, окрім коефіцієнта загальної ліквідності, на мою думку це є результатом росту коефіцієнту абсолютної ліквідності, таким чином в прогностичному періоді компанія буде мати нормальний фінансовий стан та в перспективі абсолютно ліквідним і платоспроможним підприємством.

## ВИСНОВКИ

Тож яка мета сутності фінансового аналізу – це є визначенням напрямків та обчислення резервів зростання ринкової вартості компанії, основується на дослідження стану і динаміці зміни показників, які охарактеризовують фінансову діяльність підприємства, а також факторів, які безпосередньо впливають на зміну цих показників.

І завдання вважається вирішеним, якщо після проведення фінансового аналізу були розроблені певні рекомендації та шляхи вирішення або покращення ситуації, задля розвитку компанії, які в свою чергу і призводять до досягнення поставленої цілі.

У ході дослідження було реалізовано ряд методів для аналізу фінансового стану та метод екстраполяції трендів для прогнозування ситуації на майбутній період, за допомогою якого ми змогли провести дослідження фінансово-економічної ситуації досить великої корпорації як “Philip Morris International”, було виявлено сильні та слабкі сторони, а також досліджено фінансовий стан. І як висновок нашого дослідження має, що компаніє має достатній рівень ліквідності, платоспроможності та конкуренто спроможності, що і дозволяє компанії конкурувати на ринку тютюнових виробів з такими ж великими корпораціями.

## ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Приходько Н.І. Поняття фінансового стану підприємства: деякі аспекти його визначення / Н. І. Приходько // ФП ФІП PSE. – 2010. – Т. 8. – С. 188–190
2. <https://www.pmi.com/>
3. Ковтуненко Ю.В. Методичні основи аналізу фінансового стану промислового підприємства / Ю.В. Ковтуненко // Економіка. Фінанси. Право. – 2016. – № 8/2. – С. 40-41.
4. <https://studfile.net/preview/7292831/page:21>
5. Комариця Л.Л. Аналіз фінансового стану підприємства у кризових умовах економіки / Л. Л. Комариця, Ю.Г. Кіцак – [Електронний ресурс] Режим доступу: [http://www.rusnauka.com/9\\_DN\\_2010/Economics/61392.doc.htm](http://www.rusnauka.com/9_DN_2010/Economics/61392.doc.htm).
6. Коробов М. Я. Фінансово-економічний аналіз підприємств / М.Я. Коробов. – К. : Знання, 2000. – 378 с.

# ПОРІВНЯЛЬНИЙ АНАЛІЗ МЕТОДІВ КОРОТКОСТРОКОВОГО ПРОГНОЗУВАННЯ

Ярошенко В.О.<sup>1</sup>, Бідюк П.І.<sup>2</sup>

Національний технічний університет України  
«Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна

<sup>1</sup> valiajaroshenko@gmail.com, <sup>2</sup> pbidyuke\_00@ukr.net

**Питання прогнозування супроводжує людство майже весь час його існування. Завжди було важливим спрогнозувати стан хворої людини, знати майбутню поведінку погоди та що є на сьогодні одним з ключових питань, отримати інформацію про майбутні економічні показники на основі поточних даних. Одним із найбільш популярних підходів є прогнозування на основі моделей, побудованих за експериментальними даними, а одним із результатів застосування якого є створення Системи підтримки прийняття рішень (СППР), яка дасть змогу суттєво полегшити аналіз відповідних ризиків. Метою роботи є порівняльний аналіз методів короткострокового прогнозування для отримання можливості подальшого розуміння фінансових ризиків для обраних економічних процесів. Результатом дослідження є СППР, що виконує короткостроковий прогноз. У роботі використано теоретичні та емпіричні методи дослідження.**

**Ключові слова:** короткостроковий прогноз, моделювання, нейронні мережі, фінансові ризики, економічні процеси.

## 1. ВСТУП

З великої кількості навколишніх процесів у суспільстві, фінансові та економічні є одними з найбільш поширених. До них відносять інвестиційні фінансові та процеси глобалізації і державної інтеграції у світову економіку, курси валют, акцій і облігацій, процеси виникнення міжнародних фінансових потоків, формування та використання бюджетів на державному рівні, фінансово-економічну діяльність підприємств, біржові процеси формування цін на активи різного характеру та ще багато інших. Однією із складностей, які виникають в моделюванні таких процесів, є те, що більшість таких процесів мають нелінійний нестационарний характер при будь-якому розвитку економіки, особливо в економіці перехідного періоду. Особливістю перехідного періоду в економіці є те, що в цей період еволюції суспільства відбувається те, що стара система сходить з історичної арени й водночас народжується і утверджується нова. У зв'язку з цим розвиток перехідної економіки має особливий характер, що істотно відрізняється від звичайного, нормального економічного розвитку. В цілому, однією з основних рис перехідної економіки є альтернативність шляхів подальшого розвитку та цілей, що досягаються при цьому.

Людина, як вид, протягом всього свого існування намагалася зробити прогнози та оцінити подальші наслідки різноманітних дій. Задовго до сьогодення людство уже мало попит на отримання достовірної інформації та з плином часу важливість прогнозів у житті людини не зменшилася, але їхні цілі та напрями набули кардинально нового вигляду. Для будь-якого об'єкта, будь то велике підприємство чи, навіть, держава є дуже важливо заздалегідь знати про всі можливі ризики для будь-якого розвитку подій та попередити їх. У

випадку, коли необхідно прийняти важливе рішення, потрібно добре зважити всі можливі наслідки від прийняття того чи іншого рішення та оцінити ризики для кожної можливої ситуації. Загалом, прогнозування відіграє дуже велику роль у роботі з фінансовими процесами. Воно допомагає краще розуміти ці процеси, їх взаємодію з іншими процесами та оцінювати майбутню поведінку.

Фінансово-економічне прогнозування – це процес оцінювання фінансово-економічних прогнозів, побудований на наукових методах пізнання соціально-економічних явищ з використанням усієї сукупності методів, засобів та способів прогностики. Загалом, основною метою прогнозування можна назвати отримання необхідної інформації для розуміння майбутніх ситуацій, передбачення та, за можливості, своєчасного реагування на незадовільний прогноз. Якісне прогнозування дозволяє виявляти тенденції розвитку ринку і здійснювати свою діяльність відповідно до даних тенденцій, займати лідируючу позицію на ринку і успішно розвиватися, тому держава і великі компанії та середній бізнес витрачають великі кошти на прогнозування процесів.

## **2. ОПИС МЕТОДІВ КОРОТКОСТРОКОВОГО ПРОГНОЗУВАННЯ**

За останні десятиліття в світі було створено велику кількість математичних методів, що дозволяють спрогнозувати лінійні, нелінійні нестационарні та інші процеси. Всі методи прогнозування, як правило, розділяють на три широкі класи:

1. Прогнозування, за основу ідею якого взято суб'єктивне судження, оцінка з урахуванням досвіту й інтуїції, поглиблені розуміння предметної області та інші міркування, які мають пряме, або опосередковане відношення до досліджуваного процесу.

2. Методи, що ґрунтуються на застосуванні часового ряду однієї змінної. Тут можливі такі варіанти, як авторегресія (АР), авторегресії з ковзним середнім (АРКС), АРКС з урахуванням тренду та інші подібні моделі.

3. Векторні процеси, або ж часові ряди декількох змінних, слугують як третій клас. Тут враховується залежність ендогенної змінної від деякого числа регресорів у правій частині.

Звичайно, що ці три класи не є взаємовиключними, тому цілком доцільне використання комбінацій двох, або навіть трьох приведених методів [1 – 3]. Одним із способів прогнозування є регресійні моделі, що передбачають оцінювання значення змінної  $Y$  на основі значень змінних  $X$ . Якісно проведений регресійний аналіз дає також оцінку того, наскільки припущена форма залежності відповідає даним спостереження, але тільки в межах діапазону значень наявних незалежних змінних. Це означає, що будь-яка екстраполяція значно залежить від припущень щодо структурної форми регресійної залежності. Не слід обирати залежність лінійності за змінними і лінійність за параметрами лише з міркувань зручності розрахунків, для побудови моделі слід залучати всі доступні знання. Якщо відомо, що залежні змінні не можуть вийти за межі певного діапазону значень, це може стати корисним при виборі моделі – навіть якщо в отриманій вибірці немає значень близьких до таких меж. В цілому, математичні моделі можна будувати у формі різницевого рівняння. Таких як, авторегресія, авторегресія з ковзним середнім тощо.

Розглянемо узагальнений алгоритм (Рис. 1) для побудови математичної моделі на основі статистичних даних. Алгоритм є узагальненим для застосування до систем чи процесів будь-якого типу.

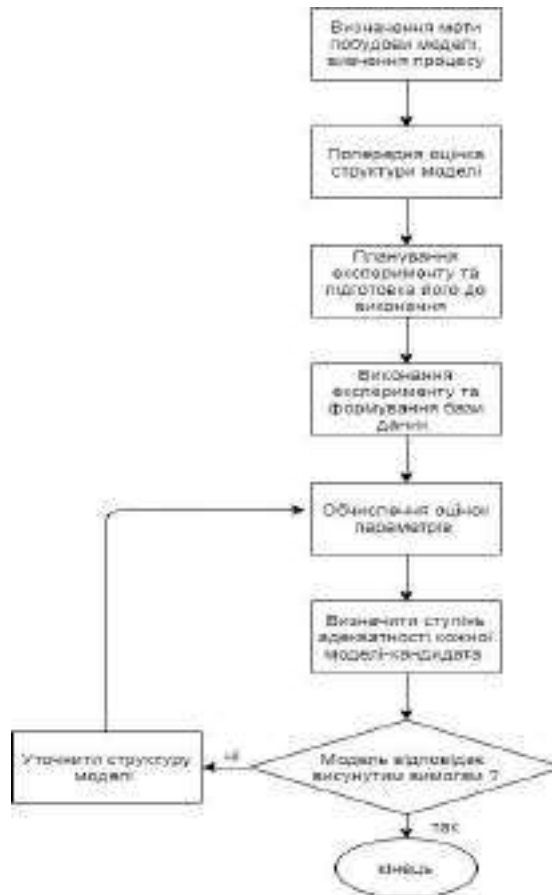


Рисунок 1. Алгоритм побудови математичної моделі на основі статистичних даних

Серед численної кількості методів моделювання, що існують в світі, можна виділити такі: аналітичне моделювання, математичне моделювання, імітаційне моделювання. Перевагою аналітичного методу моделювання є можливість отримання залежності в явному вигляді і застосування до неї методів класичного математичного аналізу. Якщо є можливість побудувати аналітичну модель системи, то завжди віддають перевагу цьому методу моделювання. Зауважимо також, що алгоритм відшукування точного розв'язку задачі може бути реалізований дослідником самостійно, за допомогою спеціального програмного забезпечення або за допомогою чисельних методів. Існують системи, опис яких не піддається опису аналітичними функціями, але процес функціонування їх може бути описаний алгоритмом імітації [4]. Під імітацією розуміють відтворення за допомогою комп'ютерної програми процесу функціонування складної системи в часі. У результаті багатократних прогонів імітаційної моделі дослідник отримує інформацію про властивості реальної системи. При виборі процесів для подальшого моделювання та прогнозування було обрано економічні процеси, оскільки більшість таких процесів є нелінійними та нестационарними. Для побудови прогнозів було обрано два методи: за допомогою нейронних мереж та системи моделювання EViews [5].

### 3. РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Обраними даними для моделювання та подальшого прогнозування є даними з фінансової біржі. Файл містить дані з 1 січня 2016 року по 6 травня 21 року. Ці дані можна використовувати для прогнозування цін на акції в майбутньому.

Розглянемо графік вхідних даних – Рис. 2.

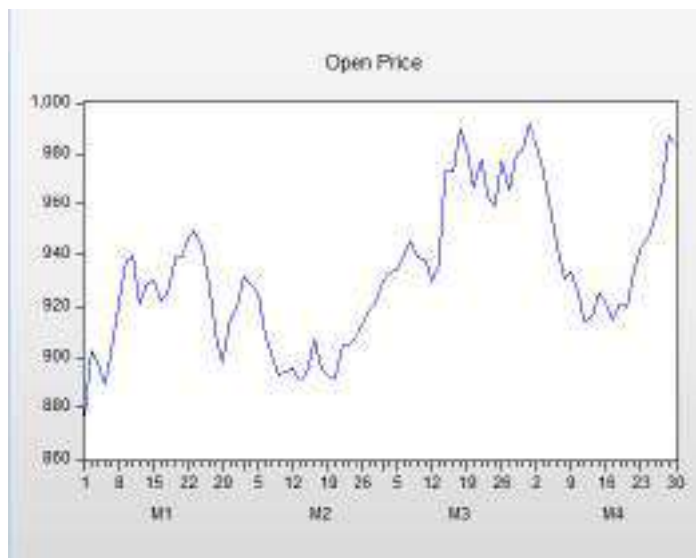


Рисунок 2. Графічне зображення даних

За допомогою системи EViews для вхідних даних були побудовані моделі другого, четвертого та сьомого порядку, але по параметрам вони не давали адекватних результатів, тому було прийняте рішення побудувати модель авторегресії. Після аналізу корелограми було прийнято рішення побудувати модель авторегресії 31-го порядку (АР (31)) та оцінивши залишки цієї моделі модель авторегресії з ковзним середнім 5-го порядку (АРКС (31, 5)). Порівняємо графік вхідних даних та вихідних даних (Рис. 3)

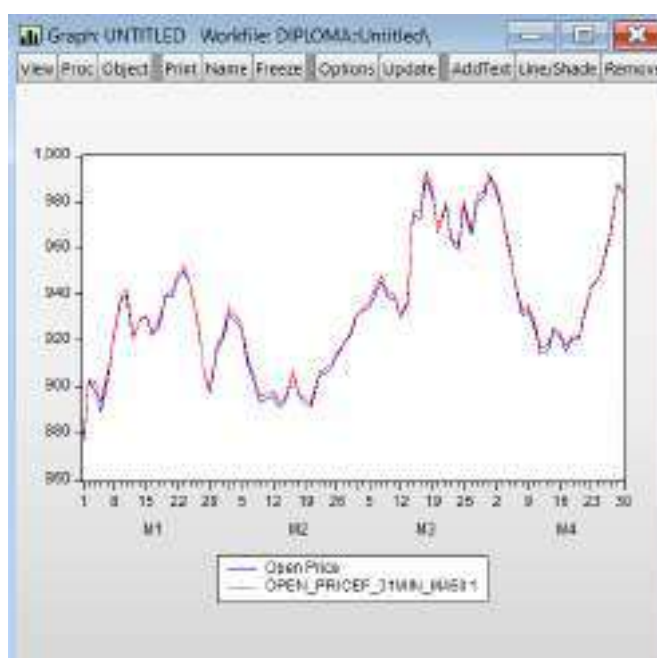


Рисунок 3. Графік вхідних даних та оцінених за моделлю АРКС(31, 5)

Подивившись ще раз на графік вхідних даних можна спробувати побудувати модель за допомогою нелінійних функцій. Оцінивши графік бачимо, що можна спробувати функцію синуса. А також Оскільки ціна відкриття в певний день на біржі на пряму залежить від ціни закриття на попередній день, то було прийняте рішення побудувати прогноз на ціну відкриття за ціною закриття попереднього дня. Порівняємо статистичні характеристики цих

моделей. Будемо порівнювати за наступними критеріями: інформаційний критерій Акаїке (це оцінювач похибки позавибіркового прогнозування, і відтак відносної якості статистичних моделей, для заданої вибірки даних). Також зручними для практичного використання є критерій Дарбіна-Уотсона і коефіцієнт детермінації (Табл. 1)

Таблиця 1. Порівняння статистичних параметрів моделей

	Критерій Акаїке	Критерій Дарбіна-Уотсона	Коефіцієнт детермінації
AP (31)	9,92	1,94	0,992
АРКС (31, 5)	9,92	1,95	0,992
sin	9,92	1,93	0,992
Попередній день	9,31	1,99	0,990

У перших трьох моделей критерій Акаїке та коефіцієнт детермінації однакові, тому дивлячись саме на критерій Дарбіна-Уотсона, вибираємо кращу серед них АРКС (31,5). У моделі побудованій на даних попереднього дня найкраще значення критерію Дарбіна-Уотсона. При порівнянні прогнозованих даних по кожній моделі до реальних найближчими даними до реальних є дані прогнозовані по моделі AP (31), також досить близькими є дані прогнозовані за допомогою нелінійної моделі. Також, не дивлячись на гарні статистичні параметри моделі, побудованою за даними попереднього дня, результати прогнозування є найбільш віддаленими від реальних.

За допомогою нейронних мереж (використовуючи інструмент Google Coolab), побудовані одношарові та багатошарові нейронні мережі, використовуючи різні види рекурентних нейронів, такі як Simple RNN, LSTM, GRU. Кожен з цих видів нейронів показував себе краще при збільшенні кількості нейронів на один шар. Також були використані згорткові нейронні мережі, що показали себе не досить ефективно, що можна пояснити. Адже згорткові нейронні мережі мають ефективне прикладне застосування в розпізнаванні зображень та відео, рекомендаційних системах та обробці природної мови. Порівняння статистик якості прогнозів різних нейронних мереж подано в Табл. 2.

Таблиця 2. Порівняння результатів застосування нейронних мереж

№	Вид	Шарів	Нейронів	drop	epochs	MSE		MAPE		MAE	
						train	val	train	val	train	val
1	SimpleRNN	1	128	-	20	0.003	0.035	20.249	44.227	0.042	0.121
2	LSTM	1	128	-	20	0.004	0.032	19.768	47.555	0.045	0.109
3	GRU	1	128	-	20	0.002	0.015	15.902	33.849	0.033	0.069
4	GRU	3	256	0.3	40	0.004	0.025	18.364	32.569	0.05	0.099
5	LSTM	2	256	-	30	0.003	0.023	17.514	38.291	0.038	0.09
6	LSTM	2	256	0.1	35	0.011	0.089	32.738	54.326	0.08	0.189
7	RNN + Conv1d	3 + 2	128	0.4	20	0.006	0.016	20.077	32.82	0.062	0.077

Всі розглянуті моделі мають спільний недолік: вони здатні прогнозувати лише на крок вперед. В даних умовах, зміна реального фінансового ряду не визначається лише авторегресійною складовою, а здебільшого зовнішніми чинниками, інформацію про які ми не маємо. В даному випадку: коли ми перевіряємо здатність узагальнення одноточкового прогнозу на великому періоді даних, то однією з найкращих моделей слід прийняти мережу з одним шаром GRU-комірок, що показала найкращий результат за усіма метриками на валідації, та при цьому має доволі мало параметрів. Також достатньо ефективними моделями показали себе GRU з трьома шарами нейронів та RNN + Conv1d.

#### **4. ВИСНОВКИ**

Сьогодні важко назвати область діяльності людини, де не має застосування моделювання. Моделювання виступає як процес поглиблення пізнання, повніше розкриває сутність явищ дійсності, що досліджуються. Сучасні технології моделювання не тільки полегшили і прискорили процес побудови та дослідження моделі, але й значно наблизили сприйняття інформації спеціаліста з моделювання систем і спеціаліста, що працює у галузі, яка моделюється. В якості прогнозованих даних було обрані дані з фінансової біржі. Були розроблені прогнози за допомогою системи EViews та нейронних мереж.

За допомогою системи EViews побудовано п'ять моделей для подальшого прогнозу: AP (31), APKC (31,5), модель на основі нелінійної функції ( sin), AP (6) для логарифмованих даних та модель на основі даних попереднього дня. Було обрано робити нелінійну модель зважаючи на графік вхідних даних. Найкраще з цих моделей показала себе APKC (31,5). У моделі побудованій на даних попереднього дня найкраще значення критерію Дарбіна-Уотсона, але якщо дивитись саме на значення прогнозованих даних, то вони є найбільш віддаленими від реальних. У моделі AP (6) для логарифмованих даних найменший критерій Акайке серед усіх значень, тому на даний момент можемо вважати її найкращою.

За допомогою нейронних мереж було побудовано багато прогнозів, варіюючи кількість нейронів, епох та саме типу нейронів. Найкраще себе показали нейронні мережі GRU, як багатошарові так і одношарові, оскільки вони були одразу максимально наближеними до реальних даних. Всі нейронні мережі давали значно кращі результати зі збільшення кількості нейронів та епох.

#### **ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Бідюк П.І., Половцев О.В. Аналіз та моделювання економічних процесів перехідного періоду. К.: НТУ КПП, 1999. 230 с.
2. Ставицький А. В. Навчально-методичний комплекс з курсів «Прогнозування» та «Фінансове прогнозування». Київ: Центр учб. літ., 2006. 107 с.
3. Половцев О. В. Системний підхід до моделювання, прогнозування та управління фінансово-економічними процесами. Донецьк: Східний видавничий дім, 2009. 286 с.
4. Бідюк П. І., Романенко В. Д., Тимошук О. Л. Аналіз часових рядів: навч. посіб. / ННК «Інститут прикладного системного аналізу» Національний технічний університет України «Київський політехнічний інститут», 2010. 317 с.
5. Молчанов И. Н. Компьютерный практикум по начальному курсу эконометрики (реализация на Eviews): практикум / Ростовский государственный экономический университет. Ростов-н/Д. 2001. 58 с.

# ЗАСТОСУВАННЯ МІКРОСЕРВІСНОГО ТА МІКРО-ФРОНТЕНДНОГО ПІДХОДІВ ДЛЯ ЗАДАЧІ СТВОРЕННЯ ВІРТУАЛЬНОГО КАБІНЕТУ ЛІКАРЯ ТА ПАЦІЄНТА

Безносик О.Ю.<sup>1</sup>, Стефура О.Я.<sup>2</sup>

Національний технічний університет України  
«Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна

<sup>1</sup>beznosyk.oleksandr@lil.kpi.ua [0000-0003-2775-6070], <sup>2</sup>stefura2010@ukr.net

Впродовж останніх років відмічається пришвидшений темп розвитку різноманітних технологій та підходів для реалізації широкого спектру веб-систем. З одного боку це стимулюється інтеграцією все нових і нових систем у сферу веб, з іншої – зростаючою складністю вже існуючих. Нові підходи та технології покликані задовільнити технологічні запити складних систем, спростити їх реалізацію. Разом із тим, однією із найбільш цікавих і водночас складних галузей є галузь охорони здоров'я. Зазвичай, такі системи вирізняються високими технологічними та етичними вимогами, складними внутрішніми процесами, потребою в постійній підтримці та оновленнях. Це створює природний інтерес до застосування сучасних підходів для спрощення та покращення функціонування таких систем. Спроба такої інтеграції представлена в цій роботі.

**Ключові слова:** мікросервісна архітектура, мікрофронтенд, складні системи, охорона здоров'я.

## 1. ВСТУП

Галузь охорони здоров'я є однією із найбільш популярних та таких, що динамічно розвиваються, з точки зору застосування різноманітних програмно-апаратних розробок. Існує дуже велика кількість різноманітних веб-систем, додатків, ресурсів в цій галузі, доступних для користування як професіоналами, так і звичайними користувачами. Не є чимось новим і онлайн системи для взаємодії лікарів, пацієнтів, представників страхових компаній, які, втім, часто є представниками саме класу складних систем, що висувають більш складні технічні вимоги до розробників. Тому природньо, що ця галузь входить до числа доменів, що особливо активно інтегруються із мікросервісним підходом. Існує досить велика кількість присвячених цій проблематиці статей, зокрема [1] або [2] тощо, де автори розглядають переваги та можливі проблеми, пов'язані з імплементацією такого підходу в галузь охорони здоров'я.

Зазвичай, подібні системи характеризуються високими вимогами до охорони даних користувача, вимагають поєднання різноманітних механізмів, але при цьому з точки зору користувача мають бути доволі простими для опанування, інтуїтивно зрозумілими, адже, згідно статистики, більшість активних користувачів подібних систем (саме через їх певну складність) все ще знаходиться в межах 18-34 років [3], хоча ця група людей напевне не є тою, якій найчастіше необхідна допомога лікарів.

На погляд авторів, означений набір проблем може бути вирішений за допомогою декомпозиційного підходу. У цій статті пропонується архітектура додатку, що реалізує

віртуальний кабінет лікаря та пацієнта, яка є ефективною з точки зору вирішення вищезначених задач і проблем.

## **2. АРХІТЕКТУРА СИСТЕМИ ВІРТУАЛЬНОГО КАБІНЕТУ ЛІКАРТЯ ТА ПАЦІЄНТА**

### **2.1. Вимоги до системи**

Головним чином, дана система створюється для взаємодії між лікарем і пацієнтом в рамках створення та проведення плану лікування тих чи інших проблем. Відповідно, система має забезпечувати, в першу чергу, комфортний інтерфейс для лікаря для створення плану лікування, його кроків та обмежень, а також не менш зручний інтерфейс для пацієнта для перегляду плану та взаємодії з ним (зокрема, можливість ставити відмітки про виконання тих чи інших приписів, уточнювати план лікування, вносити показники чи результати тих чи інших процедур в рамках кроків лікування). Система, звісно, повинна передбачати й механізм спілкування між лікарем та пацієнтом для швидкого обміну повідомленнями, можливості встановлення швидкого контакту. Але це тільки кореневий функціонал, який представляє основну практичну цінність. Як було сказано раніше, системи із галузі охорони здоров'я зазвичай відрізняються поєднанням різноманітних компонентів, що і обумовлює їхню складність. Тому система, що пропонується, повинна передбачати ще деякий функціонал, зокрема:

- перегляд персональної інформації про пацієнта, зокрема його електронної медичної карти (ЕКМ);
- моніторинг поточного стану пацієнта на основі даних, що ним вносяться;
- автоматичне сповіщення про стан здоров'я на основі внесених показників;
- допоміжні системи супроводу плану лікування, що допомагають як лікарю в менеджменті плану, так і пацієнту в його проходженні.

Безумовно, система також повинна бути готова й до внесення нового функціоналу, до інтеграції із зовнішніми сервісами, і, звісно, передбачати додаткові заходи зі збереження конфіденційності даних користувачів.

В результаті, пацієнт та лікар отримують комфортні інтерфейси, насичені підказками та всім необхідним для ефективного створення та проходження плану лікування. При цьому присутні механізми запобігання когнітивному перевантаженню користувачів: всі дані відображаються лише за потреби, є можливість керування їх відображенням.

Вже із самого опису випливає, що система містить в собі різний функціонал, котрий може (і повинен) бути рознесений в різні її компоненти. Так, можна виділити такі ключові компоненти системи

- безпосередній менеджмент плану лікування;
- механізми доступу до медичної карти та історії пацієнта;
- механізми аналітики поточного стану пацієнта;
- підсистема рекомендацій (лікарю та пацієнту);
- окремий механізм забезпечення конфіденційності даних.

В той же час, в системі мають бути передбачені можливості для майбутнього розширення функціоналу системи, покращення окремих компонентів.

### **2.2. Опис архітектури**

Згідно заданих вище умов, було запропоновано архітектуру системи, що містить набір компонентів та забезпечує взаємодію між ними, як показано на рисунку 1.

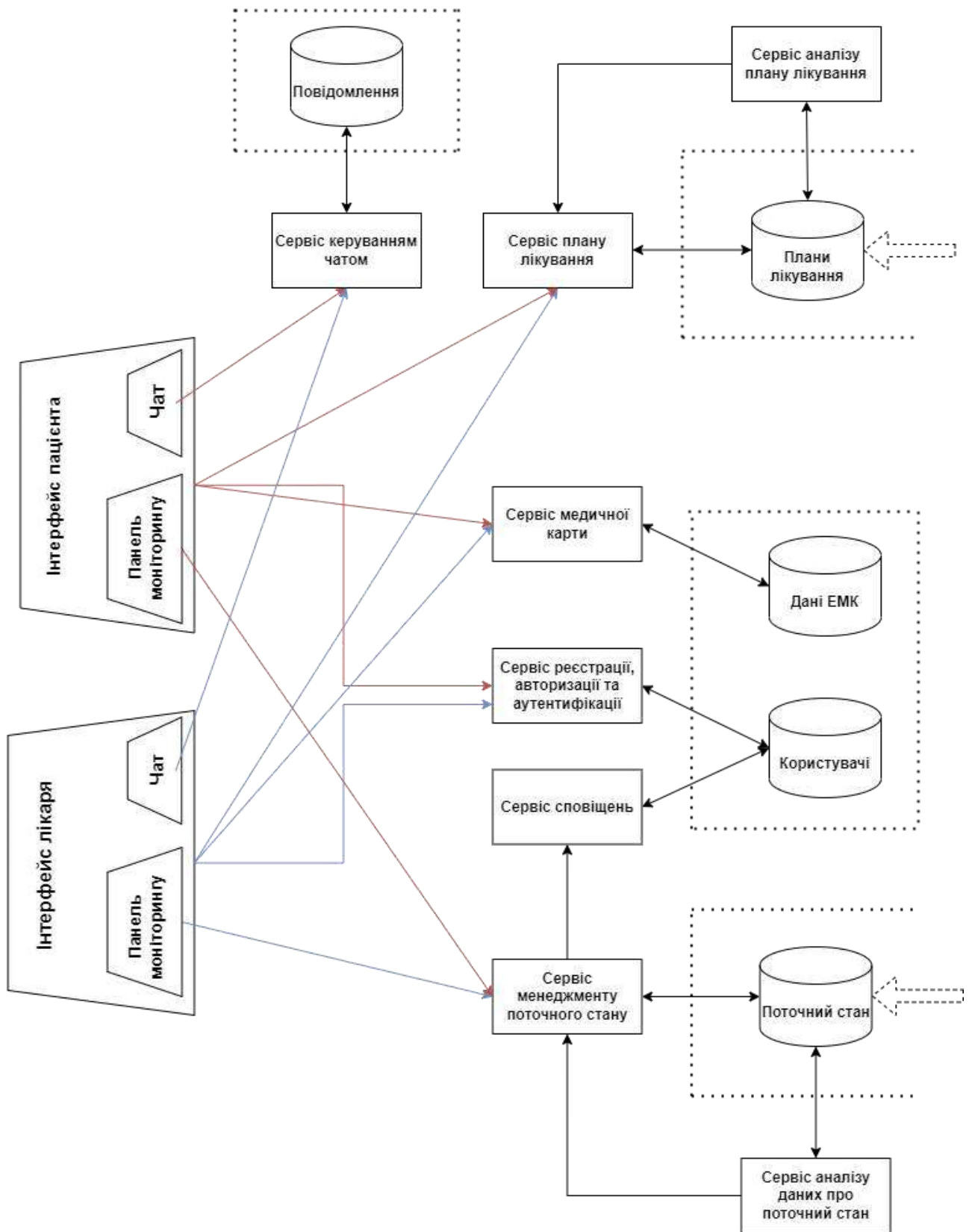


Рисунок 1. Архітектура системи віртуального кабінету лікаря та пацієнта

Компоненти системи виконують наступну роль:

Сервіс реєстрації, авторизації, автентифікації – один із фундаментальних сервісів, відповідає за правильну політику надання доступу до ресурсів, зокрема – до приватних даних.

Сервіс медичної карти – забезпечує надання персональної медичної інформації про пацієнтів.

Сервіс плану лікування – надає механізми менеджменту плану лікування, такі як створення кроків, відмітки про виконання, доступ на перегляд тощо.

Сервіс керування чатом – забезпечує механізм спілкування лікаря\пацієнта.

Сервіс сповіщень – організовує сповіщення пацієнтів\лікарів про різні події (зокрема, про кризові стани) через різні канали, як, наприклад, СМС-повідомлення або електронна пошта.

Окремий інтерес представляють такі “допоміжні” або “аналітичні” сервіси:

Сервіс аналізу даних про поточний стан – постійно аналізує вміст відповідної бази даних, та, за необхідності, видає сигнал із попередженнями або підказками сервісу менеджменту поточного стану. Той, у свою чергу, може ініціювати сповіщення чи оновлення інформаційної панелі інтерфейсу.

Сервіс аналізу плану лікування – аналізує наявні плани лікування та видає рекомендації та підказки.

Варто відмітити деякі технічні особливості системи:

- «чутлива» персональна інформація, яка дозволяє однозначно ідентифікувати користувача системи, зібрана в одному місці (на рисунку – блок з елементами «Користувачі», «Дані ЕМК»). Таким чином, ці таблиці можна винести в окремий простір, навіть в окрему базу даних, яка буде мати додаткові механізми захисту (відповідно, потребуватиме більше ресурсів). Таблиці в інших просторах не мають посилання на конкретних користувачів, відповідно, при викраденні зловмисниками, не дозволяють однозначно ідентифікувати, кому вони належать. Необхідні дані для співставлення планів лікування\повідомлень\поточного стану можуть бути отримані користувачами лише у відповідному просторі з «чутливими даними». Для наочності, приклад відповідної організації таблиць приведено на рисунку 2;

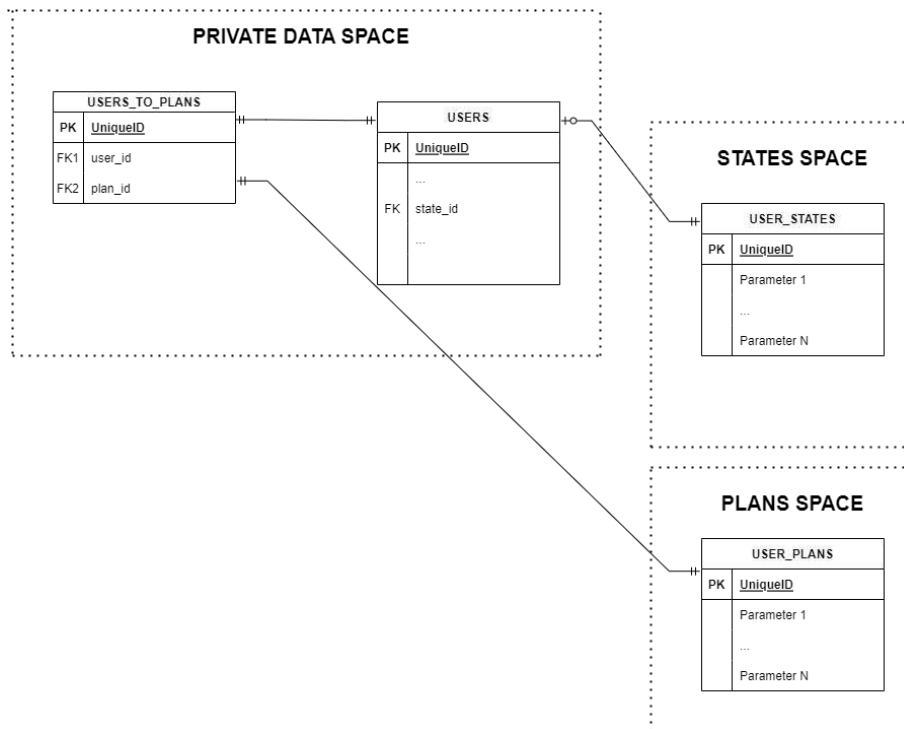


Рисунок 2. Відношення між таблицями з окремих просторів

- на сервісному рівні та\або на рівні бази даних можна обмежити можливість деяких сервісів вносити зміни в ті чи інші таблиці. Наприклад, аналітичні сервіси тільки зчитують дані, щоб далі їх аналізувати; сервіс ЕМК тільки видає дані про картки, не маючи можливість вносити зміни в них; лікар може лише переглядати дані поточного стану, не маючи змоги їх змінити;
- оскільки таблиці, котрі описують поточні стани та плани лікування є, по суті, «анонімними», є можливість відкрити до них доступ на читання стороннім сервісам для виконання іншого роду практичної аналітики або для надання інформації зовнішнім статистичним сервісам.

### 2.3. Декомпозиція інтерфейсу

Підхід із розбиттям на компоненти (мікросервіси) застосовується не тільки у внутрішній складовій системи, але й на стороні інтерфейсу користувача. Такий підхід має назву «мікрофронтендний» [4, 5]. В даній системі він використовується для інтеграції із аналітичною панеллю, чатом, в перспективі – з іншими компонентами.

Це дозволяє не тільки спростити роботу в рамках команд, що працюють над одним проектом, але й відкриває дорогу для використання зовнішніх постачальників послуг, а також для підходу «white label» [6].

Схематично, приклад сторінки плану лікування в такому випадку може виглядати так, як показано на рисунку 3. В ньому панель моніторингу (виділено синім кольором) та чат (виділено оранжевим кольором) імпортуються як окремі залежності ззовні та, за потреби, можуть бути вимкнені користувачами.

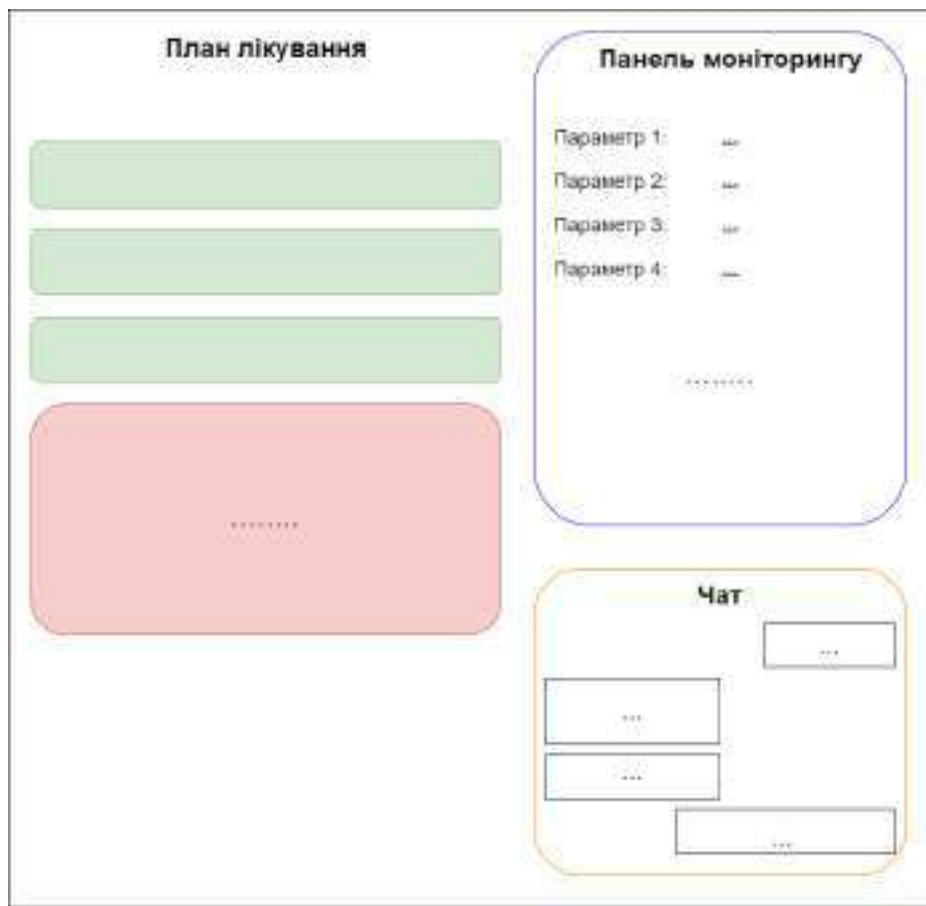


Рисунок 3. Схематичний вигляд інтерфейсу, скомпонованого із різних частин (окремих субінтерфейсів)

### 3. ВИСНОВКИ

Ця робота пропонує модель архітектури системи з акцентом на декомпозицію та слабку зв'язність компонентів не тільки на серверній стороні (що на даний момент доволі розповсюджено), але й на стороні інтерфейсу користувача (що поки не є стандартом). В даній системі така декомпозиція дозволяє, з одного боку, інкапсулювати певні критичні дані і механізми роботи над ними, відмежувавши їх від інших складників системи (з метою забезпечення), а з іншого – вивести інші, не критичні дані в своє окреме місце, що відкриває дорогу для інтеграції зі сторонніми сервісами.

Враховуючи специфіку галузі, така декомпозиція (в тому числі, на стороні інтерфейсу користувача) є досить вдалою, оскільки системи в цій галузі доволі часто є своєрідними “акумуляторами” даних з різних джерел і систем, відповідно, модульний підхід дозволяє без особливих перешкод розширювати систему, налаштовувати канали зв'язку із зовнішніми системами (як-то постачальники медичної інформації, аналітичні системи, “white label” продукти, наприклад, чати чи моніторингові системи).

Звісно, така декомпозиція має свою ціну і складнощі реалізації, однак у даному випадку є виправданою, враховуючи значний функціонал системи, вимоги до безпеки, масштабованості та інтеграції (і, разом із тим, відносно не критичні вимоги до продуктивності та швидкодії, на що така декомпозиція має безпосередній негативний вплив).

## ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. R. Hill, D. Shadija, M. Rezai, “Enabling Community Health Care with Microservices,” *2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC)*, Sept. 2017, pp. 1444-1450, doi: 10.1109/ISPA/IUCC.2017.00220.
2. R. Muhammad, N.F. Ahmad, R. Astari, “Microservices Architecture Design: Proposed for online HealthCare,” *International Journal of Emerging Trends in Engineering Research*, Volume 8, No. 4, Apr. 2020, pp. 1040-1047, doi: 10.30534/ijeter/2020/14842020.
3. Health app usage among adults in the United States as of November 2019, by demographics. URL: <https://www.statista.com/statistics/1193612/health-app-user-demographics-us/> (дата звернення: 17.11.2022).
4. Michael, G. Micro Frontends. Techniques, strategies and recipes for building a modern web app with multiple teams that can ship features independently. URL: <https://micro-frontends.org/> (дата звернення: 18.11.2022).
5. Sam, J. Micro Frontends. URL: <https://martinfowler.com/articles/micro-frontends.html> (дата звернення: 16.11.2022).
6. Amanda, L. White Label Software Explained: Benefits and Examples. URL: <https://growsurf.com/blog/white-label-software> (дата звернення: 18.11.2022).

# ПІДХОДИ ДО ОПТИМІЗАЦІЇ ЗАПИТІВ ДО РЕЛЯЦІЙНИХ БАЗ ДАНИХ З ВИКОРИСТАННЯМ МАШИННОГО НАВЧАННЯ

Булах Б.В.<sup>1</sup>, Загородній Д.О.<sup>2</sup>

Національний технічний університет України  
«Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна

<sup>2</sup>dimkazhr@gmail.com

**Використання машинного навчання, а саме нейронних мереж для підвищення ефективності та оптимізації створення плану запиту. Метою роботи є дослідження сучасних СУБД з акцентом на механізми оптимізації запитів. Дослідити можливості використання нейронних мереж для задач оптимізації запитів. Визначити чи ефективно використовувати нейронні мережі для цієї задачі. Розробити додаток що може продемонструвати ефективність використання нейронних мереж для оптимізації запитів.**

**Ключові слова:** реляційні бази даних, нейронні мережі, оптимізація запитів, СУБД, навчання з підкріпленням.

## 1. ВСТУП

Основні сучасні інформаційні технології базуються на концепції зберігання даних, згідно якої усі дані мають бути організовані у бази даних, так щоб вони могли відображати реальний світ та його мінливість, а також бути зручними та задовольняти усім інформаційним потребам користувачів. Для створення і функціонування таких баз даних існують спеціальні програмні комплекси, які називаються системами управління базами даних (СУБД).

Усі підприємства та установи, починаючи від банків, яким необхідно зберігати інформацію про клієнтів, їх рахунки та транзакції, та закінчуючи лікарнями, для яких необхідна база пацієнтів, хвороб та іншого, використовують сучасні бази даних. Усі сучасні інформаційні системи використовують сучасні СУБД для зберігання своїх даних. Наразі дані та їх структура з кожним роком стають все складніші та складніші, кількість користувачів стрімко зростає, а питання оптимізації операцій з даними стає все більш актуальним. Через це і постає проблема оптимізації СУБД. У даній роботі будуть розглянуті способи оптимізації запитів до сучасних СУБД, а саме методи використання нейронних мереж для оптимізації запитів.

Оптимізація запитів — це 1) функція СУБД, яка виконує пошук оптимального плану виконання запиту з усіх можливих для заданого запиту, 2) процес зміни запиту та/або структури БД в цілях зменшення використання розрахункових ресурсів при виконанні запиту. Один і той же результат може бути отриманий СУБД різноманітними способами (планами виконання запиту), які можуть суттєво відрізнятися як за затратами ресурсів, так і за часом виконання. Головним завданням оптимізації запитів є системні скорочення ресурсів, необхідних для виконання запиту, що в кінцевому підсумку надає користувачеві результати за менший проміжок часу, що робить додаток комфортнішим для користувача; надає можливість програмному додатку обслуговувати більше запитів за одиницю часу, оскільки кожен запит займає менше часу, ніж неоптимізовані запити; зменшує навантаження на обладнання і дозволяє серверу працювати більш ефективно.

Наразі є не мало досліджень про можливості оптимізації баз даних з використанням машинного навчання, починаючи від лінійної регресії, яка використовується з метою більш точного розрахунку використання ресурсів кожним з планів, закінчуючи використанням нейронних мереж з дослідженням їх використання для заміни сучасних методів оптимізації плану (грубої сили, динамічне програмування, генетичний алгоритм). У порівнянні з сучасними методами оптимізації запитів, використання нейронних мереж дозволяє пришвидшити роботу пошуку плану запиту до 10 раз.

## 2. ПРОБЛЕМА СУЧАСНИХ АЛГОРИТМІВ ОПТИМІЗАЦІЇ ЗАПИТІВ

SQL – декларативна мова. Виходячи з цього маємо, що користувач вказує СУБД лише операції мають бути проведені з даними. За вибір способу виконання цих операцій відповідає вже сама СУБД.

Для прикладу простий запит:

```
SELECT user FROM users WHERE year > 1980;
```

може бути виконаний двома способами: читання усіх записів з таблиці users та перевірка кожної з них на виконання умови, або використовувати індекс по полю year. У другому випадку ми не проглядаємо зайві записи, але витрачаємо більше часу на обробку одного запису через операції з індексами.

Розглянемо складніший запит:

```
SELECT messages.content FROM messages, users WHERE messages.user_id = users.id;
```

Це з'єднання можливо виконати вже трьома різними способами:

- вкладений цикл проглядає усі можливі пари записів з двох таблиць та перевіряє для кожної пари виконання умови;
- злиття відсортує обидві таблиці по полям id та user\_id відповідно, а далі використає метод двох вказівників для пошуку всіх пар записів, які задовольняють умові. Цей метод аналогічний методу сортування злиттям;
- хешування будує хеш-таблицю по полю найменшої таблиці. Хеш-таблиця дозволяє для кожного запису з однієї таблиці знайти запис з відповідністю.

Також якщо в запиті необхідно виконати більше однієї операції з'єднання таблиць, то також можна виконувати їх у різному порядку. Отриманий результат буде деревом виконання запиту (рис.1), що і являє собою план виконання запиту.

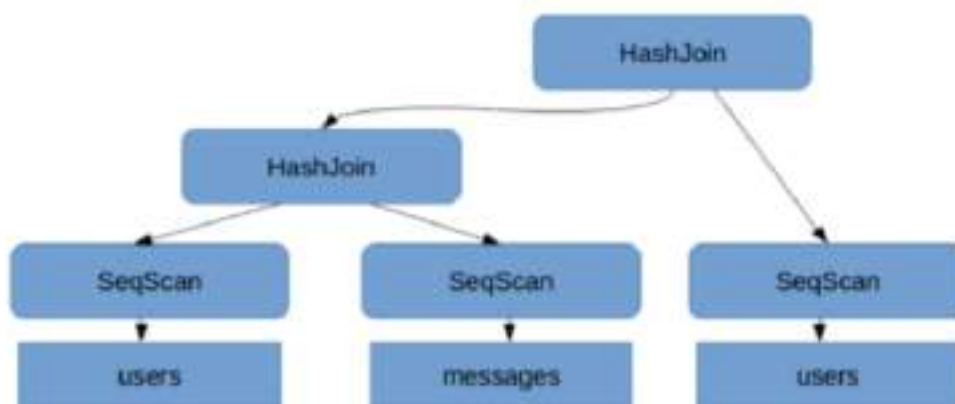


Рисунок 1. Дерево виконання запиту

Час виконання різних планів одного й того ж самого запиту може суттєво відрізнятись на багато порядків. Тому правильний вибір плану запиту сильно впливає на роботу СУБД.

На прикладі СУБД PostgreSQL розберемося як вибирається план запиту зараз. Процес пошуку оптимального плану можна розділити на дві частини:

- по-перше, необхідно вміти оцінювати вартість будь-якого плану – кількість ресурсів, що необхідні для його виконання. У випадку, коли на сервері виконуються інші задачі та процеси, оцінений час є прямо пропорційний кількості витрачених на нього ресурсів. Тому можна вважати, що вартість плану – це його час виконання в деяких умовних одиницях;
- по-друге, необхідно вибрати план з мінімальною оцінкою вартості.

Легко показати, що кількість планів зростає експоненційно зі збільшенням складності запиту, тому не можна просто перебрати усі плани, оцінити вартість кожного та обрати найбільш дешевий. Для пошуку оптимального плану використовуються більш складні алгоритми дискретної оптимізації: динамічне програмування по підмножинам для простих запитів та генетичний алгоритм для складних.

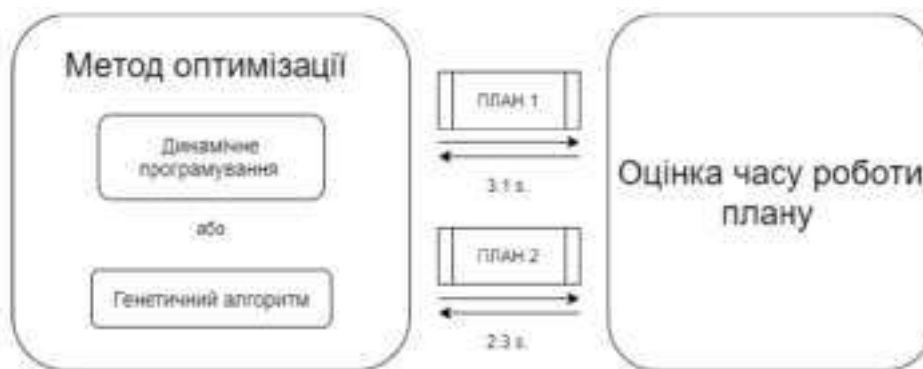


Рисунок 2. Схема процесу пошуку оптимального плану

Основною проблемою даного підходу є те що СУБД ніяк не враховує попередній досвід, через що, наприклад, деякі запити, які часто виконуються можуть мати не оптимальний план запиту, і СУБД все одно буде використовувати його. Проблема є з двома етапами побудови запиту, як кожен раз буде проводитись помилкова оцінка часу плану роботи, так і обраний метод оптимізації може виявитись не оптимальним. Саме цю проблему можливо вирішити завдяки використанню машинного навчання, в нашому випадку ми будемо розглядати оптимізацію побудови плану з використанням навчання з підкріпленням.

### 3. ВИКОРИСТАННЯ НАВЧАННЯ З ПІДКРІПЛЕННЯМ ДЛЯ ОПТИМІЗАЦІЇ ЗАПИТІВ

Навчання з підкріпленням – область машинного навчання, яка вивчає те як інтелектуальні агенти повинні приймати рішення у середовищі з ціллю максимізації отриманої нагороди.

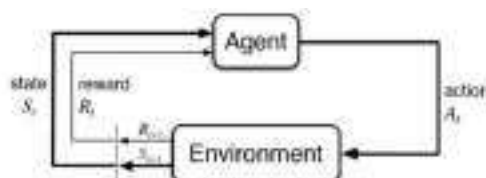


Рисунок 3. Схема навчання з підкріпленням

Постановка задачі: замість вирішення проблеми впорядкування з'єднань з використанням методів динамічного програмування або генетичних алгоритмів, ми

формуємо проблему як процес прийняття рішень Маркова (MDP) та вирішуємо її за допомогою використання навчання з підкріпленням. Тобто ми маємо визначити певне середовище, що може надавати агенту свій стан, далі в залежності від дії обраної агентом, змінювати стан, та очікувати нової дії від агента. Виконуємо цей процес ітеративно доки не досягаємо деякого кінцевого стану, коли не залишиться доступних дій – отримання плану впорядкування з'єднань, після цього агент завершує серію та отримує нагороду в залежності від прийнятих рішень. Ціль агента – максимізувати нагороду, спираючись на досвід.

Щоб використовувати навчання з підкріпленням для вирішення задачі про порядок з'єднань, ми збираємося використати наступні елементи:

- кожен стан буде представляти собою бінарне дерево з'єднань. Далі ми перетворюємо це дерево у вектор стану та додаємо додаткову інформацію, таку як предикат з'єднання та предикат вибору;

- кожна дія являє собою об'єднання двох піддерев у одне дерево;

- епізод закінчується, коли ми приєднали усі відносини;

- винагорода буде розраховуватись на основі оцінки вартості моделі з результату порядку з'єднань.

Можемо побудувати схему додатку що буде використовувати навчання з підкріпленням для пошуку оптимального порядку з'єднань (рис. 4).



Рисунок 4. Схема використання навчання з підкріпленням для задачі створення оптимального порядку з'єднань

#### 4. РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Поглянемо на результати навчання. Маємо набір тестових запитів з різною кількістю з'єднань. Для прикладу візьмемо наступний набір запитів з великою кількістю з'єднань: 19a.sql, 19b.sql, 19c.sql. Поставимо загальну кількість епізодів на 900. Результати будуть подані у вигляді графіків для кожного епізоду окремо, де по осі x епізоди, а по осі y вартість побудованого запиту. Поглянемо на результати навчання:

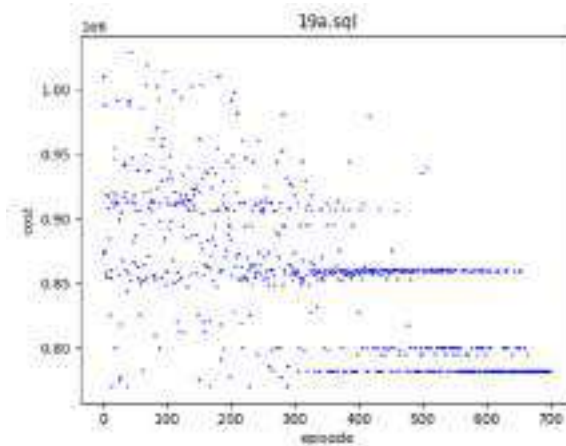


Рисунок 5. Результати навчання для запиту 19a.sql

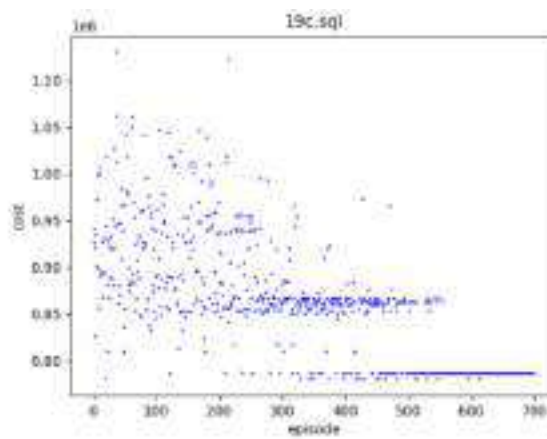


Рисунок 6. Результати навчання для запиту 19c.sql

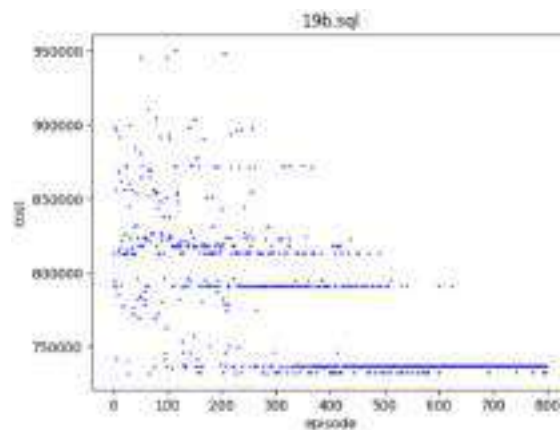


Рисунок 7. Результати навчання для запиту 19b.sql

Складемо таблицю у якій порівняємо отримані результати з вартістю запити, який складає СУБД PostgreSQL:

Таблиця 1. Порівняння результатів навчання з роботою СУБД.

Запит	Вартість за результатами навчання	Результат СУБД PostgreSQL
19a.sql	751255	767747
19b.sql	730011	731337
19c.sql	765125	766012

Як бачимо був отриманий хоч і не великий, але все ж приріст ефективності. Збільшити його можна було б якби агент вибирав не лише порядок з'єднання, але і метод приєднання. Провівши тестування для різних груп запитів приходимо до середніх результатів навчання агента:

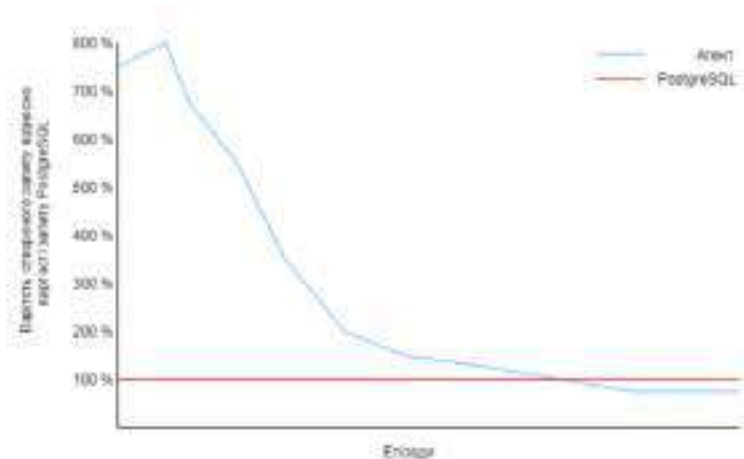


Рисунок 8. Результати тестування

Отже робимо висновок, що використання навчання з підкріпленням є ефективним для вирішення проблеми пошуку порядку з'єднань для запитів з великою кількістю з'єднань та великих БД.

## 5. ВИСНОВКИ

Отримані результати тестування показують, що використання нейромереж на запитах з великою кількістю з'єднань може бути ефективним. Було виявлено що оптимізатор запитів, що розроблений на основі навчання з підкріпленням, після навчання, ефективно вирішує проблему оптимізації порядку з'єднань у плані запиту. Для ще більшого підвищення ефективності оптимізації даним методом можна створити оптимізатор, що не лише створює порядок з'єднань у запиті, але і обирає метод з'єднань. Але є і недоліки розробленого рішення, а саме те що для кожної бази даних необхідно використовувати індивідуального агента, якого необхідно навчати окремо, а також при змінах у схемі бази даних агента необхідно перевчити. Отже робимо висновок що використання нейронних мереж для оптимізації запитів є ефективним рішенням, що дозволить пришвидшити час створення запиту та його виконання, але його доцільно використовувати для баз даних з чітко визначеною схемою. Використовуючи навчання з підкріпленням для вирішення цієї проблеми можна обійти у швидкодії сучасні динамічні алгоритми що для цього створені.

## ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Система управління базами даних. URL: [https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0\\_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0)
2. План виконання запиту. URL: [https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%BD\\_%D0%B2%D1%8B%D0%BF%D0%BE%D0%BB%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F\\_%D0%B7%D0%B0%D0%BF%D1%80%D0%BE%D1%81%D0%B0](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%BD_%D0%B2%D1%8B%D0%BF%D0%BE%D0%BB%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B7%D0%B0%D0%BF%D1%80%D0%BE%D1%81%D0%B0)
3. Методи оптимізації виконання запитів у реляційних СУБД. URL: [http://citforum.ru/database/articles/art\\_26.shtml](http://citforum.ru/database/articles/art_26.shtml)
4. Query Optimizer in PostgreSQL. URL: <https://bitnine.net/blog-postgresql/query-optimizer-in-postgresql/>
5. Reinforcement learning. URL: [https://en.wikipedia.org/wiki/Reinforcement\\_learning](https://en.wikipedia.org/wiki/Reinforcement_learning)
6. Using Reinforcement Learning to Produce Better Join Ordering Strategy. URL: <https://towardsdatascience.com/using-reinforcement-learning-to-produce-better-join-ordering-strategy-2fd2761ebf3a>
7. SQL Query Optimization Meets Deep Reinforcement Learning. URL: <https://rise.cs.berkeley.edu/blog/sql-query-optimization-meets-deep-reinforcement-learning/>
8. Join-order-benchmark. URL: <https://github.com/gregrahn/join-order-benchmark>

# МОДЕЛІ НЕЧІТКОГО ПРОГНОЗУВАННЯ В ВИМІРЮВАННЯХ

Грищенко О.Ю.<sup>1</sup>, Рогоза В.С.<sup>2</sup>

КПІ ім. Ігоря Сікорського, Київ, Україна

<sup>1</sup> helengrischenko00@gmail.com [https://orcid.org/0000-0001-6888-8665],

<sup>2</sup> rosvetnik@gmail.com

## 1. ВСТУП

Проблема прогнозування стану будь-якого процесу є складною та багатогранною, як і складність вироблення єдиної універсальної технології оцінки та контролю прогнозування, оскільки, як відомо, точність оцінок є обернено пропорційною складності об'єктів, характеристики яких оцінюються [5].

Складність системи прогнозування є її внутрішньою характеристикою. Різні методи штучного інтелекту (AI) були використані для реалізації системи прогнозування [3]. Моделі прогнозування на основі ШІ можна класифікувати на чотири групи: моделі на основі нейронних мереж, нечіткої логіки, генетичного алгоритму та експертних систем.

Нечіткі обчислення є основою нового наукового напрямку, названого «м'які обчислення» або «обчислювальний інтелект», який сформувався в останні 25 років, і включає також нейрообчислення, еволюційні і генетичні алгоритми, міркування на основі свідочств, мережі довіри та інші [3].

Вимірювання є найбільш фундаментальним методом науки в отриманні знань та управлінні системами. У складних інженерних і наукових системах частина інформації може надходити від датчиків, а інша частина може надаватися експертами. Через свою природу ці джерела відрізняються надійністю та невизначеністю отриманої інформації [2]. На невизначеність також можуть впливати характеристики процедур та інструментів, що застосовуються для отримання та обробки інформації [2].

Нечіткі моделі перетворюють суб'єктивне розуміння та досвід процесів у математично доступні цифри та правила для створення систем із певним ступенем невизначеності. Використання нечіткої логіки в експериментальній розробці програмного забезпечення для моделювання різних аспектів процесу еволюції програмного забезпечення все більше отримує визнання дослідницької спільноти [1].

## 2. КОНЦЕПЦІЯ НЕЧІТКИХ ОБЧИСЛЕНЬ

Поняття нечіткі обчислення або м'які обчислення було введено основоположником нечіткої логіки Л. Заде на семінарі в 1994 році, як консорціум обчислювальних методологій, які колективно забезпечують основи для розуміння, конструювання і розвитку інтелектуальних систем, зокрема, систем інтелектуального аналізу даних. Основою відмінності м'яких обчислень від традиційних жорстких обчислень - пристосованість до роботи з неточними, невизначеними або частково істинними даними, що виражається в «допустимому ставленні до неточності, невизначеності і часткової істинності для досягнення зручності маніпулювання, низької вартості рішення і кращої згоди з реальністю».

М'які обчислення не гарантують, що знайдене рішення буде оптимальним або буде досягнутий глобальний екстремум за прийнятний час. Проте вони можуть застосовуватися для пошуку допустимого рішення задачі за «достатньо короткий час». В рамках м'яких обчислень кожна з методологій має відмінності у використанні. Зокрема, нечітка логіка

працює з неточністю, зернистою структурою (гранульованою) інформації, наближеними міркуваннями і обчисленнями із словами. Слід відмітити, що поняття нечіткості цілком узгоджується з нашими інтуїтивними уявленнями про навколишній світ. Велика частина понять, які ми використовуємо, за своєю природою нечіткі і розмиті і спроба загнати їх в рамки загальної логіки приводить до неприпустимих спотворень.

Незважаючи на зовнішню простоту і природність базових понять нечітких обчислень, знадобилося більше п'яти років, щоб побудувати і довести комплекс постулатів і теорем, які роблять логіку логікою, а алгебру - алгеброю. Паралельно з розробкою теоретичних основ нової науки опрацьовувалися різні можливості її практичного застосування. І в 1973 р. ці зусилля увінчалися успіхом - вдалося показати, що нечіткі обчислення можуть бути покладені в основу нового покоління інтелектуальних систем управління. Практично відразу після виходу в світ фундаментальних робіт по нечітким обчисленням одна невелика фірма з Данії застосувала викладені в них принципи для удосконалення системи управління складним виробничим процесом. Результат, що називається, перевершив всі очікування - через чотири роки прибутки від впровадження нової системи обчислювалися сотнями тисяч доларів.

### **3. ОСНОВНІ ЕЛЕМЕНТИ СИСТЕМИ НЕЧІТКОГО ПРОГНОЗУВАННЯ**

Основними елементами керованих даними систем нечіткого прогнозування є [1]:

1. Модуль фазифікації (fuzzification): змінює чіткі вхідні дані на нечіткі значення за допомогою функції належності. У побудові керованої даними системи нечіткого висновку цей процес фазифікації виконується нечітким розподілом чітких значень змінних набору даних.
2. Системи нечіткого висновку (FIS): виконують відображення вводу-виводу лінгвістично переданої інформації у формі правил. Різні потенційні алгоритми створення створюють ці правила, вивчаючи дані.
3. Модуль спрощення: містить невелику кількість правил і зберігає узгоджену базу правил. Хоча цей модуль є необов'язковим у процесі нечіткого висновку, структура GUAJE забезпечує покращену читабельність створених нечітких правил за допомогою цього модуля.
4. Дефазифікація (Defuzzification): транскрибує нечіткі результати назад у чіткі значення.

### **4. ФОРМУЛЮВАННЯ МАТЕМАТИЧНОЇ ЗАДАЧІ**

Загальна мета процедури вимірювання полягає в тому, щоб знайти оцінки вектора параметрів  $X$  деякої доступної моделі системи. Також відомо, що доступна модель передбачення щодо цих значень параметрів. Модель передбачає значення деяких функцій  $f(X)$  значень параметрів. Функції  $f()$  вводяться для узагальнення випадку, коли прогноз можна зробити щодо деякого зв'язку кількох параметрів. Прогноз зазвичай виражається як модель типу «функція  $f$  вектора параметрів  $X$  дорівнює  $R$ », де  $R$  є обмежувальним відношенням. Можна побачити, що це передбачення належить до класу узагальненого обмеження, визначеного Л.Заде [4]. Ця модель може бути формалізована як нечітка модель і описана набором функцій приналежності  $\mu(f(X))$ . Традиційний спосіб знаходження оцінок параметрів моделі на основі вимірювань проходив би через його визначення як задачу математичного програмування та пошук оцінок параметра  $X^*$  шляхом максимізації деякого критерію:

$$X^* = \underset{x}{\operatorname{argmax}} F(Y_1, Y_2, \dots, Y_n, X),$$

де  $F(\cdot)$  – функціонал, форма якого визначається методами оцінювання,  $Y_i$  ( $i=1, n$ ) – набір  $m$  вимірювань  $i$ -ї змінної.

Розглянемо інформацію моделі прогнозу як нечітке обмеження для вектора параметрів  $X$  і задане набором функцій приналежності  $\mu_i(f_i(X))$ ,  $i = 1, m$ , де  $m$  – кількість прогнозів, зроблених експертами. Кожна функція приналежності описує одну модель передбачення, як правило, щодо можливого значення результату вимірювання або лінійної комбінації (наприклад, середнього або середньозваженого) результатів вимірювання [2].

## **5. ПЕРЕВАГИ МОДЕЛЕЙ НЕЧІТКОГО ПРОГНОЗУВАННЯ**

Представлена теорія дозволяє виявити наступні переваги моделей нечіткого прогнозування в порівнянні з іншими:

- можливість оперувати нечіткими вхідними даними: наприклад, значеннями, що безперервно змінюються в часі (динамічні задачі), значеннями, які неможливо задати однозначно (результати статистичних опитів, рекламні компанії і так далі);
- можливість нечіткої формалізації критеріїв оцінки і порівняння: оперування критеріями «більшість», «можливо», «переважно» і т.д.;
- можливість проведення якісних оцінок як вхідних даних, так і вихідних результатів: маємо можливість оперувати не лише значеннями даних, але і їх мірою достовірності і її розподілом;
- можливість проведення швидкого моделювання складних динамічних систем і їх порівняльного аналізу із заданою мірою точності: оперуючи принципами поведінки системи, описаними нечіткими методами, ми по-перше, не витрачаєте багато часу на з'ясування точних значень змінних і складання рівнянь, що їх описують, по-друге, можна оцінити різні варіанти вихідних значень.

## **6. ЗАСТОСУВАННЯ НЕЧІТКИХ СИСТЕМ**

Що стосується вітчизняного ринку комерційних систем на основі нечіткої логіки, то його формування почалося в середині 1995 року. Найбільш популярні в замовників наступні пакети [6]:

- CubiCalc 2.0 RTC — одна з найбільш могутніх комерційних експертних систем на основі нечіткої логіки, що дозволяє створювати власні прикладні експертні системи ;
- CubiQuick — дешева <університетська> версія пакета CubiCalc ;
- RuleMaker — програма автоматичного витягу нечітких правил із вхідних даних ;
- FuziCalc — електронна таблиця з нечіткими полями, що дозволяє робити швидкі оцінки при неточно відомих даних без нагромодження похибки;
- OWL — пакет, що містить вихідні тексти усіх відомих видів нейронних мереж, нечіткої асоціативної пам'яті тощо.

Сьогодні елементи нечіткої логіки можна знайти в десятках промислових виробів — від систем керування електропоїздами і бойовими вертольотами до пилососів і пральних машин. Рекламні кампанії багатьох фірм (переважно японських) підносять успіхи у використанні нечіткої логіки як особливу конкурентну перевагу. Без застосування нечіткої логіки немислимі сучасні ситуаційні центри керівників західних країн, у яких приймаються ключові політичні рішення і моделюються всілякі кризові ситуації. Одним із вражаючих

прикладів масштабного застосування нечіткої логіки стало комплексне моделювання системи охорони здоров'я і соціального забезпечення Великої Британії (National Health Service — NHS), що вперше дозволило точно оцінити й оптимізувати витрати на соціальні нестатки.

## 7. ВИСНОВОК

Вимірювання як базова концепція наукового пізнання – це процес формулювання та модифікації моделей вимірюваних систем і середовища на основі отриманих результатів спостережень. При вимірюванні використовуються різні типи моделей. Їх взаємозв'язок і місце в ієрархії моделей, представлених Л.Заде в його теорії узагальненої визначеності [4], а також переваги, які ми можемо отримати від їх застосування, вимагають подальшого вивчення.

Моделі нечіткого прогнозування мають ряд переваг, основною з яких є можливість проведення якісних оцінок вхідних даних та отриманих результатів.

Проблема використання моделі нечіткого прогнозування для підвищення якості вимірювальних процедур і точності отриманих оцінок може бути вирішена шляхом злиття інформації з різних джерел, які характеризуються різним ступенем невизначеності.

### ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Rinkaj Goyal, Pravin Chandra, Yogesh Singh, "Fuzzy inferencing to identify degree of interaction in the development of fault prediction models", pp. 93-102, 2017

2. Leon Reznik and Vladik Kreinovich, "Fuzzy Prediction Models in Measurement", pp. 851 – 862, 2008

3. Chu S.C. And Kim H.S, "Automatic knowledge generation from the stock market data", Proceedings of 93 Korea Japan joint conference on expert systems, pp. 193-208, 1993

4. Zadeh L. "Granular computing as a basis for a computational theory of perceptions", Proceedings of the 2002 IEEE International Conference on Fuzzy Systems, pp. 564 – 565, 12-17 May 2002, volume 1

5. А. Ковальчук, О. Суха, Л. Фабрі, "ВИКОРИСТАННЯ МЕТОДІВ НЕЧІТКОГО ВИВЕДЕННЯ В СИСТЕМАХ ПРОГНОЗУВАННЯ СТАНІВ ПРИРОДНИЧИХ ПРОЦЕСІВ", 2009

6. Нечітка логіка. URL: [https://uk.wikipedia.org/wiki/%D0%9D%D0%B5%D1%87%D1%96%D1%82%D0%BA%D0%B0\\_%D0%BB%D0%BE%D0%B3%D1%96%D0%BA%D0%B0#%D0%9D%D0%B5%D1%87%D1%96%D1%82%D0%BA%D0%B0\\_%D0%BB%D0%BE%D0%B3%D1%96%D0%BA%D0%B0\\_%D0%B2\\_Matlab](https://uk.wikipedia.org/wiki/%D0%9D%D0%B5%D1%87%D1%96%D1%82%D0%BA%D0%B0_%D0%BB%D0%BE%D0%B3%D1%96%D0%BA%D0%B0#%D0%9D%D0%B5%D1%87%D1%96%D1%82%D0%BA%D0%B0_%D0%BB%D0%BE%D0%B3%D1%96%D0%BA%D0%B0_%D0%B2_Matlab)

# КОМБІНОВАНА АРХІТЕКТУРА ДЛЯ МОБІЛЬНИХ ЗАСТОСУНКІВ

Зарічний Я.С.<sup>1</sup>, Гіоргізова-Гай В. Ш.<sup>2</sup>, Яременко В. С.<sup>3</sup>

КПІ ім. Ігоря Сікорського, Київ, Україна

<sup>1</sup> yarik120700@gmail.com [https://orcid.org/0000-0002-7596-5857], <sup>2</sup> high.victoria@iit.kpi.ua [https://orcid.org/0000-0001-6224-3532], <sup>3</sup> yaremenko.v.s@gmail.com [https://orcid.org/0000-0001-8557-6938]

## 1. ВСТУП

Розробка масштабованих і стійких додатків є складною технічною задачею, в якій необхідно врахувати не лише базові вимоги до продукту, а й закласти можливості до розширення функціоналу, масштабованості тощо. Однак, коли справа доходить до мобільних програм або платформ, завжди існувала проблема з чистою архітектурою [1]. У випадку з Android це може бути тому, що Google не підтримує та навіть не рекомендує жодної конкретної архітектури. Apple, з іншого боку, запропонувала архітектуру MVC для UIKit, але ця пропозиція викликала багато суперечок, і багато експертів стверджували, що це не дуже гарне рішення. В роботі було досліджено два типу архітектури: односпрямовані, двоспрямовані та загальні принципи чистого програмування та побудови архітектур. До односпрямованих належать UDF побідні архітектури, до двоспрямованих MVX подібні та Чиста архітектура. Базуючись на перевагах та недоліків кожного підходу було запропоновано побудувати комбіновану архітектуру, яка надає можливість комбінувати переваги кожного типу архітектур та слідувати загальним принципам побудови чистої архітектури.

## 2. ОГЛЯД ІСНУЮЧИХ АРХІТЕКТУРНИХ ШАБЛОНІВ

MVC - схема поділу даних програми, інтерфейсу користувача і керуючої логіки на три окремих компоненти: модель, відображення та контролер - таким чином, що модифікація кожного компонента може здійснюватися незалежно. Перші приклади для розробників для iPhone SDK мали у собі приклади цього шаблону: клас UIView працював як відображення, AppDelegate як контролер усього застосунку, а прикладів щодо моделей не було. В цей час, перші архітектурні патерни для ОС Android були дуже схожими, відображення будувалось за допомогою XML файлів, а клас Activity був у ролі контролера. Також і в цьому випадку теж існувала проблема - відсутність прикладів класів моделей. Ця проблема мала своє відображення на багатьох мобільних застосунках, оскільки, призводить до збільшення класів контролерів [2].

MVP - шаблон проектування, похідний від шаблону MVC, який використовується в основному для побудови інтерфейсу користувача [3]. В цьому шаблоні роль Презентатор виконує роль Контролера у MVC. Але, нажаль, такий архітектурний шаблон теж має тіж самі проблеми, як і MVC: розробникам не завжди зрозуміло, яка логіка має відноситися до Презентатора або до Моделі. Також слід помітити, що доволі часто сутність Моделей теж відсутня у даному шаблоні та склеюється разом с Презентором. Треба відмітити, що порівнюючи MVC з MPV у мобільній розробці все ж таки MVP має певні поліпшення, тому що, частіше за все у MVC роль Контролеру виконувало саме Відображення. Це призводило до ще більших Контролерів така проблема має назву Massive View Controller.

MVVM – двонаправлений шаблон проектування, який походить як логічний розвиток MVC та MVP. Особливість даного шаблону це використання реактивної парадигми

програмування [4]. В даному шаблоні використовується механізм «зв'язування», який будується на парадигмі реактивного програмування, тому оновлення моделі автоматично призводить до оновлення відображення.

Односпрямований потік даних — це техніка, яка в основному зустрічається у функціональному реактивному програмуванні. Він також відомий як односторонній потік даних, що означає, що дані мають один і лише один шлях для передачі в інші частини програми. По суті, це означає, що дочірні компоненти не можуть оновлювати дані, які надходять від батьківського компонента [5].

Одностороннє зв'язування даних забезпечує ряд наступних ключових переваг.

- Полегшується відладка програми, оскільки розробнику відомо, звідки надходять дані.
- Зменшується схильність до помилок, оскільки розробник має більше контролю над даними.

Одним з головних недоліків UDF є те, що він може не знадобитися у проекті (принаймні, поки що). Отже, як дізнатися, що додаток справді потребуватиме UDF?

- Є великий стан програми, який потрібен у багатьох частинах програми.
- Стан програми часто оновлюється.
- Стан програми складно оновити.
- Програма має кодову базу середнього та великого розміру, над якою працює багато людей.
- Повинно бути видно, як стан оновлюється з часом.

Зазвичай компонент відображення у мобільному застосунку є функцією даних, які йому надає сервер або будь-яке інше джерело інформації. Математичні функції є чистими функціями, а це означає, що певний вхід завжди відображається на той самий вихід.

У MVX архітектурі через свою імперативність не можливо сказати з високим ступенем упевненості, що враховуючи однакові дані з сервера, програма буде показувати те саме відображення. Це пов'язано з тим, що потенційно деякі частини відображення можуть бути оновлені без наявності контролю над ними, такі ситуації зазвичай мають назву Сайд-ефекти.

Вирішити ці проблеми у складних інтерфейсних програмах дозволяє концепція односпрямованого потоку даних.

Поширеною проблемою у застосуванні MVX і чистої архітектури є те, що неоптимальна початкова точка (незавершені шаблони, відсутність еталонних реалізацій) призвела до різноманітних втілень із іноді суперечливими, а іноді навіть неправильними інтерпретаціями.

### **3. ЗАГАЛЬНІ ПРИНЦИПИ ПРО ПОБУДОВІ АРХІТЕКТУРИ МОБІЛЬНОГО ДОДАТКА**

#### **Чиста архітектура**

Clean Architecture пропонує використовувати чотири шари в стилі цибулі: об'єкти, варіанти використання, інтерфейсні адаптери, фреймворки та драйвери. [6] (Рис. 1)

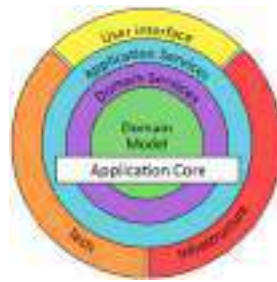


Рисунок 1. Clean Architecture [7]

Його основними цілями є незалежність від фреймворків і зовнішніх агентств (інтерфейс користувача, бази даних тощо) і можливість тестування. Ці цілі корисні для всіх видів програмного забезпечення, але вони особливо корисні для мобільних програм. Ключовою частиною досягнення цього є використання принципу інверсії залежностей для перетину меж кола. Класи внутрішнього циклу можуть спілкуватися з класами зовнішнього циклу лише через інтерфейси, реалізовані класами зовнішнього циклу (Рис. 2). [8]

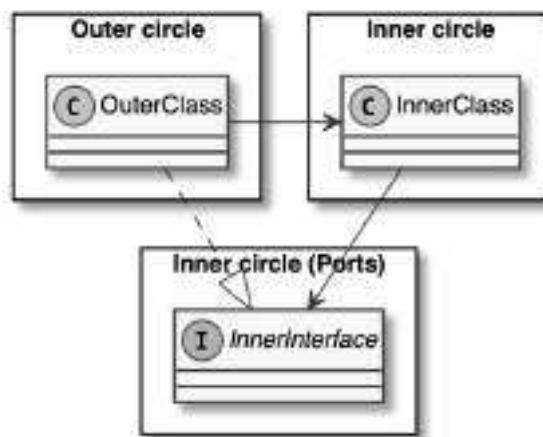


Рисунок 2. Спілкування класів через інтерфейси [8]

Однією з головних проблем чистої архітектури є відсутність офіційних еталонних реалізацій для різних платформ (наприклад, у книгах Мартіна, його блозі, його навчальних відео та обліковому записі GitHub).

Як наслідок, приклади Clean Architecture для iOS та Android, як правило, дають багато вказівок щодо конкретної платформи, часто жертвуючи незалежністю та стійкістю. Корисно повторити, що MVX — це шаблони високого рівня, які включають шаблони проектування. Чиста архітектура, також включає, принципи проектування, принципи компонентів, концепцію шарів і концепцію меж, серед інших ідей. Проект мобільного додатку з чистою глобальною архітектурою не повинен починатися з неофіційного шаблону, який уже містить низку залежних від платформи фреймворків; він повинен використовувати всі концепції, які є корисними для мобільної розробки, таким чином, щоб залишатися слабким зв'язком з платформою.

### Патерни проектування

Говорячи про шаблони високого рівня та архітектурні концепції, ми не повинні забувати про шаблони проектування, такі як шаблони GoF, які, звичайно, також можна використовувати та інтегрувати незалежно.

## **Принципи SOLID**

Чиста архітектура підтримує добре відомі принципи дизайну SOLID [7]. Оскільки вони сприяють досягненню загальних архітектурних цілей Clean Architecture, вони, природно, також актуальні для мобільних додатків.

### **Контроль інтерфейсу сутності**

Для реалізації цього принципу було запропоновано структурувати системи з трьома типами об'єктів. Спочатку вибрані типи Entity-Interface-Control пізніше були змінені на Entity-Boundary-Control, а потім на Entity-Boundary-Interactor. Об'єкти сутності містять дані, які використовує система, і всю поведінку, природно пов'язану з цими даними. Граничні об'єкти моделюють інтерфейс із системою [8]. Уся поведінка, що залишилася, поміщається в об'єкти Interactor. Martin's Clean Architecture адаптує ці типи об'єктів. Граничні об'єкти, зокрема, сприяють роз'єднанню, а отже, незалежності та тестуванню. Однак ЕВІ не дуже чітко пояснює, як отримати незалежність не тільки від інтерфейсу користувача.

### **Впровадження залежностей**

Впровадження залежностей допомагає підтримувати незалежність компонентів і їх тестування. Однак це не обов'язково означає, що потрібна структура впровадження залежностей. Може бути достатньо розглянути, які залежності мають бути гнучкими і організувати їх у прозорий спосіб, який легко змінювати.

## **4. ЗАГАЛЬНІ ВИМОГИ ДО ПОБУДОВИ АРХІТЕКТУРИ МОБІЛЬНОГО ЗАСТОСУНКУ**

Побудова мобільних додатків не завжди простий процес, оскільки мобільний пристрій має доволі багато апаратних та прикладних обмежень. Тому при побудові треба враховувати такі моменти:

- великий обсяг UI коду;
- короткий реліз-цикл розробки програмного забезпечення;
- велика кількість різних сенсорів, датчиків у смартфоні;
- збереження даних, їх синхронізація, а також підтримка автономного режиму роботи;
- концепція програмування на життєвому циклі застосунка
- код має бути ефективний, з огляду на обмеження по пам'яті та заряд батареї смартфона;
- багатопоточність з особливістю того, що оновлення відображення зазвичай трапляється тільки в спеціальному потоці;

Наданий список може бути не вичерпаним, оскільки при розробці будь-якого мобільного застосунку треба звертати увагу на вимоги бізнесу.

Наступним кроком слід зазначити, що будь-яка гарна та підтримувана архітектура має на меті зберігати такі принципи: незалежність між своїми компонентами та фреймворками, а також можливість тестування.

*Незалежність:* незалежність між компонентами та фреймворками необхідна для побудови ефективної архітектури. Оскільки це може допомогти зменшити прогалину між апаратним забезпеченням та виконанням програми, допомагаючи модульними тестами. Маючи більш гнучку архітектуру з'являється можливість оновлювати застосунок за короткі проміжки циклу випуску з різноманітністю апаратного та програмного забезпечення та потребами налаштування.

*Автоматизоване тестування:* існує багато аспектів, які ускладнюють автоматизоване тестування для мобільних застосунків. В розрізі мобільних застосунків ми можемо поділити автоматизоване тестування на чотири види: локальні модульні тести (запускаються на приладах для розробки), модульні тести на ізольованих машинах/смартфонах, UI тести, а

також Snapshot тести. Серед трьох видів тестів слід відокремити модульні тести, бо вони надають нам швидких зворотній зв'язок, тому гарні архітектурні шаблони мають підтримувати модульні тести.

## 5. КОМБІНОВАНА АРХІТЕКТУРА

З огляду на існуючі архітектурні рішення пропонується створити комбіновану архітектуру: а саме поєднання класичних двоспрямованих архітектур MV(X) або Чистої архітектури з UDF (односпрямованою архітектурою). Тобто остаточний вибір MV(X) або різних реалізацій Чистої архітектури залежить від вимог зі сторони бізнесу, а також досвіду команди розробки.

Для цього пропонується створити DSL навколо імперативних підходів побудови компонентів відображення. В більшості випадках такий підхід виглядає більш привабливим в порівнянні з побудовою інтерфейсу у імперативному стилі або використанням XML конструкторів. Привабливість забезпечується через мінімізацію змінності даних, що покращує безпеку програми, оскільки незмінні структури даних у багатьох випадках менш схильні до помилок. Також пропонується побудова слою абстракцій, які будуть допомагати поєднувати UDF базовані архітектури з класичними MV(X). Це необхідно для того, щоб в залежності від поставлених вимог створювати той, чи інший модуль в потрібній архітектурі. Також при побудові даної архітектури треба оптимізувати роботу UDF архітектур, де можуть виникнути проблеми з продуктивністю. Для цього пропонується використовувати композицію Стану з під-станів, а також використовувати можливість порівняння одноманітності об'єктів Станів, та оновлювати відображення в разі нерівності теперішнього і нового стану.

Очевидно, з першу необхідно визначити модулі загального призначення які не прив'язані до конкретної програми і можуть бути повторно використані будь-де. Такі модулі використовуються верхнім рівнем програми, і хоча вони також можуть мати залежність один від одного, але це рідкісний випадок. Тому цей рівень називається App Independent (Рис. 3).

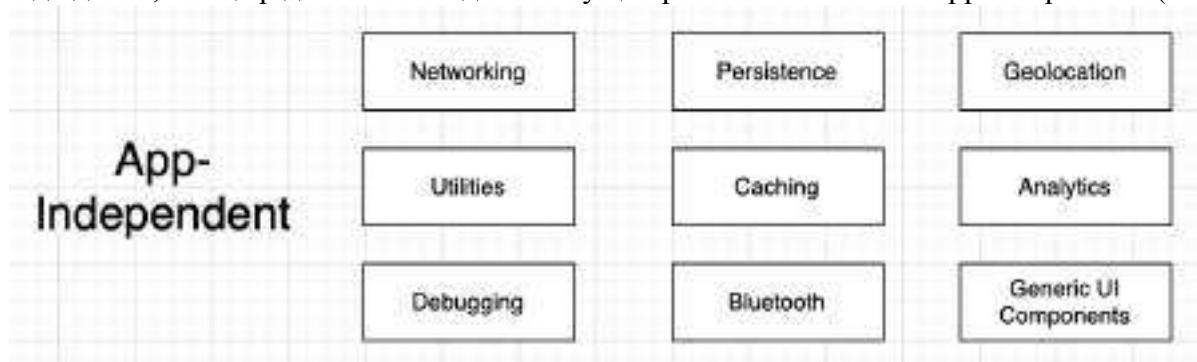


Рисунок 3. App-Independent модуль

Наступним кроком при проектуванні архітектури є створення окремих модулів функціоналу застосунку. Зазвичай можна побачити наскільки тісно окремі модулі функціоналу можуть залежати один від одного, що вказує на недоліки та слабкі сторони архітектури. Тому, основне правило, якого повинні дотримуватися розробники під час роботи над цими модулями — відсутність горизонтальної залежності на рівні функцій. Це означає, що один модуль функцій не повинен залежати від іншого. Архітектура кожного модуля може відрізнятися, тобто кожний модуль може бути реалізований ізольовано як і на UDF-подібній архітектурі, так і на MVX.



Рисунок 4. Features модуль

Наступним кроком розробки архітектури за стосунку є створення компонентів, які з одного боку, належать до рівня Features, а з іншого — не належать до жодного конкретного модуля. Таким чином, пропонується створити ще один рівень, App Specific для застосунку (Рис. 5).

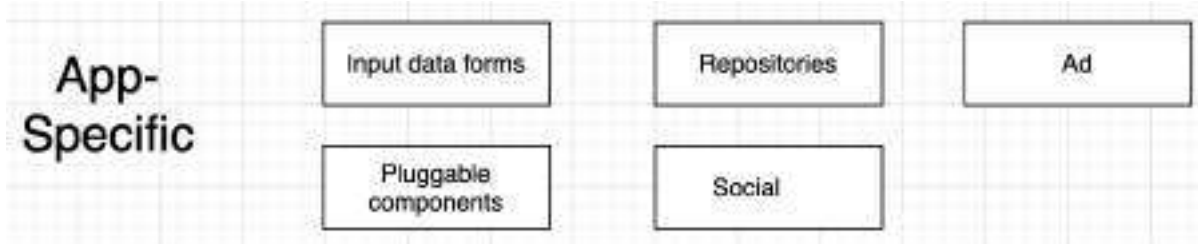


Рисунок 3. App-Specific модуль

Host App – це верхній рівень архітектури. Цей рівень визначає стан програми та тип конфігурації. Він отримує сповіщення від ОС. (Рис. 6).

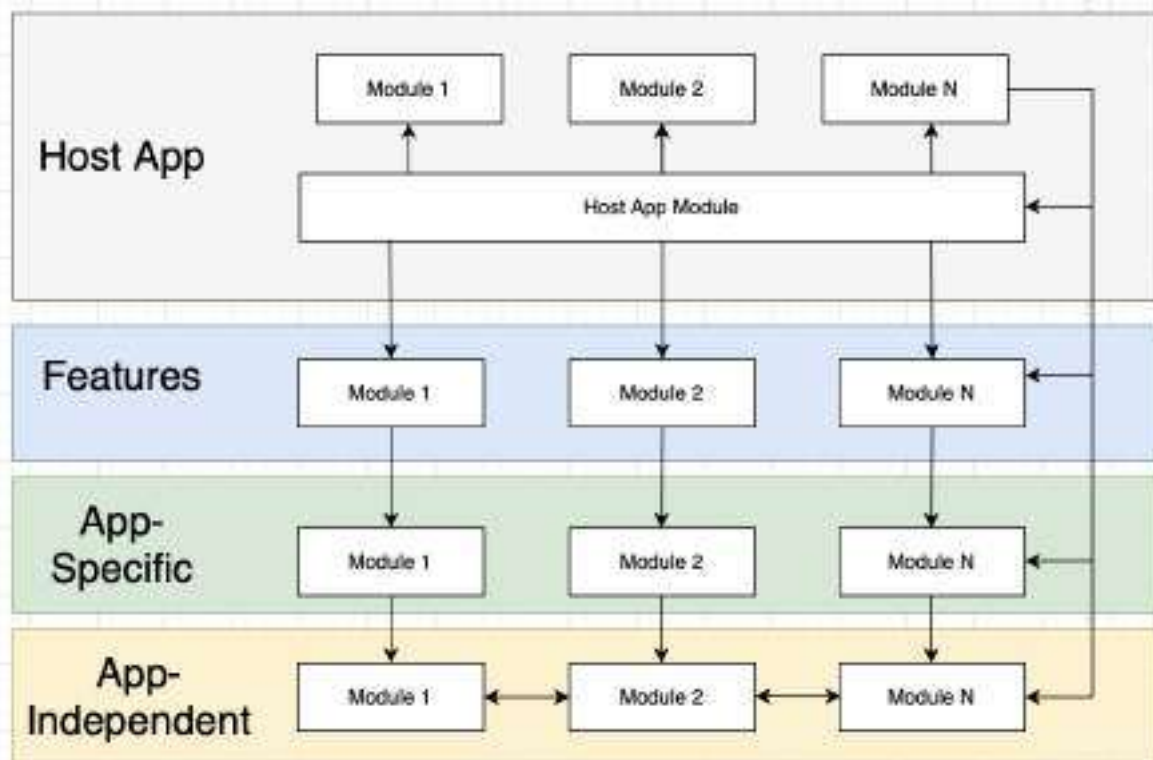


Рисунок 6. Загальний вигляд запропонованої архітектури

Таким чином, Host App є єдиним рівнем, який відповідає за навігацію між функціями, конфігурацію функцій і зв'язок між ними. Щоб забезпечити низький зв'язок модулів, ми повинні дотримуватися принципу інверсії залежностей, щоб в разі підвищити придатність до відладки та тестування. Такі залежні модулі, як репозиторії, служби та аналітика, мають

бути введені в модулі функцій за допомогою інтерфейсу з необхідними параметрами та налаштуваннями на рівні програми App Host.

Для здійснення переходів між функціями було запропоновано додати сутність Coordinator, яка належить кожному компоненту Features. Реалізація кожного Coordinator створюється в Host App модулі застосунку, який відповідає інтерфейсу навігації для певного модулю Feature. Цей підхід забезпечує детермінованість порядку показу екранів та логіку переходів. Приклад представлення збірки модуля «Список продуктів» і навігації від цієї функції до функції «Відомості про продукт» за допомогою Coordinator (Рис. 7). Пунктирними лініями відображена композиція, а суцільними відображено взаємодію модулів між собою, тобто перехід з відображення «Список продуктів» до відображення «Відомості про продукт».

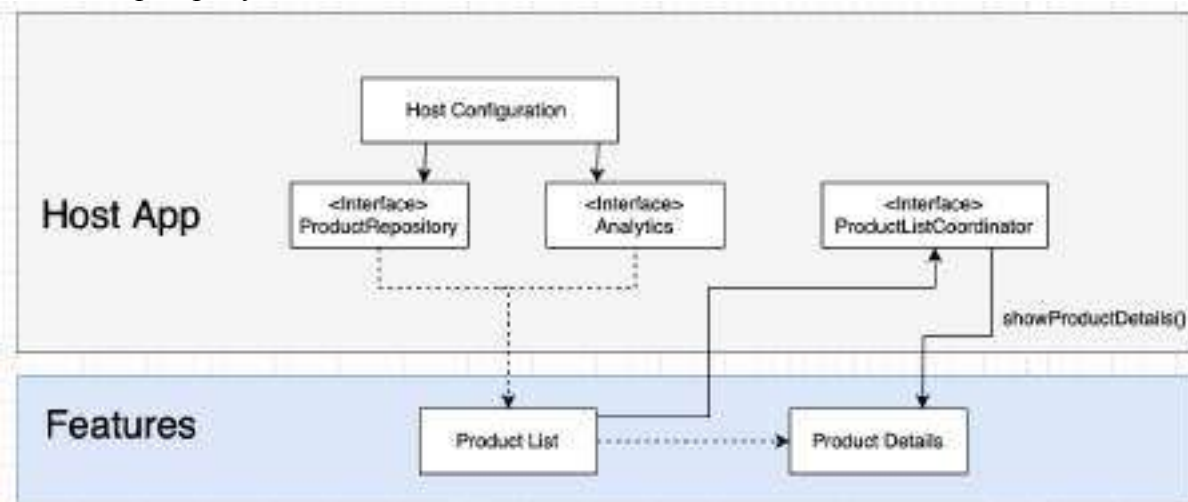


Рисунок 7. Взаємодія Coordinator з модулями функціональності застосунку

Також у кожного модуля Feature буде існувати LocalState, цей локальний Стан модифікує виключно керуючий модуль (Reducer, Presenter, ViewModel) батьківського Store, після модифікації загального AppState, LocalStore за допомогою механізму підписки отримує оновлений LocalState та зберігає його. Наступним кроком, усі підписники LocalStore отримують оновлений LocalState.

## 6. ВИСНОВКИ

З огляду на проведений аналіз можливо стверджувати, що популярні сучасні архітектурні підходи, такі як Clean, Onion необхідні для складних мобільних додатків, оскільки вони забезпечують незалежність від зовнішніх фреймворків і можливість тестування. Однак відсутність стандартних реалізацій із гарантованою якістю для iOS і Android призводить до різноманітних реалізацій і прикладів різної якості. Тому в даній роботі було запропоновано перенести їх принципи для побудови комбінованої архітектури, яка б дозволила поєднати їх переваги. На основі UDF подібних архітектур було запропоновано декомпонувати єдиний глобальний Стан на під-стани, а також створити порівняння цих станів, аби уникнути випадків оновлення відображення, коли це не потрібно.

## ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. R. C. Martin, “Clean Architecture. A Craftsman’s Guide to Software Structure and Design”. Englewood Cliffs, NJ: Prentice Hall, 2017
2. Apple Computer, Inc., “Cocoa fundamentals guide,” 2006.
3. Архітектурний шаблон MVP. URL: <https://en.wikipedia.org/wiki/Model-view-presenter> (дата звернення: 16.11.2022)
4. Архітектура MVVM. URL: <https://jeremybytes.blogspot.com/2012/04/overview-of-mvvm-design-pattern.html> (дата звернення: 17.11.2022)
5. UDF Архітектура. URL: <https://habr.com/ru/company/inDrive/blog/571394/> (дата звернення: 18.11.2022)
6. J. Palermo, “Onion architecture”, 2008. URL: <https://jeffreypalermo.com/2008/07/the-onion-architecture-part-1/> (дата звернення: 18.11.2022)
7. R. Martin and M. Martin, Agile Principles, “Patterns, and Practices in C#”, ser. Robert C. Martin Series. Pearson Education, 2006.
8. I. Jacobson, M. Christerson, P. Jonsson, and G. Overgaard, “Object- Oriented Software Engineering”. New York, NY, USA: ACM, 1992.

# **ПРЕДСТАВЛЕННЯ ІНТЕРОПЕРАБЕЛЬНИХ СЕРВІСІВ ДЛЯ РОБОТИ З МЕДИЧНИМИ ДАНИМИ В МІКРОСЕРВІСНІЙ АРХІТЕКТУРІ**

Кандель К.В.<sup>1</sup>, Кулик В.О.<sup>2</sup>, Мельник О.Р.<sup>3</sup>, Письменний І.О.<sup>4</sup>

Національний технічний університет України  
«Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна

<sup>1</sup> kirill.kandel@gmail.com [0000-0001-7152-5945], <sup>2</sup> slava.kulik27@gmail.com [0000-0003-3833-1529], <sup>3</sup> melnyk.oleksii@lil.kpi.ua [0000-0002-0297-6635],  
<sup>4</sup> ihor.pismennyu@gmail.com [0000-0001-7648-2593]

**Розглядається задача аналізу проблематики архітектурних підходів при проектуванні систем медичних сервісів. Досліджено наявні недоліки при побудові та впровадженні механізмів обробки та взаємодії з медичною інформацією. Завдяки порівнянню архітектурних підходів, буде спроектовано та запропоновано мікросервісну архітектуру взаємодії інтероперабельних медичних сервісів.**

**Ключові слова:** медичні сервіси, мікросервісна архітектура, інтероперабельні сервіси, медичні стандарти.

## **1. ВСТУП**

За останні десятиліття галузь охорони здоров'я досягла багатьох вагомих успіхів. Досягнення генетики та біотехнології дозволили вченим зрозуміти та лікувати хвороби на молекулярному рівні. Мабуть, найбільш вражаючим є той факт, що за останні 50 років тривалість життя зросла на 40%[1]. Незважаючи на ці успіхи, все ще є величезні можливості для вдосконалення. Найголовнішим серед цих можливостей є перехід до інформаційно-орієнтованої медицини, моделі індивідуальної медичної допомоги, за якої пацієнти отримують персоналізовані цільові рішення для лікування, специфічні для їхніх індивідуальних стадій захворювання, а також їхніх індивідуальних генетичних і метаболічних параметрів. Такий перехід, насамперед, супроводжується значними удосконаленнями в моделях взаємодії медичних сервісів - видобутку та обробці медичної інформації про пацієнта, її передача та зберігання у медичних інформаційних сервісах.

Медична сфера вважається однією з найскладніших сфер для досліджень. Найбільш поширені підходи зосереджені на забезпеченні надійних систем класифікації, які використовуються для допомоги лікарю в процесі діагностики/прогнозування. Проблеми аналізу медичних даних мають деякі унікальні особливості, які відрізняють їх від інших сфер і ускладнюють їх вирішення. Основні труднощі пов'язані зі складним характером оброблюваних даних, їх якістю і кількістю.

Хоча в лікарнях зберігається величезна кількість інформації, що відноситься до пацієнтів, які лікувалися раніше, частина цих даних не в електронному вигляді, і ідея їх цифровізації зазвичай розглядається як трудомістка. Крім того, зазвичай лікарі в різних лікарнях мають дещо різні методи дослідження. Це призводить до різних структур даних, що надходять із різних джерел, що у більшості випадків унеможливує їх поєднання. Крім того, оскільки на кону стоїть людське життя, точний діагноз має вирішальне значення. Тому

стандартизація є дуже важливим кроком, що дозволить суттєво спростити механізми передачі та взаємної обробки інформації у медичних сервісах.

Одним з поширених проблемних питань у інформатизації медицини є постійний процес виникнення та застосування нових типів медичної інформації - стрімкий розвиток та впровадження інформаційних систем у галузь охорони здоров'я призводить до більш детальних та різноманітних потоків інформації, що потребують обробки. Як результат, безліч компаній, що розробляють інформаційні системи та сервіси для роботи з медичною інформацією, зіштовхуються з проблемою надзвичайної складності та комплексності при розробці, підтримці та оновленні програмного забезпечення.

## 2. ПОСТАНОВКА ПРОБЛЕМИ

Вагомою проблемою, що постає перед розробниками інформаційних систем взаємодії медичних сервісів, є потреба у комбінуванні можливостей отримання даних про пацієнта або медичні засоби та можливостей їх обробки та транспортування. Кожна потенційна компанія-розробник оперує лише власним переліком сервісних можливостей, що можуть не включати логістичні сервіси доставки медикаментів або, наприклад, сервіси своєчасного зчитування та зберігання інформації про пацієнта. Натомість, зосереджуючи розробку на обмеженому переліку потенційних сервісів, розробляється архітектура системи, що не може забезпечити повноцінний обіг інформації про пацієнта, ліки та засоби лікування, а також не надає можливості керувати даною інформацією та перебігом подій, пов'язаних з пацієнтом чи ліками.

Іншою проблемою закритих інформаційних медичних систем, окрім неповноцінного інтерфейсу взаємодії з даними та інструментами, є обмежена можливість масштабування. Архітектура системи, що з самого початку була побудована монолітно, є надзвичайно вразливою до проблем потенційного горизонтального масштабування, якщо виникне потреба. Причиною для розширення функціоналу може слугувати будь-що: збільшення джерел надходження інформації чи кількості медичних сервісів, що надає розробник - кожен з пунктів може потребувати розробки окремого сервісу для забезпечення функціонування, тому можливість горизонтального масштабування у архітектурі системи є вкрай необхідною.

Слід також відзначити проблему незалежної розробки сервісів при обробці інформації чи впровадженні певного функціоналу. Так, наприклад, компанія-розробник може створювати певний набір медичних інформаційних сервісів, що незалежно один від одного обробляють інформацію та не мають автоматизованих засобів збереження та передачі даних між собою. Це призводить до наступних проблем:

- 1) Зменшення швидкодії повного життєвого циклу медичної інформації через відсутність єдиної автоматизованої системи
- 2) Необхідність залучення спеціалістів для керування та обміну даних між сервісами
- 3) Проблематика обробки різних форматів даних у сервісах, що не мають єдиного стандарту обміну інформацією
- 4) Проблема безпеки даних пацієнта та засобів лікування - необхідність забезпечувати конфіденційність на рівні кожного сервісу

Вирішенням даного ряду проблем може стати мікросервісна архітектура, що забезпечує повноцінну можливість оперування даними пацієнта, знаходити, обробляти та зберігати медичну інформацію. Такий підхід дозволить забезпечити гнучкість при масштабуванні, дозволивши створити необхідний перелік медичних сервісів, а також впровадити засоби їх взаємодії.

Цей матеріал має за мету запропонувати вирішення проблеми обробки медичної інформації масштабованою системою сервісів, що може бути впроваджена та вдосконалена

за потреби у разі виникнення нових технічних вимог відносно типів інформації та алгоритмів взаємодії медичних сервісів.

### 3. МЕТА ТА ЗАДАЧІ ДОСЛІДЖЕННЯ

Медичні дані пацієнта є критичними, чутливими та конфіденційними даними, оскільки є частиною медичної таємниці. Такі медичні дані, як медична картка пацієнта, діагнози, курси лікування і т.д. є важливими як для пацієнта, у разі невірної лікування, так і для страхових компаній, які повинні сплачувати страхову компенсацію. Тому ці медичні дані напрочуд важливі і їх конфіденційність та захищеність на всіх етапах використання є важливою частиною функціонування медичної установи.

Враховуючи те, що сучасні медичні системи постійно розвиваються і створюють архітектури обчислювальних систем та інфраструктури їх взаємодії, постає нагальна проблема до забезпечення безпеки даних в розподілених медичних системах, що включають в себе інтегровані медичні сервіси. Також, провайдери медичних сервісів будують власні архітектури медичних сервісів базуючись на різних підходах, у тому числі й на монолітному підході побудови обчислювальної системи. Попри те, що у монолітних систем є свої переваги і недоліки [2], варто зазначити, що сучасні архітектурні рішення будуються за допомогою мікросервісного підходу або взагалі без залучення серверів - використовуючи хмарні платформи [3]. Тобто у результаті ми отримуємо значну кількість однакових за суттю систем, які побудовані різноманітним чином.

Тобто, як було сказано вище, можна виокремити мету роботи. Метою є представлення певної уніфікованої архітектури інтегрованих сервісів для роботи з медичними даними.

Взявши до уваги варіативність існуючих обчислювальних систем та архітектур, потреб сучасної медицини та медицини наступного дня, запропонована архітектура повинна мати можливість для розширення, стабільно виконувати роботу при навантаженнях і мати повний контроль над конфіденційними даними, що є у системі без можливості їх пошкодження чи витоку. Таким чином, нам необхідно дослідити сучасні архітектурні підходи побудови таких систем і запропонувати ту, яка буде найбільш гнучкою під описані потреби.

Через значні регуляції та обмеження в сфері охорони здоров'я, багато країн мають свої сертифікаційні процедури, у тому числі й для електронних медичних систем, що накладають свої обмеження на безпеку, цілісність і конфіденційність даних в медичній системі [3]. Тобто запропонована архітектура повинна враховувати можливі рестрикції в обробці, передачі, збереженню чутливих медичних даних, що необхідно передбачити, враховуючи існуючі стандарти.

Таким чином, буде представлено структуру взаємозв'язків інтегрованих сервісів для роботи з медичними даними. Головними перевагами отриманої системи будуть наступні характеристики:

1. Безпека і конфіденційність даних при зчитуванні, передачі та взаємодії на всіх рівнях.
2. Можливість масштабування при зміні навантаження на систему.
3. Уніфікація архітектурного рішення, для спрощення роботи профільним спеціалістам та підтримки гнучкості системи.
4. Спрощена інтеграція з існуючою системою новітніх сервісів, завдяки поширеного стандарту обміну медичними даними.
5. Розширення, тестування і підтримка сервісів та їх кодової бази буде зручнішою завдяки уніфікації взаємодії та архітектурному підходу.

Такі конкурентні переваги будуть значним корисним впливом на загальний підхід до культури побудови розподілених систем, що включають в себе значні безпекові обмеження та можливості динамічної зміни самої системи на рівні її компонентів.

## 4. РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ

Обираючи стандарти для передачі і документування медичних даних, було розглянуто наступні:

1. Health Level 7 (HL7) - є сімейством міжнародних стандартів для передачі клінічних та адміністративних даних між програмними додатками, що можуть використовуватися різними провайдерами медичних сервісів. Є поширеним міжнародним стандартом, що постійно розвивається і використовується в різних електронних медичних сервісах. Значна кількість використань цього стандарту на науковому ресурсі говорить про його значну популярність [6].
2. Continuity of Care Record (CCR) - це стандарт медичної документації, був створений медичними працівниками на основі їхніх поглядів на дані, якими вони можуть захотіти поділитися в будь-якій ситуації. Значна, але менша за HL7, кількість використань цього стандарту на науковому ресурсі також говорить про його значну поширеність [5].
3. ISO/TC 215 - це стандарт для медичних інформаційно-комунікаційних технологій (ICT), щоб забезпечити сумісність і взаємодію між незалежними системами. Значна, але менша за HL7 та CCR, кількість використань цього стандарту на науковому ресурсі також говорить про його поширеність [7].

Розглядаючи ці стандарти, звернемо увагу на їх частоту використання у наукових роботах, також за тим, наскільки уніфікованим є стандарт і наскільки він є зручним для імплементації та розуміння. Виходячи з цих критеріїв, HL7 є розповсюдженим сімейством стандартів, які підійдуть для імплементації передачі, взаємодії (Рисунок 1) та збереження електронних карток пацієнтів між різними програмними сервісами та додатками.



Рисунок 1. Опис сімейства HL7

Розглядаючи різні типи архітектур, необхідно керуватися предметно-орієнтованим підходом для розробки такої розподіленої системи. Беручи до уваги те, що медичні системи повинні бути гнучкими і відмовостійкими, то монолітна архітектура не може конкурувати з мікросервісною архітектурою через свою природу. Приведемо порівняльну таблицю (табл. 1) цих архітектур і зможемо дізнатися, яка краще підійде для сучасної архітектури медичних інтероперабельних сервісів.

Таблиця 1. Порівняльна характеристика монолітної та мікросервісної архітектури

Монолітна архітектура	Мікросервісна архітектура
Висока зв'язаність компонентів архітектури	Низька зв'язаність компонентів архітектури
Важкість внесення змін до системи	Спрощене внесення змін
Складність масштабування	Простіше масштабування
Технологічний бар'єр та важча підтримка з часом	Кожен мікросервіс простіше перевести на нові технології без зміни інших частин системи
Простий процес розгортки додатку	Ускладнена розгортка додатку

Роблячи проміжний висновок у тому, що більшу гнучкість та кращу сумісність надає нам саме мікросервісна архітектура, оскільки можливість горизонтального масштабування є однією з головних переваг, а розподілення відповідальності між мікросервісами призводить до природної відсутності зчеплення між мікросервісами, якщо використовувати предметно-орієнтований підхід до усієї розробки такої системи, що важче уникнути в монолітних архітектурах.

Також, важливим елементом мікросервісної архітектури є уніфікованість прикладного програмного інтерфейсу кожного мікросервісу. Це робиться за допомогою специфікації OpenAPI, що дозволяє зручно описувати та документувати API кожного мікросервісу для його подальшого використання. Таким чином у поєднанні зі стандартом HL7 та предметно-орієнтованим підходом, можна провести уніфікацію API [8].

Також треба згадати про важливі елементи мікросервісної архітектури, такі як:

1. API Gateway - це інтерфейс між клієнтами та іншими сервісами, що може їх викликати для бажаного клієнту результату [9].
2. Service Discovery - це сервіс, автоматично виявляє та реєструє стан сервісів у комп'ютерній мережі [10].

Використовуючи ці сервіси, можна побудувати базову інфраструктуру для мікросервісної архітектури. На Рис. 2 зображена базова мікросервісна архітектура.

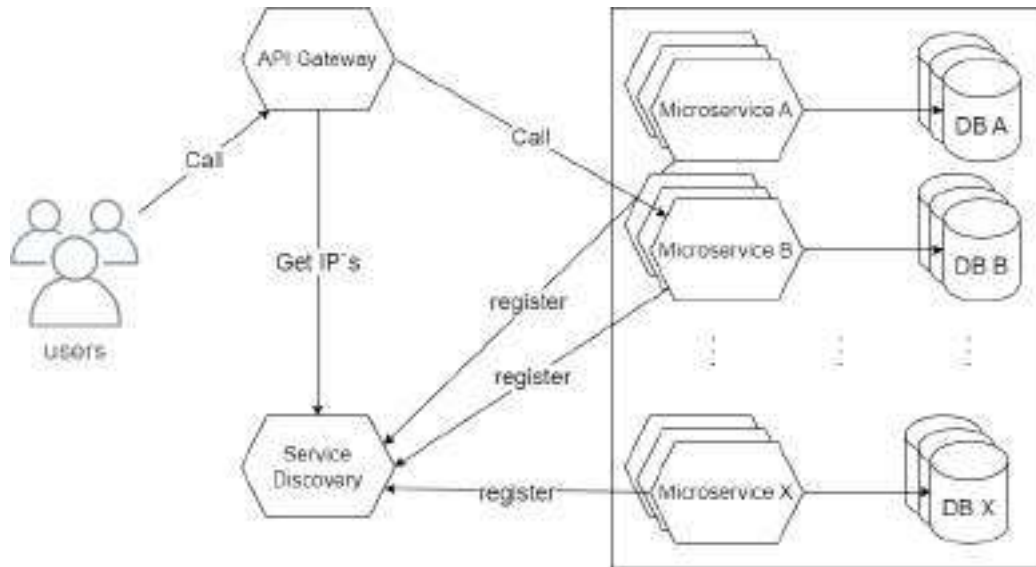


Рисунок 2. Базова мікросервісна архітектура

Використовуючи схожий підхід можна розширити його до медичної області застосування. На Рисунок 3 зображена мікросервісна архітектура для інтероперабельних медичних сервісів із обов'язковими та додатковими мікросервісами, та з прикладом виклику деякого мікросервісу.

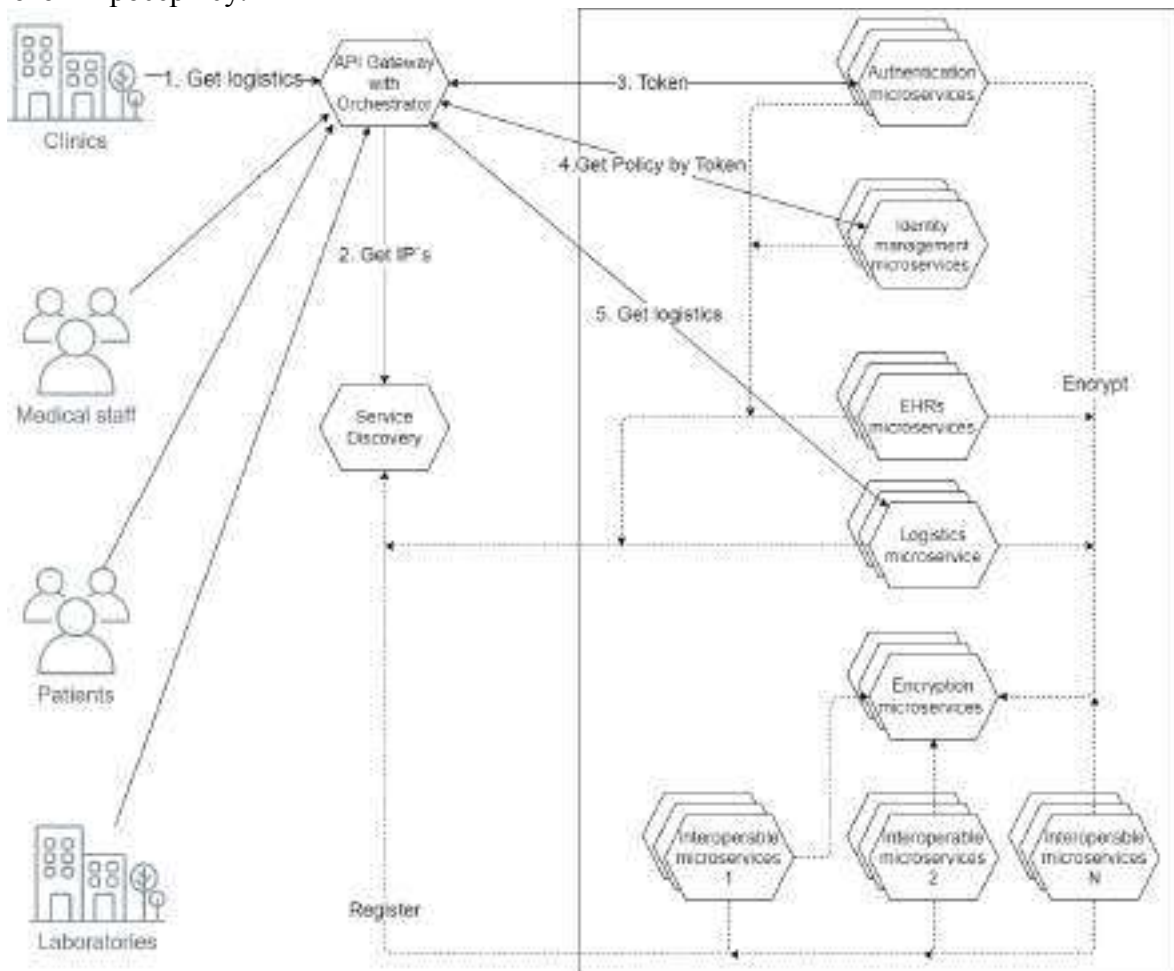


Рисунок 3. Мікросервісна архітектура медичних сервісів

## 5. ВИСНОВКИ

На сьогоднішній день розробники програмного забезпечення в галузі охорони здоров'я все більше починають стикатися з проблемами масштабування та додавання нового функціоналу в існуючі системи. Перш за все це пов'язано з тим, що компоненти системи є тісно зв'язаними. Для вирішення цих проблем пропонується застосовувати мікросервісну архітектуру.

Мікросервісна архітектура - це підхід у розробці програмного забезпечення шляхом створення незалежних компонентів - мікросервісів, які поєднуються між собою. Разом такі мікросервіси складають готовий додаток, але в будь-який момент часу він може бути розділений на незалежні блоки та інтегрований в іншу комп'ютерну систему без великих зусиль.

Використання мікросервісної архітектури в галузі охорони здоров'я надає такі переваги:

1. Портативність і сумісність. Мікросервісна архітектура використовує для взаємодії компонентів популярні стандарти JSON, HTML та ін., що дозволяє об'єднати в одному додатку мікросервіси, написані з використанням різних мов програмування. Це призводить до того, що кожен з мікросервісів є повністю незалежним та за необхідністю його можна перевикористати в іншому додатку. Це також означає, що додавання нових компонентів у існуючу систему здійснюється простіше та з меншим ризиком впливу на існуючі компоненти в процесі.
2. Ізоляція компонентів прискорює дотримання нормативних вимог. Одна з переваг, яка має особливе значення в галузі охорони здоров'я — це ізоляція між компонентами. Деякі компоненти є більш важливими за інші та потребують ретельного тестування, а також підпадають під регулятивний нагляд (наприклад, медичних стандартів). Мікросервісна архітектура дозволяє повністю ізолювати такі компоненти від інших компонентів системи, що значно спрощує їх тестування, оновлення та валідацію з існуючими стандартами.
3. Вбудована сумісність з нормами HIPAA. Мікросервіси за своїм походженням є подійно-орієнтованими та добре підходять для рішень, які вимагають відповідності нормам HIPAA, оскільки будь-яку важливу інформацію, наприклад особисту інформацію про пацієнта, можливо ізолювати від інших даних, а доступ до цих даних простіше контролювати.
4. Зменшений час розробки. Аспект «багаторазових компонентів» дозволяє не витратити час на розробку існуючого функціоналу, а зосередитись на додаванні нових компонентів до системи, що значно пришвидшує розробку та впровадження нових додатків.

Також мікросервісна архітектура має уніфікований програмний інтерфейс, створений за допомогою специфікації OpenApi, що дозволяє зручно описувати та документувати API кожного мікросервісу для його подальшого використання. Поєднавши даний інтерфейс з ключовими компонентами мікросервісної архітектури: Api Gateway та Service Discovery була розроблена модель архітектури інтероперабельних медичних сервісів.

### ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Medical informatization: A combination of digital technology and medical care [Електронний ресурс] – Режим доступу до ресурсу: <https://www.atcinitiative.org/medical-informatization-a-combination-of-digital-technology-and-medical-care>

2. A Comparative Review of Microservices and Monolithic Architectures [Електронний ресурс] – Режим доступу до ресурсу: <https://ieeexplore.ieee.org/abstract/document/8928192>

3. Microservices vs Serverless: A Performance Comparison on a Cloud-native Web Application [Электронный ресурс] – Режим доступа до ресурсу: [https://www.researchgate.net/publication/341483472\\_Microservices\\_vs\\_Serverless\\_A\\_Performance\\_Comparison\\_on\\_a\\_Cloud-native\\_Web\\_Application](https://www.researchgate.net/publication/341483472_Microservices_vs_Serverless_A_Performance_Comparison_on_a_Cloud-native_Web_Application)
4. Security Techniques for the Electronic Health Records [Электронный ресурс] – Режим доступа до ресурсу: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5522514/>
5. HL7 - Overview [Электронный ресурс] – Режим доступа до ресурсу: <https://www.hl7.org/>
6. Continuity of Care Record [Электронный ресурс] – Режим доступа до ресурсу: [https://en.wikipedia.org/wiki/Continuity\\_of\\_Care\\_Record](https://en.wikipedia.org/wiki/Continuity_of_Care_Record)
7. ISO/TC 215 Health informatics [Электронный ресурс] – Режим доступа до ресурсу: <https://www.iso.org/committee/54960.html>
8. OpenApi Initiative [Электронный ресурс] – Режим доступа до ресурсу: <https://www.openapis.org/>
9. What does an Api Gateway do? [Электронный ресурс] – Режим доступа до ресурсу: <https://www.redhat.com/en/topics/api/what-does-an-api-gateway-do>
10. Service discovery [Электронный ресурс] – Режим доступа до ресурсу: [https://en.wikipedia.org/wiki/Service\\_discovery](https://en.wikipedia.org/wiki/Service_discovery)

# МЕТОДИ РОЗПІЗНАВАННЯ ТЕКСТІВ ТА ПОШУКУ КЛЮЧОВИХ СЛІВ ДЛЯ АВТОМАТИЧНОГО РЕФЕРУВАННЯ ТЕКСТІВ

Кузнецов О.А.<sup>1</sup>, Кисельов Г.Д.<sup>2</sup>

НН ПСА КПІ ім. Ігоря Сікорського, Київ, Україна

<sup>1</sup>oleksiy1908@gmail.com, <sup>2</sup>g.kyselov@gmail.com

Метою дослідження є перевірка ефективності основних підходів до автоматичного реферування тексту, а також дослідження різних методів для різних підходів реферування тексту. У доповіді представлено опис основних підходів, які на даний момент використовують для автоматичного реферування тексту, а також різні методи, які використовуються в цих підходах. На думку автора, дослідження методів автоматичного реферування текстів дуже важлива, так як вони здатні суттєво прискорити та спростити цей досить важливий алгоритм.

**Ключові слова:** реферування текстів, добуваючий підхід, абстрактний підхід, метод тематичних слів, центрування речень, індикаторні представлення.

## 1. ОСНОВНІ ПІДХОДИ ДО АВТОМАТИЧНОГО РЕФЕРУВАННЯ ТЕКСТУ

Існує два основних підходи до автоматичного реферування тексту :

- Extractive summarization (так називається добуваючий підхід);
- Abstractive summarization (так називається абстрактний підхід).

Добуваючий підхід – це автоматичне анотування, що засноване на виділенні з первинних документів ключових фраз, слів, чи навіть цілих абзаців, які потім додаються в вихідний документ без змін та саме в порядку їх появи в первинному тексті.

Абстрактний підхід – це автоматичне анотування, що засноване на виділенні найбільш важливої інформації, а також навіть можливого створення нових текстів на базі узагальнення первинних документів. Цей метод, на відмінну від добуваючого методу дає можливість використовувати слова, яких не було у вхідному документі. Анотації, що були отримані використовуючи саме цей підхід схожі на ті, що люди пишуть самі. Але реалізація цього методу є дуже нетривіальною задачею, адже модель має вирішувати досить складні проблеми, такі як семантичне представлення тексту та генерація природної мови. Тому, зважаючи на складність даного підходу як у реалізації так і у рамках практичного застосування, а також значні обмеження які стосуються текстів, що можна реферувати використовуючи добуваючий метод надалі будуть проведені дослідження методів реферування в рамках добуваючого підходу.

## 2. ДОБУВАЮЧИЙ ПІДХІД АВТОМАТИЧНОГО РЕФЕРУВАННЯ ТЕКСТІВ

Зараз основа всіх добуваючих методів складається з трьох частин[1]:

– побудова проміжного представлення вхідного тексту. Існує два типи підходів до показу початкового тексту: topic representation (тематичне представлення) та indicator representation (індикаторне представлення). Тематичне представлення трансформує текст в

проміжне представлення за допомогою інтерпретацій тем, що представлені у тексті. Індикаторне представлення анує речення як перелік формальних ознак, тобто індикаторів, що відрізняються в залежності від алгоритму та використовує їх для ранжування тексту. До таких ознак відносяться довжина речення, місце в тексті, наявність ключових фраз і т.д.;

– оцінка речень на основі проміжного представлення вхідного тексту. При формуванні проміжного представлення кожному реченню присвоюється якась оцінка важливості цього речення для анотації. У різних підходах до подання теми оцінка речення відображає, наскільки добре те чи інше речення відображає найважливіші теми тексту;

– формування підсумку. Система вибирає декілька найважливіших речень для створення вихідної анотації. Деякі підходи для вибору важливих речень вибирають жадібні алгоритми, а деякі підходи можуть перетворити вибір речень у одну з задач оптимізації, де вибирається набір речень, враховуючи обмеження задачі оптимізації, на меті якого стоїть максимізувати важливість та мінімізувати надмірність деяких слів.

## 2.1. Підходи з використанням тематичного представлення

### Метод тематичних слів.

Техніка тематичних слів є загальним підходом, головною ідеєю якого є виділення слів, що описують тему вхідного тексту. Тематичні слова можуть бути визначені досить різними способами, наприклад робота[2] була однією з перших, в яких використовувався цей метод, пошук описових слів у документі відбувався за рахунок використання порогових значень частоти цих слів у тексті. У роботі[3] використовували логарифмічний алгоритм відношення правдоподібності для виявлення описових слів.

Визначення важливості речення базується на розподілі інформативних слів у реченнях. Існує два способи обчислити важливість речення: як функцію кількості тематичних слів, яку вона містить, або як частку тематичних слів у одному реченні. Обидві функції оцінки речень стосуються одного і того ж подання теми, однак вони можуть призначати різні оцінки різним реченням, через різну специфіку оцінювання. Перший метод може призначати вищі оцінки довшим реченням, оскільки вони мають більше слів, а інший більше звертає увагу на щільність тематичних слів. Найбільш поширеними методами в цій категорії є TF-IDF модель, лямбда-відношення правдоподібності та їх модифікації.

TF-IDF (term frequency — inverse document frequency, частота терму – зворотна частота документа) – статистична міра яка використовується щоб оцінити важливість терму у межах документу, який є частиною набору документів.

TF (term frequency – частота слова) – відношення кількості появи слова до загальної кількості слів у документі (1). Таким чином, оцінюється важливість терма  $t$  в рамках окремого документа  $d$ .

$$tf(t, d) = n_t / \sum_k n_k \quad (1)$$

де  $n_t$  – кількість входжень терма  $t$  в документ;

$\sum_k n_k$  – загальна кількість слів у документі.

IDF (inverse document frequency – зворотна частота документа) – інверсія частоти, з якою деяке слово зустрічається в документах колекції (2). IDF дозволяє зменшити оцінку широковживаних слів, та враховувати унікальність слів. Для кожного унікального слова в межах конкретного набору документів існує лише одне значення IDF. Більшу оцінку в TF-IDF отримують слова з високою частотою вживань в межах конкретного документа і з низькою частотою вживань в інших документах.

$$idf(t, D) = \log (|D| / |\{d_i \in D \mid t \in d_i\}|) \quad (2)$$

де  $|D|$  – кількість документів у колекції;

$|\{d_i \in D \mid t \in d_i\}|$  – кількість документів із колекції  $D$ , в яких зустрічається  $t$  (коли  $n_t \neq 0$ ).

$\lambda$ -відношення правдоподібності (log-likelihood ratio) є логарифмом відношення ймовірності спостереження слова з однаковою ймовірністю в корпусі вхідних документів і корпусі відповідних їм резюме, до ймовірності появи слова з різними ймовірностями в цих корпусах.

**Метод заснований на центруванні речень.**

Головна ідея цього підходу заснована на допущенні, що найбільш цікава для анатоції інформація знаходиться не лише в одному реченні. Він полягає в обчисленні «відстаней» між реченнями і у виборі тих з них, що в середньому знаходяться «ближче» до інших. Для визначення близькості речень використовуються алгоритми, що засновані на наборах слів (Bag-of-words). Наприклад найближчі в середньому речення можна визначити наступним чином:

- обчислити близькість між усіма парами речень за якимось параметром, наприклад перекриттям змісту одного речення іншим;
- для кожного речення визначити середню близькість до інших речень;
- впорядкувати ці значення і вибрати з мінімальну близькістю (наприклад використовуючи Латентний семантичний аналіз (LSA)).

Тематична модель, що представляється в роботі [4], це неконтрольований метод вилучення прихованої семантики тексту на того, що слова з подібним значенням частіше зустрічаються разом, чим окремо. Простіше кажучи LSA бере текстові документи та відтворює їх у декількох різних частинах, де кожна частина виражається у різний спосіб погляду на значення тексту. Якщо уявити текстові дані як ідею, існувало б декілька різних способів розгляду цієї ідеї або декілька різних способів адаптації усього тексту.

Метод, що представлений у роботі [5] полягає у виборі одного речення для кожної з тем (рисунок 1), таким чином, щоб зберегти початкову кількість тем. Ця стратегія в початковому вигляді має недолік через те, що темі може знадобитися більше одного речення для передачі своєї інформації, а ми вибираємо лише 1. Тому, були запропоновані декілька альтернативних рішень для поліпшення ефективності методів узагальнення на основі LSA. Одним із покращень було використання ваги кожної теми, що надало гнучкість у варіативності кількості речень. Інше поліпшення базується на тому, що його автори[6], зрозуміли, що речення, які обговорюють деякі важливі теми, є хорошими кандидатами для узагальнення теми, отже, для того, щоб знайти ці речення, вони визначили вагу речення за допомогою формули (3):

$$(s_i) = \sqrt{\sum_{j=1}^m d_{ij}^2} \tag{3}$$

де  $g()$  – функція «зважування»;  
 $m$  – кількість речень;  
 $d_{ij}$  – вага теми  $i$  в реченні  $j$ .

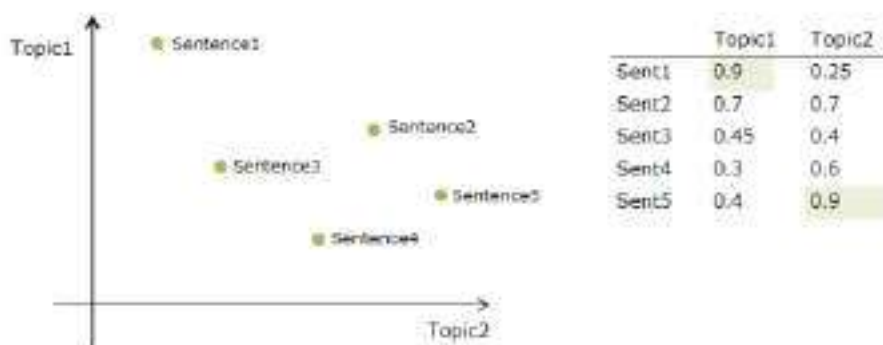


Рисунок 1. Простий вибір основних речень, що представляють теми документу (LSA)

## 2.2. Підходи з використанням індикаторного представлення

Підходи до подання показників (індикаторів) спрямовані на моделювання подання тексту у вигляді деякого набору ознак, які потім будуть використовуватися для класифікації.

До них відносяться методи на основі графів та техніки машинного навчання, що використовуються для визначення речень, які потрібно включити в кінцеве рев'ю. Основні використовуванні ознаки для цих методів:

- Речення на початку або в кінці тексту є більш інформативними;
- Занадто короткі або занадто довгі речення є малоінформативні;
- Наявність визначених сигнальних фраз або ключових;
- Слова з заголовку дають право вважати що це речення має ознаки відношення до цієї теми;
- Наявність емоційно забарвлених розділових знаків (знак питання, знак оклику, трикрапка, тощо);

Методи на основі графів, утворені під впливом алгоритму PageRank [7], тому вони представляють з себе документ, який був відображений у вигляді зв'язаного графу. Кожне речення утворює вершини графу, а вага ребр між реченнями вказують, на зв'язок подібності двох речень. Приклад подання тексту у вигляді графа можна побачити на рисунку 2. Подібність речень може бути виміряна як змістовним перекриття між реченнями, так і з використанням методу TF-IDF наприклад.

Графічна інтерпретація тексту має два основних результати. По-перше, кожен підграф – це окремий розділ, що пов'язаний однією. Другий результат – ідентифікація важливих речень у документі заснована на припущенні, що речення, які пов'язані з багатьма іншими реченнями, а також є центрами підграфів і, швидше за все, важливі і відображають тематику вхідного тексту, тому їх потрібно включити у вихідний документ.

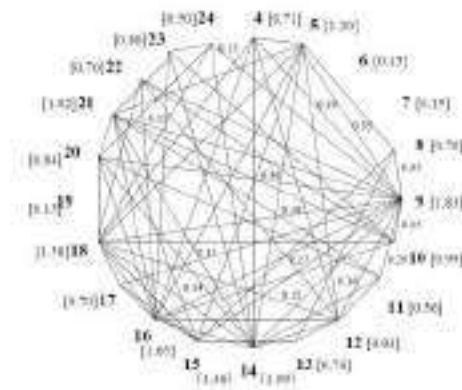


Рисунок 2. Графічне подання тексту у вигляді зваженого графу (речення представляють вершини з встановленою результуючою оцінкою)

Підходи машинного навчання представляють реферування як проблему класифікації. У роботі<sup>[8]</sup> представлена рання дослідницька спроба застосування технік машинного навчання для реферування. Автори розробили функцію класифікації, яка базувалася на наївному баєсовській класифікаторі. Ймовірності для класифікації були визначені за допомогою тренувальних даних з використанням правила Байеса (4):

$$P(s \in S | F_1, F_2, \dots, F_k) = P(F_1, F_2, \dots, F_k | s \in S) P(s \in S) / P(F_1, F_2, \dots, F_k), \quad (4)$$

де  $s$  – це речення із колекції документів;  $F_1, F_2, \dots, F_k$  – ознаки, що використовуються для класифікації;  $S$  – резюме, що має бути створено.

Отже, незважаючи на різноманітність методів та обґрунтовану дієвість кожного із них, кожен знає має досить суттєві недоліки. Наприклад, однією з основних проблем окрім складності деяких підходів є те, що для даних алгоритмів необхідна спеціальна вибірка, і не кожна випадкова вибірка документів згодиться для класифікації. Задача створення такої вибірки є більш складною, ніж створення цього всього власноруч, які містять об'єднані речення, переформульовані фрази чи навіть суто нові речення, до того ж такі навчальні дані також мають створюватися і робота алгоритму повністю залежить від якості таких навчальних даних. Якби вдалося б прибрати ці недоліки та обмеження, то можна було б скористатися даними алгоритмами для більш широкого спектру областей, але на даний момент вони накладають значні обмеження на тексти.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. A. Nenkova, K. McKeown. A survey of text summarization techniques.
2. Hans Peter Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of research and development* 2, 2 (1958), 159–165.
3. Ted Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational linguistics* 19, 1 (1993), 61–74.
4. Scott C. Deerwester, Susan T Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. *JASIS* 41, 6 (1990), 391–407
5. Yihong Gong and Xin Liu. 2001. Generic text summarization using relevance measure and latent semantic analysis. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 19–25
6. Josef Steinberger, Massimo Poesio, Mijail A Kabadjov, and Karel Ježek. 2007. Two uses of anaphora resolution in summarization. *Information Processing & Management* 43, 6 (2007), 1663–1680.
7. Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into texts. *Association for Computational Linguistics*.
8. Julian Kupiec, Jan Pedersen, and Francine Chen. 1995. A trainable document summarizer. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 68–73.

# ПІДТРИМКА ПРИЙНЯТТЯ РІШЕНЬ ЗА ДОПОМОГОЮ ЗАСОБІВ DECISION INTELLIGENCE

Макаров І.В.<sup>1</sup>, Кисельов Г.Д.<sup>2</sup>

Національний технічний університет України  
«Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна

<sup>1</sup> makarov@dlit.dp.ua

**Розглядається задача аналізу процесу використання засобів Decision Intelligence для підтримки прийняття рішень. Описано структуру технології та перелік переваг, які можна досягти завдяки впровадженню наявних інструментів. Розглянуто сфери застосування Decision Intelligence та описано алгоритмічну основу технології, а саме сутність глибокого навчання з підкріпленням.**

**Ключові слова: Decision Intelligence, глибоке навчання з підкріпленням, машинне навчання, Data Science.**

## 1. ВСТУП

Decision intelligence (інтелект при прийнятті рішень) — це нова сфера, яка допомагає підтримувати, доповнювати та автоматизувати бізнес-рішення, пов'язуючи дані з рішеннями та результатами. Він використовує поєднання методів (наприклад, відображення рішень і теорій прийняття рішень) і технологій (наприклад, машинне навчання та автоматизація) для покращення способів прийняття рішень у компаніях. Розвідка про прийняття рішень включає постійну оцінку результатів рішень та їх оптимізацію за допомогою системи зворотного зв'язку.

Концепція ґрунтується на ідеї, що компанії повинні ставитися до прийняття рішень як до інших сучасних бізнес-процесів і таким чином впроваджувати системи запису для моделювання, відстеження, навчання та вдосконалення конкретних управлінських рішень. Цей підхід починається з найважливіших рішень, які сприяють ефективності компанії, а потім повертається до людей, процесів і ідей, необхідних для прийняття послідовних високоякісних рішень.

## 2. РІВНІ DECISION INTELLIGENCE

Є три рівні, на яких DI може підтримувати бізнес-рішення.

Перший рівень — це підтримка прийняття рішень, на якому машини надають деякі основні інструменти для підтримки прийняття рішень людиною, такі як сповіщення, аналітика та дослідження даних. Самі рішення приймаються виключно людьми.

Другий рівень — це розширення прийняття рішень, на якому машини відіграють більшу та активнішу роль у процесі прийняття рішень. Вони аналізують дані та генерують рекомендації та прогнози для осіб, які приймають рішення, для перегляду та підтвердження. Наприклад, вони можуть дати рекомендацію на кшталт: «Вам слід купити 200 продуктів у постачальника А до 30 березня; це дозволить вам заощадити 20 000 доларів США». Люди можуть приймати рішення на основі пропозицій машини, просто прийнявши рекомендацію, або вони можуть співпрацювати з машиною, щоб змінити рекомендацію.

Третій рівень – це автоматизація прийняття рішень, яка ще більше зменшує необхідну участь людини в процесі прийняття рішень. На цьому рівні машини виконують як етап прийняття рішення, так і етап виконання автономно. На першому етапі вони приймають

автономні рішення, використовуючи комбінацію таких інструментів, як правила, оптимізація та передбачення на основі ШІ. На другому етапі вони автоматично виконують ці рішення без участі людини. Натомість люди мають огляд на високому рівні, контролюють ризики та будь-які незвичайні дії та регулярно переглядають результати для вдосконалення системи.

Це здебільшого модель зрілості з можливістю просування по рівнях. Однак не всі рішення слід автоматизувати або доповнювати. Деякі настільки чутливі, складні або рідкісні, що краще тримати їх на рівні підтримки прийняття рішень і тримати людину в курсі. Це може включати рішення високого рівня, такі як розробка нової стратегії.

Ефективна система ДІ повинна пропонувати всі три режими роботи (тобто підтримку, доповнення та автоматизацію). Це дає змогу користувачам просуватися на рівні автоматизації, коли вони розвивають довіру до технології та її можливостей.

### **3. КОМПОНЕНТИ DECISION INTELLIGENCE**

Існує цілий набір технологій і алгоритмів, які забезпечують роботу машини прийняття рішень:

Машинне навчання. Алгоритми ML працюють з певною кількістю структурованих даних і виробляють пропозиції або рішення відповідно до заданих параметрів. Найпростішим прикладом є системи боротьби з шахрайством, які використовуються банками. Наприклад, коли користувач отримує доступ до свого банківського додатку з підозрілої IP-адреси, система приймає рішення про необхідність додаткової автентифікації користувача.

Глибоке навчання. Глибоке навчання є наступним етапом еволюції машинного навчання. У цьому випадку машина прийняття рішень враховує раніше прийняті рішення та їх результати при внесенні кожної нової пропозиції.

Візуальне моделювання рішень. Прийняття рішень штучним інтелектом є надійною відправною точкою, але рішення все одно приймаються власниками бізнесу та/або їхніми працівниками. Візуальне моделювання рішень є однією з особливостей програмного забезпечення для аналізу рішень, щоб показати людям, які приймають рішення, доступні варіанти та їхні результати.

Моделювання складних систем. Однією з переваг інтелектуального прийняття рішень є швидке створення складної бізнес-логіки, керуючись наявними даними та кінцевою метою.

Прогностична аналітика. Рішення, які приймають системи ШІ, базуються на досить точних прогнозах. Найпростіший приклад – прогнозування цін і автоматизована оптимізація в роздрібній торгівлі. У цьому випадку пропозиції, зроблені системою аналізу рішень, базуються на поточних і історичних коливаннях цін, прогнозованому попиту, майбутніх тенденціях і безлічі даних про поведінку клієнтів.

### **4. СФЕРИ ЗАСТОСУВАННЯ DECISION INTELLIGENCE**

Інтелектуальні рішення трансформують організації, забезпечуючи оптимальну продуктивність у таких критичних сферах, як продажі, маркетинг, управління магазинами, управління талантами та багато іншого.

Маркетинг. Сильний бренд є результатом ретельного планування та аналізу різних точок взаємодії. Маркетологи можуть швидко виявити високоефективні ініціативи в кількох каналах від соціальних мереж до електронної пошти та оптимізувати рентабельність інвестицій для кожної кампанії, використовуючи інструменти аналізу рішень.

Продажі. Було б найкраще в режимі реального часу мати інформацію про вподобання потенційних клієнтів і те, як вони хочуть отримати інформацію про ваші продукти чи

послуги, щоб створити потенційних клієнтів для вашої організації. Ви можете отримати цінну інформацію з даних споживачів за допомогою технологій аналізу рішень і використовувати їх для адаптації повідомлень, підвищення коефіцієнтів конверсії та збільшення доходу.

Управління магазином. Вирішальне значення для забезпечення прибутковості роздрібною торгівлі має визначення точного рівня запасів і ціноутворення. Роздрібні торговці можуть використовувати інструменти аналізу рішень, щоб оптимізувати стратегії ціноутворення та рівні запасів на основі моделей попиту, гарантуючи, що вони задовольняють запити споживачів, оптимізуючи прибуток.

Управління талантами – сьогодні організації отримують пул резюме під час публікації вакансії. Залучення талановитих людей може бути одним із найскладніших завдань для вашої організації, якщо у вас обмежені ресурси. Однак ви можете використовувати програмне забезпечення для аналізу рішень, щоб перевіряти дані кандидатів і надавати пропозиції на основі їх профілю та попередніх досягнень. ДІ дозволяє рекрутерам зосередити свої зусилля на тих, хто з більшою ймовірністю досягне успіху в їхніх фірмах.

Ланцюжок поставок – Ланцюг поставок є однією з важливих сфер організації, оскільки він підтримує безперервність процесів. ДІ може творити такі чудеса, як

- Автоматизований аналіз першопричини дефектів або затримок
- Оптимізація запасів
- Прогнозування попиту
- Ефективність постачальника
- Моніторинг
- Аналітика якості

HR – ДІ може допомогти з спеціальними показниками аналізу, такими як чисельність персоналу, вивільнення, винагорода, заохочення тощо, що полегшить відділу кадрів виконання завдань.

Випадки використання Decision Intelligence в галузях

- Сектор охорони здоров'я – можна покращити медичні результати за допомогою ДІ. Ви можете отримати детальне уявлення про наступний крок лікування, краще передбачити прогноз і досягти кращих результатів, даючи лікарям суперздібність.
- Екологічний сектор – Інтелектуальні рішення можуть прогнозувати та виявляти вразливі місця на основі історичних і поточних даних. Ми можемо прогнозувати кліматичні умови з кращою точністю за допомогою ДІ. Це допомагає уникнути будь-якого лиха і озброїтися.
- Банківський і фінансовий сектор – аналіз рішень може вивчати поведінку споживачів, вимоги та больові точки та допомагає організаціям персоналізувати рішення для клієнтів. Клієнти можуть скористатися видатними методами інвестування ДІ. Потім фахівці перевіряють ці стратегії, перш ніж рекомендувати їх клієнту.
- Енергетичний сектор – ДІ допомагає користувачам краще керувати своїми енергетичними ресурсами та забезпечує автоматичне прийняття рішень щодо енергії та витрат. ДІ може передбачати сонячну енергію та відповідно регулювати ємність акумулятора.
- Страхування – ДІ може проводити андеррайтинг полісів, моделювати ризики та виявляти шахрайство, допомагаючи організаціям обробляти справжні запити та підтримувати максимальну безпеку.

## 5. ГЛИБОКЕ НАВЧАННЯ З ПІДКРІПЛЕННЯМ

Decision intelligence базується на технології Глибокого навчання з підкріпленням.

Глибоке навчання з підкріпленням — це розділ машинного навчання, який поєднує в собі навчання з підкріпленням (RL) і глибоке навчання. RL розглядає проблему обчислювального агента, який навчається приймати рішення методом проб і помилок. Deep RL включає глибоке навчання в рішення, дозволяючи агентам приймати рішення на основі неструктурованих вхідних даних без ручного проектування простору станів. Алгоритми Deep RL здатні приймати дуже великі вхідні дані (наприклад, кожен піксель, що відображається на екрані у відеогрі) і вирішувати, які дії потрібно виконати для оптимізації цілі (наприклад, максимізація результату гри). Глибоке навчання з підкріпленням використовувалося для різноманітних програм, включаючи, але не обмежуючись, робототехніку, відеоігри, обробку природної мови, комп'ютерний зір, освіту, транспорт, фінанси та охорону здоров'я.

Походження глибокого навчання з підкріпленням — це чисте навчання з підкріпленням, де проблеми зазвичай формулюються як процеси прийняття рішень за Марковим (MDP). MDP складається з набору станів  $S$  і дій  $A$ . Переходи між станами виконуються з ймовірністю переходу  $P$ , винагородою  $R$  і коефіцієнтом дисконтування  $\gamma$ . Ймовірність переходу  $P$  відображає кількість різних переходів і винагород з одного стану в інший, де послідовний стан і винагорода залежать лише від стану та дії, виконаної на попередньому кроці.

Навчання визначає середовище для виконання Агентом певних дій (відповідно до політики), щоб максимізувати винагороду. Основою оптимальної поведінки Агента є рівняння Беллмана, яке є широко використовуваним методом вирішення практичних задач оптимізації.

Коли агент існує в середовищі і переходить в інший стан, нам потрібно оцінити значення стану  $V(s)$  (позиція) — функція значення стану. Коли ми знаємо значення кожного стану, ми можемо з'ясувати, який найкращий спосіб діяти  $Q(S, A)$  — функція значення дії (просто слідкуючи за станом із найвищим значенням).

Оптимальна функція значення стану — це та, яка має вище значення порівняно з усіма іншими функціями значення (максимальна віддача), тому функцію оптимального значення також можна оцінити, взявши максимум  $Q$ :

$$v_*(s) = \max_{a \in \mathcal{A}(s)} q_*(s, a)$$

Як правило, проблеми навчання з підкріпленням вирішуються за допомогою  $Q$  — алгоритмів навчання. Тут, як було сказано вище, Агент взаємодіє з Середовищем і отримує винагороду. Мета полягає в тому, щоб розробити оптимальну політику (стратегію вибору дій), щоб максимізувати винагороду. У процесі навчання Агент оновлює таблицю  $Q(S,A)$  (припинення відбувається, коли епізод закінчився — досягнуто мети).

$Q$  — алгоритм навчання виконується за наступними кроками:

1. Ініціалізуйте таблицю  $Q(S,A)$  випадковими значеннями.
2. Виконайте дію ( $A$ ) за допомогою політики  $\epsilon$ -greedy і перейдіть до наступного стану  $S'$
3. Оновіть значення  $Q$  попереднього стану, дотримуючись рівняння оновлення:

$$Q(s, a) = \overbrace{Q(s, a)}^{\text{Old value}} + \underbrace{\alpha}_{\text{learning rate}} \underbrace{(r + \gamma \max_{a'} Q(s', a'))}_{\text{optimal future value}} - \overbrace{Q(s, a)}^{\text{Old value}}$$

Навчання з підкріпленням може бути застосовним до середовища, де можна керувати (ітерувати) усіма досяжними станами та зберігати їх у стандартній оперативній пам'яті комп'ютера. Однак у середовищі, де кількість станів перевищує можливості сучасних комп'ютерів (для ігор Atari є 12833600 станів), стандартний підхід Reinforcement Learning не дуже застосовний. Крім того, в реальному середовищі Агент повинен стикатися з проблемами безперервних станів (не дискретних), безперервних змінних і безперервного контролю (дій).

Беручи до уваги складність середовища, в якому агент повинен працювати (кількість станів, безперервний контроль), стандартну добре визначену таблицю Reinforcement Learning, Q — таблицю, замінено глибокою нейронною мережею (Q — мережа), яка відображає середовище станів до дій Агента. Архітектура мережі, вибір гіперпараметрів мережі та навчання виконується на етапі навчання.

## 6. ВИСНОВКИ

Decision intelligence — це поєднання існуючих технологій, зокрема штучного інтелекту та автоматизації процесів, з можливістю робити більше, ніж будь-яка з них окремо. ШІ та машинне навчання зосереджені на даних; вони можуть генерувати ідеї, але часто не пов'язані з виконанням та результатами рішень. А додатки для бізнес-процесів (зокрема роботизована автоматизація процесів, інтелектуальний аналіз процесів і виявлення процесів) орієнтовані на завдання. Вони можуть автоматизувати завдання та зробити їх ефективнішими, але можуть виконувати лише те, на що їх запрограмували, і мають обмежений вплив на ефективність рішень.

Нижче наведено основні переваги інтелектуальних рішень, на які можуть розраховувати компанії.

Рішення на основі даних. У той час як 91% компаній вважають, що прийняття рішень на основі даних може прискорити розвиток їхнього бізнесу, лише 57% з них покладаються на свої дані. Щоб отримати конкурентну перевагу, необхідно правильно проаналізувати наявні дані, зробити деякі прогнози та вибрати найкращий варіант. ШІ може краще розглянути масив даних і знайти невидимі закономірності та можливі аномалії, які можуть суттєво вплинути на результат. ДІ надає доступ до всієї необхідної інформації та інструментів, необхідних для прийняття більш обґрунтованих рішень на основі складних даних, а не на передчуттях чи інтуїційних реакціях.

Швидше прийняття рішень. Згідно з опитуванням McKinsey, лише 20% організацій задоволені своєю швидкістю прийняття рішень. Інші визнають, що витрачають надто багато часу на правильний вибір, який насправді не завжди правильний. Системи прийняття рішень ШІ максимально прискорюють процес, оскільки вони здатні майже миттєво обробляти величезні обсяги даних.

Підвищена точність – коли люди залучені, під час прийняття рішень можуть виникнути особисті упередження та неточності. ДІ пом'якшує вплив цих помилок і упереджень. Про все це піклується запрограмований алгоритм, підвищуючи точність прийняття рішень. ДІ може допомогти прийняти кращі, більш обґрунтовані рішення та уникнути конфлікту цінностей та

інтересів. Ці рішення не залежать від когнітивних упереджень і тому можуть допомогти звузати найкращі результати.

Кілька варіантів вирішення проблеми. Алгоритми прийняття рішень на основі штучного інтелекту також можуть бути досить гнучкими та виділяти кілька результатів певного рішення, коли змінюється один із параметрів. Ця функція може допомогти бізнесу зробити найкращий вибір із безлічі варіантів, враховуючи поточні цілі та стратегії розвитку.

Зменшує залежність – це позбавляє потреби залежати від аналітиків для створення звітів і інформаційних панелей і надання інформації зацікавленим сторонам. Цей метод обмежує продуктивність даних, відповідаючи лише на кілька бізнес-питань. За допомогою ДІ зацікавлена сторона сама може отримати інформацію та рекомендації, які надають безмежні можливості для прийняття високоякісних рішень.

Моніторинг і масштабованість – можливість безперервного моніторингу ДІ робить його проактивним, виявляє викиди та забезпечує персоналізований досвід для користувача. ДІ розглядає мільярди точок даних, щоб прийняти рішення. Організаціям потрібна масштабована система, оскільки дані зростають експоненціально. З цієї причини ДІ має масштабований, надійний, високошвидкісний механізм аналітики, обчислювальну структуру та засоби безпеки/контролю доступу.

## **СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Is Decision Intelligence The New AI? [Електронний ресурс] - Режим доступу до ресурсу: <https://www.forbes.com/sites/forbestechcouncil/2022/05/25/is-decision-intelligence-the-new-ai> Дата доступу - 16.09.2022.

2. Why Decision Intelligence Is The Most Important Data Analytics Trend Of This Decade [Електронний ресурс] - Режим доступу до ресурсу: <https://www.atscale.com/blog/decision-intelligence-most-important-data-analytics-trend-this-decade/> Дата доступу - 16.09.2022.

3. Decision Intelligence: Diving Into One of The Top Trends of 2022 [Електронний ресурс] - Режим доступу до ресурсу: <https://statisticallyrelevant.com/decision-intelligence-diving-into-one-of-the-top-trends-of-2022/> Дата доступу - 16.09.2022.

4. Introduction to Decision Intelligence [Електронний ресурс] - Режим доступу до ресурсу: <https://www.infopulse.com/blog/introduction-decision-intelligence> Дата доступу - 16.09.2021.

5. Decision Intelligence Use Cases to Prioritize in 2022 [Електронний ресурс] - Режим доступу до ресурсу: <https://diwo.ai/blog/4-decision-intelligence-use-cases-to-prioritize-in-2022/> Дата доступу - 24.09.2022.

# **ОЦІНЮВАННЯ І ПРОГНОЗУВАННЯ СТАНУ ПАЦІЄНТІВ ТА ПОЛІПШЕННЯ ЕФЕКТИВНОСТІ ВИМІРНИХ ДАНИХ ТА ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ ПРИ АНАЛІЗІ НЕТОЧНИХ ТА НЕПОВНИХ ВИМІРЯНИХ ДАНИХ**

Медвідь В.А.<sup>1</sup>, Харченко К.В.<sup>2</sup>

Національний технічний університет України  
«Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна

<sup>1</sup> medved.vlad01@gmail.com, <sup>2</sup> konst1970@gmail.com

**Метою дослідження є розгляд різноманітних методів, які можна використовувати для оцінювання і прогнозування стану пацієнтів, а також методів та алгоритмів прийняття рішень при неповних чи неточних даних. Буде розглянуто особливості даних методів, а також засоби мови програмування Python, котрі підходять для їх програмної реалізації. На думку авторів, розглянуті методи та програмні засоби здатні покращити та пришвидшити процес постановки діагнозу та оцінювання стану пацієнтів**

**Ключові слова: методи прогнозування, методи прийняття рішень, мова програмування Python.**

## **1. ВСТУП**

Медичні дані є одними з найцінніших і водночас найскладніших даних для аналізу. Як постачальники медичних послуг можуть використовувати сучасні інструменти та технології аналізу даних для аналізу та створення цінності складних даних? Аналітика даних, яка обіцяє ефективно виявляти цінні закономірності шляхом аналізу великої кількості неструктурованих, неоднорідних, нестандартних і неповних даних охорони здоров'я. Він не лише прогнозує, але й допомагає у прийнятті рішень і все частіше помічається як прорив у постійному прогресі з метою покращення якості обслуговування пацієнтів і зниження вартості медичної допомоги.

Через зростання витрат на охорону здоров'я рання профілактика захворювань ніколи не була такою важливою, як сьогодні. Це, зокрема, пов'язано зі збільшенням загроз нових варіантів захворювань, біотероризму, а також нещодавніх удосконалень збору даних і обчислювальних технологій. Збільшення кількості даних про охорону здоров'я збільшує попит на розробку ефективного, чутливого та економічно ефективного рішення для профілактики захворювань. Традиційні профілактичні заходи в основному зосереджені на просуванні переваг охорони здоров'я та не мають методів обробки величезної кількості даних. Використання ІТ для підвищення якості охорони здоров'я може сприяти зміцненню здоров'я та профілактиці захворювань. Це справжній міждисциплінарний виклик, який вимагає низки типів досвіду в різних областях досліджень і справді великих даних.

## **2. МЕТОДИ ПРОГНОЗУВАННЯ СТАНУ ПАЦІЄНТІВ**

Існує три основні типи методів прогнозування: якісні методи, аналіз часових рядів і прогнозування, а також моделі причинно-наслідкових зв'язків.

Перший використовує якісні дані (наприклад, експертну думку) та інформацію про особливі події, згадані вище, і може брати або не брати до уваги минуле.

Другий, з іншого боку, повністю зосереджується на шаблонах і змінах шаблонів і, таким чином, повністю покладається на історичні дані.

Третій використовує високоточну та специфічну інформацію про взаємозв'язки між елементами системи та є достатньо потужним, щоб формально враховувати спеціальні події. Як і в аналізі часових рядів і методах прогнозування, минуле є важливим для причинно-наслідкових моделей.

### **2.1. Якісні методи**

В основному вони використовуються, коли даних мало, наприклад, коли продукт вперше виходить на ринок. Вони використовують людське судження та рейтингові схеми, щоб перетворити якісну інформацію на кількісні оцінки.

Мета тут полягає в тому, щоб логічним, неупередженим і систематичним чином об'єднати всю інформацію та судження, які стосуються факторів, що оцінюються. Такі методи часто використовуються в сферах нових технологій, де розробка ідеї продукту може вимагати кількох «винаходів», так що потреби в дослідженнях і розробках важко оцінити, і де рівень прийняття на ринку та проникнення вкрай невизначений.

### **2.2. Аналіз часових рядів**

Це статистичні методи, які використовуються, коли доступні дані за кілька років для продукту чи лінійки продуктів і коли зв'язки та тенденції чіткі та відносно стабільні.

Одним із основних принципів статистичного прогнозування — фактично будь-якого прогнозування, коли доступні історичні дані — є те, що прогнозіст повинен використовувати дані про минулі показники, щоб отримати «показник спідометра» поточного темпу і як швидко ця швидкість зростає або зменшується. Основою прогнозу є поточний темп і його зміни — «прискорення» і «уповільнення». Коли вони відомі, різні математичні методи можуть розробити проєкції з них.

Зазвичай складно робити прогнози на основі необроблених даних, оскільки показники та тенденції не відразу очевидні; вони змішуються, наприклад, із сезонними коливаннями і, можливо, спотворюються різними факторами. Необроблені дані необхідно обробити, перш ніж їх можна буде використовувати, і це часто робиться за допомогою аналізу часових рядів.

Зараз часовий ряд — це набір упорядкованих у хронологічному порядку точок необроблених даних — наприклад, продажі певного продукту підрозділом за місяцями за кілька років. Аналіз часових рядів допомагає визначити та пояснити:

1) Будь-яка регулярність або систематична зміна в ряді даних, яка зумовлена сезонністю — «сезонність».

2) Циклічні моделі, які повторюються через будь-які два-три роки або більше.

3) Тенденції в даних.

4) Темпи зростання цих тенденцій.

Після завершення аналізу можна починати роботу з прогнозування. Слід зазначити, більшість методів статистичного прогнозування фактично поєднують обидві функції в одній операції.

### **2.3. Моделі причинно-наслідкових зв'язків**

Коли доступні історичні дані та проведено достатній аналіз, щоб чітко визначити зв'язки між фактором, який потрібно прогнозувати, та іншими факторами, прогнозіст часто будує причинно-наслідкову модель.

Причинно-наслідкова модель є найдосконалішим інструментом прогнозування. Вона математично виражає відповідні причинно-наслідкові зв'язки та може включати міркування щодо конвеєра (тобто запаси) та інформацію з огляду ринку. Вона також може безпосередньо включати результати аналізу часових рядів.

Причинно-наслідкова модель бере до уваги все, що відомо про динаміку потокової системи, і використовує передбачення пов'язаних подій. Якщо дані доступні, модель зазвичай включає фактори для і з'єднує їх рівняннями для опису загального потоку продукту.

Якщо певних типів даних бракує, спочатку може знадобитися зробити припущення щодо деяких зв'язків, а потім відстежити, що відбувається, щоб визначити, чи припущення відповідають дійсності. Як правило, причинно-наслідкова модель постійно переглядається, оскільки стає доступним більше знань про систему.

### 3. МЕТОДИ ПРИЙНЯТТЯ РІШЕНЬ ПРИ НЕПОВНИХ ТА НЕТОЧНИХ ДАНИХ

Коли ми маємо неповні чи неточні дані, тоді використовуються наступні метод та алгоритми прийняття рішень.

#### 3.1. Критерій Лапласа

Критерій Байеса-Лапласа — один з критеріїв прийняття рішень в умовах невизначеності. Умовами невизначеності вважається ситуація, коли наслідки прийнятих рішень невідомі, і можна лише приблизно їх оцінити. За цим критерієм множина оптимальних альтернатив знаходиться так: критерій передбачає існування імовірнісних мір  $p(x,s)$  на  $X \times S$ ,  $u(x,s)$ , де — імовірнісна міра на декартовому добутку  $X \times S$ , де  $X$  — множина альтернатив,  $S$  — множина станів, які до того ж є стабільними протягом тривалого періоду часу.

Для того, щоб це було це було так ЗПП повинна бути добре досліджена статистично(на основі тривалих або частих спостережень).

Отже спочатку обчислюється:

$$E(X_k) = \sum_{j=1}^N u_{kj} p_{kj}, k = \overline{1, N}$$

де  $u(x,s)$  — функція рішень, визначена на  $X \times S$ , де  $X$  — множина альтернатив,  $S$  — множина станів, а  $p(x, s)$  — ймовірнісна міра ситуації  $fx, sg$

Для скінченновимірною випадку набуває:

$$E(X_k) = \sum_{j=1}^N u_{kj} p_{kj}, k = \overline{1, N}$$

де  $p_{kj}$  імовірність ситуації  $\{x_k, s_j\}$ , де  $u_{kj}$  матриця рішень.

Далі вже множина оптимальних альтернатив визначається так:

$$X_{BL} = \arg \max_{x \in X} E(x)$$

$$X_{BL} = \arg \max_{k=1, N} E(x_k)$$

### 3.2 Критерій Вальда

Критерій Вальда є критерієм крайнього песимізму, оскільки статистик вважає, що "природа" діє проти нього найгіршим чином. Це критерій гарантованого результату. Нехай гру задано матрицею виграшів гравця А. Тоді на думку статистика - гравця А, дії гравця "природа", якій діє проти нього найгіршим чином, відображуються в реалізації гравцем "природа" таких своїх стані П<sub>і</sub>, при яких величина виграшу гравця А (статистика) приймає найменше значення min a<sub>ij</sub>. Виходячи з цього статистик обирає таку чисту стратегію А<sub>і</sub>, при якій найменший виграш min a<sub>ij</sub> буде максимальним.

Величина α<sub>В</sub>, що розраховується за вищенаведеною формулою, називається нижньою ціною гри - це максимальний виграш, що є гарантованим в грі з певним противником шляхом вибору однієї зі своїх стратегій при мінімальних результатах. Нехай гру задано матрицею програшів гравця А, тоді найгірші дії гравця "природа", будуть реалізовуватися в таких станах П<sub>і</sub>, при яких величина програшу гравця А (статистика) приймає найбільше значення max a<sub>ij</sub>. Виходячи з цього статистику необхідно обрати таку чисту стратегію А<sub>і</sub>, при якій найбільший програш max a<sub>ij</sub> буде мінімальним.

### 3.3 Критерій песимізму

Якщо суб'єкт управління орієнтується на вкрай несприятливий розвиток подій, для обґрунтування рішення застосовується критерій песимізму. Цей критерій передбачає визначення варіанту рішення, що дозволяє мінімізувати мінімальні виграші для кожного варіанту економічного середовища, й за вихідними даними матриці виграшів у формалізованому виразі має вигляд:

$$\alpha_s = \min_i \min_j a_{ij}$$

За умови застосування даних матриці ризиків визначення критерію песимізму передбачає знаходження варіанту рішення, який максимізує максимальні програші для кожного варіанту економічного середовища:

$$\beta_s = \max_i \max_j H_{ij}$$

### 3.4 Критерій Севіджа

*Мінімакський критерій Севіджа* використовується у випадках, коли потрібно за будь-яких умов уникнути великого ризику. Відповідно до цього критерію перевагу слід надати варіанту рішення, для якого максимальні втрати за різних варіантів стану економічного середовища виявляться мінімальними. Його формалізований вираз має вигляд:

$$\alpha_c = \min_i \max_j H_{ij}$$

### 3.5 Критерій Гурвіца

*Критерій узагальненого максиміна (песимізму-оптимізму) Гурвіца* використовується, якщо потрібно знайти певну комбінацію оптимістичної та песимістичної позицій щодо прийняття рішення. Відповідно до цього критерію перевагу надають варіанту рішення, для якого виконується умова:

$$H = \max_i \left( k \min_j X_{ij} + (1-k) \max_j X_{ij} \right).$$

де  $k$  – коефіцієнт, що розглядається як показник песимізму  $0 \leq k \leq 1$ ,  $a_{ij}$  – виграш, що відповідає  $i$ -му рішенню при  $j$ -му варіанті стану економічного середовища.

Значення коефіцієнту песимізму встановлюються суб'єктивно, залежно від конкретних обставин та схильності до ризику особи, що приймає рішення. При  $k = 0$  має місце оптимістична позиція та орієнтація на граничний ризик, а величина критерію Гурвіца співпадає з величиною критерію максимакса. При  $k = 1$  суб'єкт управління налаштований песимістично та прагне уникати ризику, а величина критерію Гурвіца співпадає з величиною максимінного критерію Вальда. Значення  $k$  між 0 і 1 є проміжними між ризиком і обережністю. Чим більша можлива небезпека, тим більш наближуватиметься до одиниці значення коефіцієнта песимізму. При зміні значення коефіцієнта песимізму змінюватиметься й варіант рішення, якому надається перевага. Отже, схильність до ризику суб'єкта управління значною мірою впливає на вибір рішення.

## 4. ПРОГРАМНІ ЗАСОБИ ДЛЯ РЕАЛІЗАЦІЇ МЕТОДІВ ТА АЛГОРИТМІВ

Для програмної реалізації вищерозглянутих методів якнайкраще підходить мова програмування Python, з її величезним різноманіттям модулів. Саме велика кількість різних модулів та бібліотек дозволяє цій мові бути гнучкою для вирішення різноманітних завдань, наприклад обробка великих масивів даних, машинне навчання і так далі. З огляду на специфіку розглянутої теми та методів, можна зробити висновок, що мова програмування Python ідеально підходить, так як має засоби для обробки та аналізу чималих об'ємів даних з використанням розглянутих методів. Розглянемо далі які модулі якнайкраще підходять для програмної реалізації вище зазначених методів та алгоритмів.

### 4.1 NumPy

NumPy це open-source модуль для python, який надає загальні математичні та числові операції у вигляді пре-компільованих, швидких функцій. Вони поєднуються у високорівневі пакети. Вони забезпечують функціонал, який можна порівняти із функціоналом MatLab. NumPy (Numeric Python) надає базові методи для маніпуляції з великими масивами та матрицями. SciPy (Scientific Python) розширює функціонал numpy величезною колекцією корисних алгоритмів, таких як мінімізація, перетворення Фур'є, регресія та інші прикладні математичні техніки.

Навіщо використовувати NumPy?

У Python у нас є списки, які служать для цілей масивів, але вони повільно обробляються. NumPy має на меті надати об'єкт масиву, який у 50 разів швидший за традиційні списки Python. Об'єкт масиву в NumPy називається ndarray, він надає багато допоміжних функцій, які роблять роботу з ndarray дуже легкою. Масиви дуже часто використовуються в науці про дані, де швидкість і ресурси дуже важливі.

Чому NumPy швидший за списки?

Масиви NumPy зберігаються в одному безперервному місці в пам'яті, на відміну від списків, тому процеси можуть отримувати до них доступ і маніпулювати ними дуже ефективно.

Така поведінка в інформатиці називається локальністю посилання.

Це головна причина, чому NumPy швидше, ніж списки. Крім того, він оптимізований для роботи з останніми архітектурами ЦП.

## 4.2 SciPy

SciPy в Python — це бібліотека з відкритим кодом, яка використовується для вирішення математичних, наукових, інженерних і технічних задач. Це дозволяє користувачам маніпулювати даними та візуалізувати дані за допомогою широкого спектру високорівневих команд Python. SciPy побудовано на розширенні Python NumPy.

Навіщо використовувати SciPy?

Якщо SciPy використовує NumPy нижче, чому ми не можемо просто використовувати NumPy? SciPy оптимізував і додав функції, які часто використовуються в NumPy і Data Science.

## 4.3 Matplotlib

Що означає Matplotlib?

Matplotlib — це бібліотека для побудови графіків, доступна для мови програмування Python як компонент NumPy, ресурсу чисельної обробки великих даних. Matplotlib використовує об'єктно-орієнтований API для вбудовування графіків у програми Python.

Оскільки Python широко використовується в машинному навчанні, такі ресурси, як NumPy і matplotlib, часто корисні для моделювання технологій машинного навчання. Ідея полягає в тому, щоб програмісти отримували доступ до цих бібліотек для виконання ключових завдань у ширшому середовищі Python та інтегрували результати з усіма іншими елементами та функціями програми машинного навчання, нейронної мережі чи іншої вдосконаленої машини. Утиліта NumPy і matplotlib пов'язана з числами — утиліта matplotlib конкретно пов'язана з інструментами візуального малювання. Тож у певному сенсі ці ресурси більше аналітичні, ніж генеративні. Однак уся ця інфраструктура працює разом, щоб дозволити програмам машинного навчання давати результати, корисні для людей, які працюють із ними.

## 5. ВИСНОВКИ

Постійний розвиток технологій дозволяє покращувати різноманітні сфери людського існування, якою і являється медицина. На основі розглянутих методів прогнозування та прийняття рішень, а також з використанням можливостей мови програмування Python, створюється можливість створювати програми та системи які можуть ефективно оцінювати та прогнозувати стан різних пацієнтів, приймати рішення щодо подальшого лікування чи стану пацієнту, а також покращувати дані, навіть за умови, коли вони неточні та неповні, так як не завжди є можливість отримати повний набір даних, який би ідеально підходив, як вхідні дані.

У даній роботі розглянуто як методи так і програмні засоби, які здатні покращити якість обслуговування у сфері медицині, зекономити час та кошти для запобігання, лікування а також профілактики різноманітних захворювань.

## ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. «How to Choose the Right Forecasting Technique» by John C. Chambers, Satinder K. Mullick, and Donald D. Smith, URL: <https://hbr.org/1971/07/how-to-choose-the-right-forecasting-technique>.
2. Принятие решений при неполной информации, URL: [https://studwood.net/1682726/matematika\\_himiya\\_fizika/prinyatie\\_resheniy\\_nepolnoy\\_informatsii](https://studwood.net/1682726/matematika_himiya_fizika/prinyatie_resheniy_nepolnoy_informatsii)
3. Критерії прийняття рішень в умовах невизначеності, URL: [https://stud.com.ua/34760/finansi/kriteriy\\_laplasa](https://stud.com.ua/34760/finansi/kriteriy_laplasa)
4. NumPy Introduction, URL: [https://www.w3schools.com/python/numpy/numpy\\_intro.asp](https://www.w3schools.com/python/numpy/numpy_intro.asp)
5. SciPy in Python, URL: <https://www.guru99.com/scipy-tutorial.html>
6. What is Matplotlib in Python?, URL: <https://www.activestate.com/resources/quick-reads/what-is-matplotlib-in-python-how-to-use-it-for-plotting/>

# ПЕРЕВІРКА ЕФЕКТИВНОСТІ ВИКОРИСТАННЯ МАШИННОГО НАВЧАННЯ У АВТОМАТИЗАЦІЇ РЕГРЕСИВНОГО ТЕСТУВАННЯ

Михайловин Р. Г.<sup>1</sup> Булах Б. В.<sup>2</sup>

КПІ ім. Ігоря Сікорського, Київ, Україна

<sup>1</sup>mihailovinroman@gmail.com , <sup>2</sup>bogdan.bulakh@gmail.com

**Метою дослідження є перевірка ефективності використання машинного навчання на прикладі нейронних мереж у автоматизації регресійного тестування. У доповіді представлено опис причин та аргументи на користь даного підходу, та результати тестової реалізації. На думку автора, залучення подібних методів до автоматизації тестування здатне прискорити та спростити ітерації регресивного тестування.**

**Ключові слова: нейронні мережі, машинне навчання, регресійне тестування, автоматизація.**

## 1. ВСТУП

Розробка програмного забезпечення - складний і трудомісткий процес, в ході якого неодмінно виникають помилки. Частина з них так і не будуть виявлені, поки програмний продукт не потрапить до кінцевого користувача. Для того щоб скоротити число прихованих проблем, застосовуються різні методи тестування ПЗ.

Важливою частиною тестування програмного продукту є - регресивне тестування.

Регресивне тестування— збірна назва для різних способів тестування ПЗ, що мають на меті виявлення помилок у вже протестованих ділянках коду. Помилки, що виникають при внесенні змін до програми називають регресивними.

В умовах, коли нові версії програмного забезпечення з'являються одна швидше іншої, вимагаючи скорочення часу виходу на ринок і підвищення якості, автоматизація регресійного тестування є логічним і ефективним кроком.

Одним з варіантів автоматизації є використання нейронних мереж.

Добре навчена нейронна мережа здатна передбачати результат по набору вхідних значень. Разом з цим додатково вона вирішує завдання класифікації, якщо заздалегідь задані класи у вхідній множині, або завдання кластеризації, тобто виявлення класів в множині.

## 2. РЕГРЕСИВНЕ ТЕСТУВАННЯ

### 2.1. Завдання регресивного тестування

Основне завдання регресивного тестування - перевірка того, що виправлення помилки не торкнулося існуючої функціональності. Через часте виконання одних і тих же наборів сценаріїв, рекомендується використовувати автоматизовані регресивні тести, що дозволить скоротити терміни тестування.

- перевірка та затвердження виправлення помилки;
- тестування наслідків виправлень, так як внесення виправлення можуть принести помилку в код який виконувався правильно;
- гарантувати функціональну спадкоємність і сумісність нової версії (релізу, патча) з попередніми;

- зменшення вартості і скорочення часу виконання тестів.

## **2.2. Використання регресивного тестування**

Для регресивного тестування використовуються тест-кейси, написані на попередніх стадіях розробки і тестування. Це дає гарантію того, що зміни в новій версії програми не пошкодили вже існуючу функціональність. Рекомендується проводити автоматизацію регресійних тестів, для прискорення подальшого процесу тестування і виявлення дефектів на ранніх стадіях розробки програмного забезпечення.

Проводити регресивне тестування, слід після будь-якої зміни функціоналу, для того, щоб переконатися у відсутності нових і / або усунення попередніх помилок. Включення блочного регресивного тестування в процес розробки дозволяє захиститися від помилок. Баги будуть виявлені відразу після виникнення і не зможуть стати причинами поширення помилок в додатку. Перевірка цілісності проекту після внесення змін призначена для того, щоб протестувати загальний функціонал оточення, в якому були зроблені зміни.

Повторна поява одних і тих же помилок при розробці програмного забезпечення досить часто явище. Це відбувається через помилки при роботі з системою контролю версій або через людські помилки. Але настільки ж часто вирішення проблеми буває «недовготривалим»: рішення може працювати до наступної зміни в програмі. І нарешті, при зміні частини коду часто можуть з'являтися ті ж помилки, що були в попередній реалізації [1].

Спочатку потрібно обрати спосіб тестування програми. Після того як обраний спосіб тестування програми, його потрібно оптимізувати, інакше ефективність дій буде мінімальною.

Для цього існують такі поширені і дієві методи:

1. Із застосуванням дворівневого підходу.
2. Із застосуванням сортування тест-кейсів за пріоритетом.
3. За допомогою автоматизованого підходу.

## **2.3. Методи оптимізації**

Метод оптимізації із застосуванням дворівневого підходу полягає в тому, що регресійне тестування розбивається на два етапи. На першому етапі кожен тестувальник зосереджується на тих ділянках коду, які були недавно змінені або створені. В цей час розглядаються всі внесені зміни і їх вплив на загальну функціональність проекту (іmpact analysis). На наступному етапі проводиться повне тестування ПЗ на всіх ділянках коду. Зазвичай цю процедуру виконують перед випуском нової версії, щоб бути повністю впевненими в працездатності програми. На перший погляд може здатися, що цей етап досить простий. Однак тут є свої нюанси, які обов'язково необхідно враховувати. Безперервна комунікація з розробниками програмного продукту дозволить своєчасно виявити тонкі місця в системі. Завдяки цьому забезпечується можливість проводити ефективні тести і економити час на необов'язкових перевірках.

Другий метод оптимізації із застосуванням сортування тест-кейсів за пріоритетом базується на виборі ділянок коду для тестування, в яких були зроблені останні зміни. При цьому кожен тест-кейс сортується за пріоритетністю. Вибрані тест-кейси можуть мати такі пріоритети:

- високий;
- середній;
- низький.

Тестувальники включають у високий пріоритет тести, які перевіряють критичні ділянки програми, що відповідають за її головні функції. Тестами із середнім пріоритетом перевіряються області коду, в яких раніше були виявлені недоліки. Тести з низьким пріоритетом в основному використовують перед великим випуском нової версії програми.

Метод оптимізації за допомогою автоматизованого підходу полягає в правильній розстановці етапів тестування в процесі розробки проекту. На ранній стадії розробки важливо проводити функціональне тестування тільки після того, як програмний продукт досяг певної зрілості і може виконувати покладені на нього функції. Також варто враховувати, що, незважаючи на високий рівень продуктивності автоматизованих тестів і економію часу, вони не завжди залишаються актуальними. Тому варто після випуску нових версій програмного продукту переглядати автоматизовані тести і в разі необхідності замінювати застарілі на нові пакети. Завдяки такому підходу можна забезпечити високу якість програмної продукції [2].

### **3. МАШИННЕ НАВЧАННЯ І НЕЙРОННІ МЕРЕЖІ**

#### **3.1 Машинне навчання**

У випадку автоматизації процедур тестування програмного забезпечення, стандартне машинне навчання і, більш конкретно, методи глибокого навчання, такі як когнітивні нейронні мережі, можуть бути навчені даними, що генеруються діями користувача, у поєднанні з відповідним результатом тестованої програми. Для цього дії тестера, який проводить регресійне тестування, є гілками в нейронній мережі, тоді як елементи сторінки - це вузли. Коли вимірний результат згодом кваліфікується в контрольованих навчальних моделях, він може бути застосований для прогнозування результатів для майбутніх циклів регресійного тестування, а також для автономного генерування нових тестових випадків. Значення підготовки моделей глибокого навчання для прогнозування введення користувачем і вихідних даних системи зростає і стає все більш точним, оскільки дані накопичуються в кожному тестовому циклі [3].

#### **3.2 Нейронні мережі**

Нейронна мережа являє собою систему  $N$  розподілених елементів, пов'язаних між собою змінними зв'язками. Ці елементи називаються формальними нейронами і характеризуються рівнем збудження. Нейрон - це атомарна одиниця нейронної мережі. Отримавши дані на вхід, він обробляє їх і передає вихід, який є входом для наступного рівня нейронів. Зв'язки між нейронами визначені коефіцієнтами передач.

Штучна нейронна мережа має три складові:

- Вхідний шар;
- Приховані (обчислювальні) шари;
- Вихідний шар.

Іншими словами, нейронна мережа - це математична модель, заснована на когнітивних процесах мозку. Так само як і мозок, нейронна мережа складається з вузлів (нейронів) і змінних зв'язків між ними (ваги нейронних зв'язків). Нейрони поділяються на 3 групи за своїми властивостями: вхідні нейрони, вихідні нейрони і нейрони прихованого шару (обчислювальні вузли). Фактично нейронна мережа - це особливий спосіб завдання функції[4].

Мережі бувають:

1. Одношарові. У даній структурі сигнали зі вхідного шару відразу спрямовуються на вихідний шар, який, перетворюючи сигнал, видає відповідь. Таким чином, 1-й вхідний шар тільки приймає і розподіляє сигнали, а другий шар проводить обчислення та

видає результат. Вхідні нейрони об'єднані з основним шаром за допомогою зв'язків, які називають синапсами з різними вагами, що забезпечує якість.

2. Багатошарова нейронна мережа. Має, між вхідним та вихідним ще декілька шарів. Такі шари називаються прихованими. Такі мережі мають більший функціонал ніж одношарові, але також навчаються значно повільніше [5].

Також нейронні мережі поділяють за такими критеріями:

За типом нейронів:

- Однорідні
- Гібридні

За методом навчання:

- Навчання з учителем
- Навчання без учителя
- Навчання з підкріпленням

За типом вхідних даних:

- Аналогові
- Двійкові
- Образні

За налаштуванням синапсів:

- З фіксованими зв'язками
- З динамічними зв'язками

Для навчання нейронній мережі дають приклади – навчальні вибірки. Представницькі дані підбираються користувачем, після чого запускається алгоритм навчання. Для навчання нейронної мережі потрібен набір евристичних знань про відбір і підготовку даних, вибір потрібної архітектури мережі та аналізу результатів, хоча, для успішного застосування нейронних мереж, потрібен менший об'єм знань, ніж, наприклад, при використанні традиційних методів статистики.

Привабливість нейронних мереж зумовлена їх інтуїтивністю, адже вони засновані на примітивній біологічній моделі нервових систем [6].

Процес навчання достатньо простий. Підготовується навчальна вибірка - множина пар вхідних та вихідних векторів, із завчасно відомими та правильними значеннями. Елементи цієї множини задовольняють наступним властностям - незалежність, послідовність, вибірку можна розширити, а її елементи можуть бути переставлені у будь-якому порядку. Навчальну вибірку пропускають через нейронну мережу і в результаті цього процесу ваги в нейронних зв'язках змінюються таким чином, щоб задати функцію, що задовольняє навчальну вибірку. Це призводить до того, що якість роботи нейромереж залежить від якості її навчання.

Далі готують тестову вибірку аналогічну навчальній, з іншими значеннями. Якщо результат тестування невідповідний, варто краще навчити мережу або змінити її структуру.

Нейромережеві технології спільно з широко використовуваними методами автоматичного тестування цілком здатні поліпшити результати автоматичного тестування в задачах тестування складних програмних систем [7].

## 4. ОПИС ПРАКТИЧНОЇ ЧАСТИНИ

Для дослідження було обрано багатошарову нейронну мережу навчену за методом – навчання з учителем.

Навчання з учителем (supervised learning) потребує наявності повного набору розмічених даних для тренування моделі на всіх етапах її навчання.

Це означає, що для кожного прикладу з навчального набору існує відповідь, що йому відповідає, її алгоритм і повинен отримати. Таким чином, мережу навчають. Потім, коли мережа отримує, наприклад, новий набір змін, вона порівнює його з прикладами з навчального датасета, щоб передбачити відповідь.

Зазвичай навчання з учителем використовується для задач регресії і класифікації.

Таким чином, навчання з учителем найбільше ефективно, коли є значний набір достовірних даних для навчання алгоритму. На жаль такі випадки досить рідкісні. Брак даних – одна з найбільших проблем у машинному навчанні [8].

Для дослідження було проведено експеримент у ході якого було розроблено програму з використанням нейронних мереж. Завдання програми було при отриманні набору внесених змін, обрати найбільш релевантні для проходження тест-кейси. Отже вхідними даними для програми є – тип внесених змін, а результатом її роботи – набір тест-кейсів обов'язкових до виконання. Схему роботи програми можна побачити на Рис. 1.

Програма представляє з себе набір з нейронних мереж типу багатошаровий перцептрон. Результати її тестування зображені на Рис. 2.

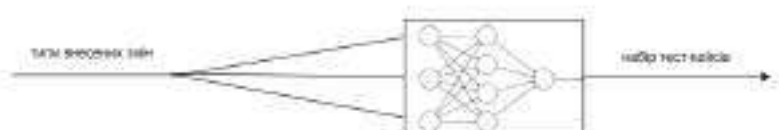


Рисунок 1. Схема роботи програми

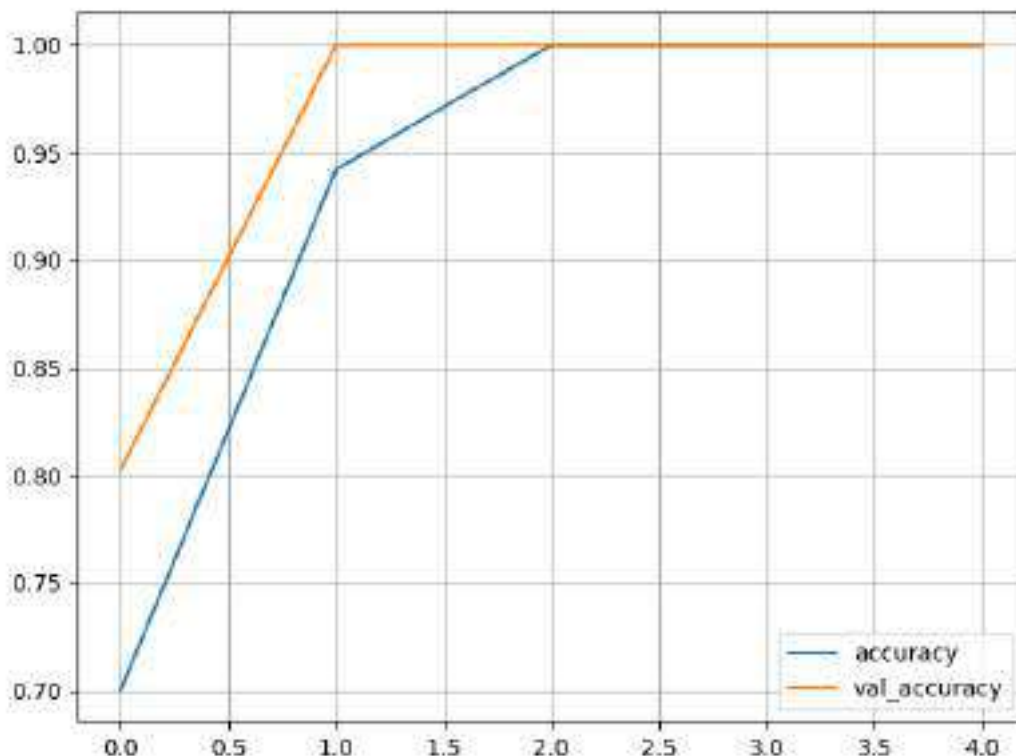


Рисунок 2. Точність програми на тестових даних.

Як бачимо, правильно навчені нейронні мережі показують високу точність. До переваг даної системи відносимо:

- Можливість паралельного навчання мереж
- Висока точність
- Можливість додати мережу без необхідності перенавчати інші

Недоліками подібної системи є:

- Необхідність у розмічених навчальних даних
- Необхідність перенавчання системи при зміні набору вхідних даних

## 5. ВИСНОВКИ

Як бачимо на прикладі даної програми, використання машинного навчання може бути ефективним у автоматизації регресивного тестування, доведення чого і було ціллю дослідження. Даний підхід може бути експериментально порівняний з іншими методами автоматизації для обрання найбільш ефективного підходу.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Лучшие практики автоматизации тестирования URL: <http://getbug.ru/luchshie-praktiki-avtomatizatsii-testirovaniya/>
2. Особенности регрессионного тестирования программ А. Коттов URL: <https://dnipro.deveducation.com/blog/osobennosti-regressionnogo-testirovaniya-programm/>
3. Automated GUI Regression Testing Using AI Planning URL: <https://www.functionize.com/blog/how-ai-impacts-regression-testing/>
4. Применение нейронных сетей для построения адаптивных систем управления технологическими процессами А. Юдашкин URL: <http://tekhnosfera.com/primenenie-neuronnyh-setey-dlya-postroeniya-adaptivnyh-sistem-upravleniya-tehnologicheskimi-protsessami#ixzz6q0IpkIxO>
5. Типы нейронных сетей. Принцип их работы и сфера применения А. Павленко URL: <https://otus.ru/nest/post/1263/>
6. Нейронные сети URL: <http://statsoft.ru/home/textbook/modules/stneunet.html>
7. Использование нейронных сетей в тестировании сложных программных систем Бабин Д. В. URL: [https://www.okbsapr.ru/library/publications/babin\\_tezisy2013/](https://www.okbsapr.ru/library/publications/babin_tezisy2013/)
8. Обучение нейросети с учителем, без учителя, с подкреплением – в чём отличие? Какой алгоритм лучше? К. Беликова URL: <https://neurohive.io/ru/osnovy-data-science/obuchenie-s-uchitelem-bez-uchitelja-s-podkrepleniem/>

# СИСТЕМА МОНІТОРИНГУ В ЕЛЕКТРОННІЙ МЕДИЧНІЙ СИСТЕМІ

Музика О.А.<sup>1</sup>, Булах Б.В.<sup>2</sup>

ІПСА, КПІ ім. Ігоря Сікорського, Київ, Україна

<sup>1</sup>oleksandr.muzyka.a@gmail.com, <sup>2</sup>bogdan.bulakh@gmail.com

**Предметом дослідження є проблематика побудови системи дистанційного медичного моніторингу. Актуальність теми визначається потребою пошуку рішення для організації збору, зберігання та візуалізації медичних даних в моніторинговій системі, яка буде інтегрована в загальну інформаційну систему охорони здоров'я. В даній роботі розглянуто основні пристрої та підходи для проведення моніторингу, які використовуються як в клінічній практиці, так і для повсякденних потреб. Наведено огляд існуючих підходів до організації локального моніторингу в медичних закладах та підходи до організації дистанційного моніторингу. Враховуючи переваги та недоліки оглянутих архітектур, запропоновано концепцію моніторингової системи, яка може бути інтегрована в комплексну медичну систему.**

**Ключові слова:** медичний моніторинг, інтернет речей, телемедицина, системи моніторингу.

## 1. ВСТУП

Останні здобутки інформаційних та комп'ютерних технологій в області інтернету речей, бездротових мереж, портативних девайсів та мініатюрних сенсорів дають простір для розробки методів дистанційного моніторингу пацієнтів, аналізу великої кількості медичних метрик для клінічних досліджень та підбору персоналізованого лікування, ефективного амбулаторного та постклінічного супроводу пацієнта тощо. Даний вектор розвитку є актуальним, адже попри розширення технологічних можливостей біомедичної науки та покращення розуміння патологічних процесів та методів їх лікування, проблеми обмеженості ресурсів сфери охорони здоров'я, які виливаються в економічні та логістичні бар'єри в доступі до медичних послуг, як в країнах, що розвиваються [1], так і в розвинених країнах [2], залишаються невирішеними. Часткова автоматизація процесів діагностики, супроводу пацієнтів, створення доступних широкому загалу носимих девайсів та розвиток систем моніторингу для контролю здоров'я може зарадити цій проблемі шляхом цифровізації процесів, які вимагають дорогого часу медичних працівників та знизить необхідність особистої присутності в медичних закладах для діагностики та обстеження. Також впровадження сенсорів в повсякденне життя може стати джерелом великої кількості даних, які матимуть змогу використовувати аналітики та фахівці з машинного навчання для побудови прогностичних моделей та дати додаткові інсайти щодо здоров'я населення в статистичному перебізі та особливостей клініки окремих захворювань. [3]

## 2. МОНІТОРИНГОВІ ПРИСТРОЇ ТА ДАНІ

В контексті розробки ПЗ для моніторингу за станом здоров'я пацієнтів можна виділити дві категорії приладів та систем, які надають інформацію про ті чи інші показники: клінічні моніторингові системи та портативні девайси.

До першої категорії відносяться спеціалізовані прилади для вимірювання окремих показників та комплексні системи, які вимірюють ряд показників та мають базове ПЗ для аналізу даних: сигналізація про критичний стан, співставлення декількох показників для виявлення конкретної проблеми. Як правило, такі системи та прилади постійно використовуються в інтенсивній терапії та під час оперативного втручання чи інших медичних процедур. До них відносять інвазивні (монітор внутрішньочерепного тиску чи монітор для катетеру легенової артерії, втрачає популярність в медичній практиці) та неінвазивні кардіологічні монітори (пульсометри, ЕКГ, монітор Доплера), аналізатори крові (лейкоцитарної формули, рівню вмісту окремих речовин), респіраторні монітори та інші. [4]

До другої категорії відносять спеціальні прилади для зчитування метрик з подальшою передачею їх в хмарне сховище та універсальні пристрої загального призначення, що мають функціонал для проведення моніторингу. Окремі спеціалізовані прилади можуть використовуватись при заняттях спортом для контролю пульсу, на регулярній основі в побуті для контролю здоров'я (наприклад, прилади для вимірювання тиску, пульсоксиметри, термометри, глюкометри та ін.) а також можуть тимчасово видаватись пацієнтам для діагностики (наприклад, монітор Холтера дає зафіксувати серцевий ритм протягом доби в різних ситуаціях, як-то в стані спокою, під час сну, під час фізичної активності та ін.). [5] Також варто відзначити універсальні пристрої, наприклад смартфон, які, за наявності ПЗ та технічного функціоналу, дають змогу проводити моніторинг: вимірювання пульсу за допомогою відеофіксації, логування рутинних звичок (контроль раціону, контроль прийому медикаментів), трекінг фізичної активності (зокрема ходьби, за допомогою GPS) тощо.

Таке різноманіття ПЗ та приладів для моніторингу породжує велику кількість даних, аналіз яких значною мірою ускладнює їх великі обсяги та варіативність структури. Природу та характер даних, які фігурують в таких приладах та системах, варто розглянути більш детально.

По-перше, в залежності від цільової метрики відрізняється тип та структура моніторингових даних. Частина даних за своєю природою є цілочисельним типом. До таких даних відносяться показники пульсу, кількості подихів за хвилину, для більшості приладів артеріальний тиск та ін. Також є показники, які описуються дійсним числом, як-то показники в лейкоцитарній формулі, рівень глюкози в крові, оксигенація, тиск. Інші дані можуть бути булеву природу, категоріальну (де показник описується як одне якісне значення з обмеженої кількості можливих), можуть бути описані зображенням чи відеорядом, мати складну багаторівневу структуру (наприклад, зведений звіт декількох показників), або структура може бути відсутня зовсім (наприклад, суб'єктивне відчуття пацієнта, яке він зазначає кожен день в мобільному додатку).

По-друге, такі дані відрізняються періодичністю замірів. В інтенсивній терапії ключові вітальні показники вимірюються безперервно з певним проміжком часу (одна чи декілька секунд); при супроводі пацієнта, який приймає медикаменти з серйозними побічними ефектами, може братись аналіз крові щодня чи раз на декілька днів; для пацієнтів з онкологічним захворюванням в анамнезі може раз на декілька тижні проводитись тест на рівень онкомеркерів та/або раз на рік чи пів року проводитись КТ/МРТ.

По-третє, моніторингові дані відрізняються періодом своєї актуальності. Приміром, показники в інтенсивній терапії є актуальними лише в околі критичного стану, аби лікар мав змогу його коректно визначити; виміри тиску при прийомі препаратів для визначення їх ефективності на пореби корегування актуальні в контексті курсу прийому препаратів; результати КТ/МРТ при веденні пацієнта з онкологією в ремісії актуальні протягом декількох років, аби відслідковувати прогрес захворювання.

Отже в контексті розробки моніторингових систем варто враховувати природу даних для розуміння можливих опцій представлення даних на моніторингових панелях, періодичності замірів для визначення коректних способів отримання даних з джерела та методів роботи з великою кількістю даних, та “термін життя” даних, аби правильно організувати збереження даних.

Наприклад, представлення зміни даних протягом часу на графіку актуальне тільки для чисельних даних. Для булевих та категоріальних можливо робити зведену статистику за певний період часу (кругові чи стовпчикові діаграми). Для даних зі складною багаторівневою структурою потрібно визначати алгоритми аналізу (наприклад, обрахунок якоїсь цільової метрики з масиву різних даних, чи виділення тільки одного чи декількох показників простої структури). Для неструктурованих даних доцільним буде тільки логування чи представлення у форматі таблиць.

### **3. АРХІТЕКТУРА МОНІТОРИНГОВИХ СИСТЕМ В МЕДИЧНІЙ СФЕРІ ТА ДЛЯ ЗАГАЛЬНОВЖИВАНИХ ДЕВАЙСІВ**

Загалом в медичній сфері використовуються локальні моніторингові системи (ПЗ, яке працює в межах локальної мережі медичного закладу) та системи для дистанційного моніторингу.

Попри значні досягнення в комп'ютерних науках, організація моніторингової інфраструктури в медичних закладах досі є складною задачею. Як зазначається в [6], головна проблема полягає у відсутності чітких стандартів інтерфейсів та формату даних, ліцензійні обмеження, несумісність приладів та відсутність можливості інтеграції деяких приладів в цифрову систему з технологічної точки зору. Через це єдина система, яка агрегує всі дані відсутня. З деякими даними доводиться працювати мануально (наприклад, показники відображаються на індикаторі приладу), деякі дані складно структурувати для ретроспективного аналізу, а для деяких категорій приладів треба використовувати різне ПЗ та орієнтуватися в великій кількості моніторів одночасно. В архітектурному плані якогось єдиного стандарту організації мережі приладів немає, однак над цим ведеться активна робота та створюються прототипи єдиних систем. Зокрема, “The Integrated Medical Environment”, описана в статі. В архітектурному плані ПЗ — це єдиний інтерфейс для отримання даних з різних сховищ, як то БД лабораторії, БД з медичними візуалізаціями та ін.

Прилади для дистанційного моніторингу в загальному випадку стикаються з такими ж проблемами. Однак, тут варто розрізняти дві категорії таких приладів: прилади, які розробляються для повсякденного вжитку (фітнес-браслети, пульсоксиметри, електронні ваги тощо) та спеціалізовані прилади для медичного використання.

Для перших, як правило, розробляється якісне програмне забезпечення, яке дозволяє на смартфонах та у веб-додатках візуалізувати дані, робити аналітику. В межах одного бренду часто можна збирати дані в одному додатку і формувати більш комплексну статистику. До того ж розробники спеціально створюють API для того, щоб вони могли з легкістю інтегруватись в інші додатки, як, наприклад, Google Fitness. Загалом архітектуру таких приладів можна описати як клієнт-серверний додаток, де клієнт — мобільний додаток, а сервер — система та сховище в хмарі. Обчислення можуть відбуватись як локально, так і в хмарі. Окремі девайси під'єднані до телефону, через який передають інформацію в систему.

При використанні спеціалізованих приладів для дистанційного моніторингу підхід трохи інший, адже дані збираються з приладів, під'єднаних до пацієнта, а аналізує дані медичний працівник. В [7] дані із сенснів за допомогою брокерів повідомлень передаються до сховища даних, де вони зберігаються для подальшого аналізу. Окремий додаток

звертається в БД, аби проаналізувати дані та надати звіт. В [8] наведена архітектура більш схожа на моніторингові девайси та додатки, які використовуються в широкому загалі. Група сенсорів під'єднана до смартфона чи планшета, де в мікроконтрольному інтерфейсі попередньо обробляється та пріоритезується. Після чого в зашифрованому вигляді надсилається до mobile gateway через WiFi або Bluetooth (в залежності від відстані) Після чого дані потрапляють до хмарного середовища, де відбуваються відбopідні обчислення (аналіз даних за допомогою нейронних мереж та побудова прогнозів), до результатів яких медичний працівник матиме доступ через клієнський застосунок.

#### **4. ПРОБЛЕМАТИКА АКТУАЛЬНИХ МОДЕЛЕЙ ТА КОНЦЕПЦІЯ ІНТЕГРАЦІЇ СИСТЕМИ МОНІТОРИНГУ В ЕЛЕКТРОННУ МЕДИЧНУ СИСТЕМУ**

В попередньому розділі розглянуто підходи, які використовуються для створення моніторингових систем при використанні девайсів-сенсорів для широкого загалу, організації медичних моніторингових систем на локальному рівні та побудови систем дистанційного моніторингу для медичних закладів. Проблема моніторингових систем для особистого користування в тому, що вони оптимізовані під використання однією людиною в особистих цілях, а хмарне сховище даних скоріше використовується як резервна копія даних. Огляд проблематики побудови системи моніторингу на локальному рівні висвічує проблеми, з якими потенційно можна стикнутися при інтеграції моніторингової системи в екосистему мікросервісів медичної системи, адже однією з потенційних задач є робота з моніторинговими пристроями.

Головний недолік запропонованих архітектур, які використовуються для побудови дистанційного моніторингу, полягає в тому, що всі дані проходять через централізоване хмарне сховище. Така концепція є прийнятною для розробки моніторингової системи, яка використовується для візуалізації даних в ретроспективі, однак для трансляції даних в режимі реального часу призведе до значних затримок та просідань з точки зору швидкодії системи.

Однак, повністю позбутися централізованого хмарного сховища, яке є джерелом моніторингових даних, не варто. Ба більше, в системі моніторинга воно може покрити значну частину потреб в наданні інформації на відображення на моніторинговій панелі, позбавивши необхідності адаптовувати сервіс під API кожного окремого девайсу чи групи девайсів.

Оптимальним рішенням проблеми буде використання патерну Адаптер для роботи з різними джерелами даних. Для роботи з центральним сховищем функція адаптеру буде скоріше номінальною, адже дані, які надаються по API загалом відповідатимуть формату даних, який необхідний для представлення на панелі. А для інших джерел даних чи груп джерел даних можна створити окремі адаптери, які зводитимуть дані до стандартного формату, готового для представлення на панелях.

Для кожного такого адаптера має конфігуруватись чіткий перелік індикаторів, які можна створити за допомогою даних з джерела. Очевидно, наприклад, що при прямому підключенні до сервіса, який надає лише останній замір пульсу, без зберігання даних в центральному датасховищі побудувати графік зміни пульсу за останні 2 години буде неможливим.

Також центральне сховище можна оптимізувати, зробивши його тимчасовим хабом даних. Дані, які актуальні лише пару родин, немає сенсу зберігати в БД декілька місяців. Можна розбити дані на декілька категорій за їх "терміном життя" та періодично очищати БД від застарілих даних. В т.ч. для аналітики можна створити спеціальний API, за допомогою якого можна отримати деякі дані та зберегти в постійному сховищі для подальшого

статистичного аналізу чи використання як навчальних вибірок для побудови моделей машинного навчання. З таким підходом база не буде швидко розростатись, що через деякий час призведе до зниження ефективності.

## 5. ВИСНОВКИ

В даній роботі було розглянуто пристрої, які використовуються для медичного моніторингу, на локальному рівні в медичних закладах та для дистанційного моніторингу. Для таких пристроїв наведено огляд природи даних, з якими потрібно взаємодіяти при організації моніторингу стану здоров'я. Також було розглянуто наявні архітектурні рішення для організації сенсорних пристроїв, хмарних застосунків так лієнтів в єдину архітектуру. На основі наявних рішень запропоновано концепцію системи моніторингу в екосистемі мікросервісів загальної системи віртуального кабінету лікаря, де централізоване хмарне сховище поєднується з можливістю роботи з сервісами напряму за допомогою адаптерів та періодична очистка застарілих даних з централізованого хабу (періодичність визначається “терміном актуальності” даних), який використовується як основне джерело.

## ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Poverty and Access to Health Care in Developing Countries / D. H. Peters та ін. *Annals of the New York Academy of Sciences*. 2008. Т. 1136, № 1. С. 161–171. URL: <https://doi.org/10.1196/annals.1425.011> (дата звернення: 14.11.2022).
2. Cylus J., Papanicolas I. An analysis of perceived access to health care in Europe: How universal is universal coverage?. *Health Policy*. 2015. Т. 119, № 9. С. 1133–1144. URL: <https://doi.org/10.1016/j.healthpol.2015.07.004> (дата звернення: 14.11.2022).
3. Machine Learning for Healthcare Wearable Devices: The Big Picture / F. Sabry та ін. *Journal of Healthcare Engineering*. 2022. Т. 2022. С. 1–25. URL: <https://doi.org/10.1155/2022/4653923> (дата звернення: 17.11.2022).
4. Berry C. Monitoring and Testing the Critical Care Patient - *Critical Care Medicine - MSD Manual Professional Edition*. MSD Manual Professional Edition. URL: <https://www.msmanuals.com/professional/critical-care-medicine/approach-to-the-critically-ill-patient/monitoring-and-testing-the-critical-care-patient> (дата звернення: 18.11.2022).
5. A Review of Wearable Internet-of-Things Device for Healthcare / N. Surantha та ін. *Procedia Computer Science*. 2021. Т. 179. С. 936–943. URL: <https://doi.org/10.1016/j.procs.2021.01.083> (дата звернення: 18.11.2022).
6. Information Technology in Critical Care: Review of Monitoring and Data Acquisition Systems for Patient Care and Research / M. A. De Georgia та ін. *The Scientific World Journal*. 2015. Т. 2015. С. 1–9. URL: <https://doi.org/10.1155/2015/727694> (дата звернення: 14.11.2022).
7. Vitabile, S. et al. (2019). Medical Data Processing and Analysis for Remote Health and Activities Monitoring. In: Kołodziej, J., González-Vélez, H. (eds) *High-Performance Modelling and Simulation for Big Data Applications*. Lecture Notes in Computer Science, vol 11400. Springer, Cham. [https://doi.org/10.1007/978-3-030-16272-6\\_7](https://doi.org/10.1007/978-3-030-16272-6_7) (дата звернення: 17.11.2022).
8. Iranpak S., Shahbahrami A., Shakeri H. Remote patient monitoring and classifying using the internet of things platform combined with cloud computing. *Journal of Big Data*. 2021. Т. 8, № 1. URL: <https://doi.org/10.1186/s40537-021-00507-w> (дата звернення: 18.11.2022).

# ОРГАНІЗАЦІЯ ЗБОРУ ДІАГНОСТИЧНИХ ПОКАЗНИКІВ В СИСТЕМАХ КРАЙОВИХ ОБЧИСЛЕННЯХ

Охота Д.Л.<sup>1</sup>, Булах Б.В.<sup>2</sup>

Національний технічний університет України  
«Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна

<sup>1</sup>qwerty165343@gmail.com [0000-0001-8364-1900], <sup>2</sup>bogdan.bulakh@gmail.com  
[0000-0001-5880-6101]

**Розглядається задача аналізу процесу організації систем крайових та туманних обчислень медичних показників в трьохрівневій архітектурі. Завдяки впровадженню наявних інструментів, буде реалізовано швидке впровадження різного роду рішень моніторингу стану суб'єктів охорони здоров'я. Буде оцінено вплив такого підходу на ефективність збору та опрацювання показань датчиків .**

**Ключові слова: крайові обчислення, інфраструктура, крайовий пристрій, Інтернет речей.**

## 1. ВСТУП

Граничні обчислення являють собою технологічну пропозицію, яка намагається вирішити обмеження мережевої затримки, безпекової складової, використовуючи зростаючу конвергенцію між хмарними і мережевими інфраструктурами та забезпечуючи основу для еволюції до конвергентного прозорого середовища обробки інформації, здатного підтримувати розподілені, мобільні додатки, що вирішують набагато ширший спектр завдань, і стають ключовим інструментом для втілення обіцянок щодо мереж наступного покоління (5G і далі) в реальність.

Стрімкий розвиток Інтернету речей (IoT) трансформувалася бізнес та обслуговування клієнтів у багатьох аспектах сучасного життя завдяки повсюдному використанню сенсорних і обчислювальних можливостей мобільних пристроїв, таких як смартфони, фітнес-браслети, ноутбуки та планшети.

Велика тенденція використання “розумних” речей спостерігається у сфері охорони здоров'я, де відслідковування усіх життєвих показників стану людини, грає значну роль у визначенні аномалій роботи організму, хвороб. Що за собою призводить до великою утилізації інтернет каналів провайдерів мережеских послуг. Тому у даній роботі розглядається специфіка організації роботи систем граничних обчислень з використанням медичних вимірювальних пристроїв.

## 2. КРАЙОВІ ТА ТУМАННІ ОБЧИСЛЕННЯ

### 2.1. Визначення

Крайові обчислення - це парадигма розподілених обчислень, що здійснюються в межах досяжності кінцевих пристроїв. Цей тип обчислень використовується для скорочення часу мережного відгуку, а також ефективнішого використання пропускну здатності мережі.

Туманні обчислення — це модель, яка забезпечує обчислення та зберігання даних між кінцевими пристроями та традиційними центрами хмарних обчислень. Туманні обчислення — концепція, за якою частина даних оброблюється в локальних мережах, а не виключно в дата-центрі. Будь-який пристрій, що має обчислювальні здібності, сховище та мережу

підключення, може бути вузлом туману. Приклади пристроїв включають промислові контролери, комутатори, маршрутизатори, вбудовані сервери та камери відеоспостереження [1].

Сервіси додатків, які використовують крайові обчислення, скорочують обсяги даних, які мають бути передані, наступний трафік та відстань, яку мають пройти дані. Ця архітектура обчислень забезпечує меншу затримку відгуку всередині мережі та знижує витрати на обмін даними. Розвантаження обчислень у додатках, які виконуються у реальному часі (алгоритми розпізнавання осіб) з допомогою цієї технології, показала значну ефективність у сфері поліпшення часу відгуку мережі, що було продемонстровано у ранніх досліджах. Подальші дослідження показали, що використання хмаринок (cloudlets) у мережі користувачів мобільних додатків, що пропонують послуги, зазвичай що у хмарі, забезпечує скорочення часу виконання операцій, коли деякі завдання вивантажуються на граничний вузол. З іншого боку, розвантаження кожної оброблюваної задачі може призвести до уповільнення обміну даними через збільшення часу передачі між кінцевими пристроями і вузлами.

Інші можливі застосування технології включають управління «розумними» системами автомобілів, створення «розумних» міст з розвиненою мережевою інфраструктурою, проекти Індустрії 4.0, а також використання граничних обчислень в системах охорони здоров'я.

## 2.2. Загальна схема

Туманні обчислення - це обчислювальний шар між хмарою і периферією. Там, де периферійні обчислення можуть відправляти величезні потоки даних безпосередньо в хмару, туманні обчислення можуть отримувати дані з периферійного шару до того, як вони потраплять в хмару, а потім вирішувати, які з них є важливими, а які ні. Релевантні дані зберігаються в хмарі, в той час як нерелевантні дані можуть бути видалені або проаналізовані на рівні туману для віддаленого доступу або для інформування локалізованих моделей навчання.

Прикладом такої роботи туманних обчислень може слугувати медичний додаток пацієнтів, які знаходяться віддаленому лікуванні, де датчик температури, підключений до периферійного сервера, вимірює температуру щосекунди. Потім ці дані передаються в хмарний додаток для моніторингу температурних стрибків. Уявіть собі, що всі вимірювання температури, кожна секунду циклу вимірювань 24/7, відправляються в хмару.

У випадку з туманним шаром, крайовий пристрій спочатку надсилає дані до туманного шару через локальну мережу. Туманний сервер отримував би ці дані і, відповідно до певних параметрів, вирішував би, чи варто їх відправляти в хмару. Кінцевий результат - зменшення трафіку та часу відповіді. Для простих вимірювань показників здоров'я така економія даних може здатися незначною. Але уявіть, що ви постійно передаєте складну інформацію або великі файли, наприклад, зображення чи відео. Вплив на пропускну здатність і затримку може бути величезним в залежності від програми та пристрою.

У випадку розгортання системи моніторингу стану пацієнта, важливими є характеристики: реакції на подію, що відбулась на краю, відповідність нормативним вимогам, захищеність від несанкціонованого доступу. Також при реалізації такої моделі є можлива відсутність зв'язку із центральною хмарою, у разі чого система буде працювати у штатному режимі [2].

Така архітектура (Рис. 1) гарантує доступність крайових пристроїв до інструментів зв'язку з вузлами підтримки системи, операційною складовою та моніторинговими та контролюючими складовими. Рішення даного типу передбачає його використання в доволі малій географічній зоні, що має достатній рівень мережевого забезпечення.

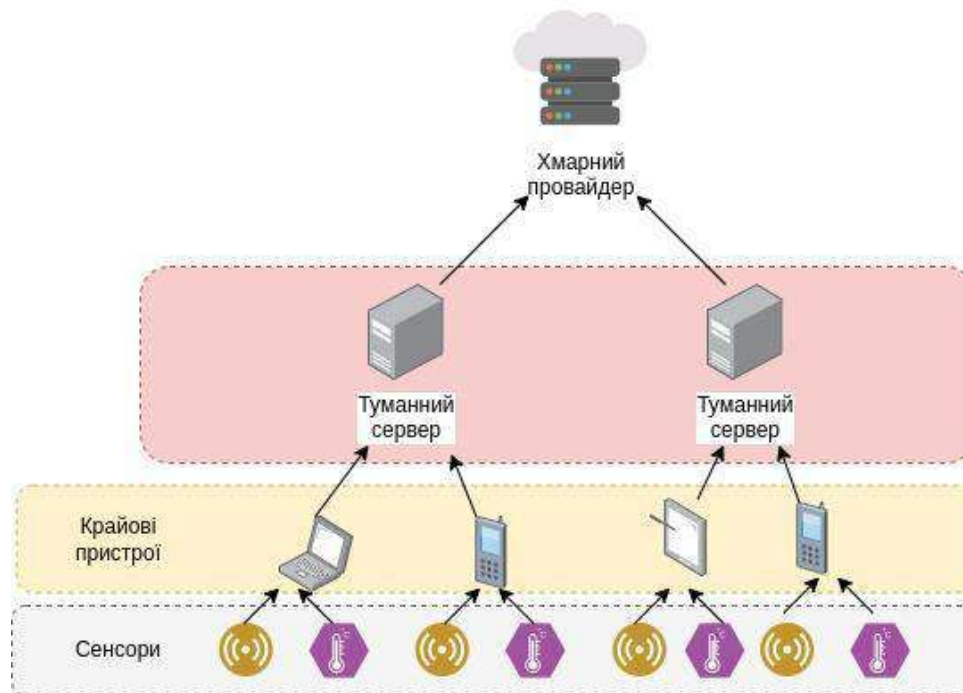


Рисунок 1. Організаційна схема крайових обчислень

### 2.3. Поведінкова схема

Поведінкова схема (Рис. 2) описує увесь алгоритм зв'язку процесів що відбувається на кожному із рівнів системи.

На нижньому рівні (рівні крайових пристроїв) розташовуються кінцеві пристрої (датчики), а також крайові пристрої та шлюзи. Цей шар також включає програми, які можуть бути встановлені на кінцеві пристрої для розширення їхньої функціональності. Елементи цього шару використовують наступний шар, мережу, для зв'язку між собою, а також між ними та хмарою. Рівень туманних пристроїв містить обчислювальні вузли, які підтримують управління ресурсами та обробку IoT завдань, що далі надходять у хмару. Поверх туманного рівня розташовується програмне забезпечення для управління ресурсами, яке керує всією інфраструктурою та забезпечує якість обслуговування додатків туманного обчислення [3].

Трирівнева архітектура для інфраструктури інтелектуальної охорони здоров'я, що складається з рольової моделі, багаторівневої хмарної архітектури та прошарку туманних обчислень, забезпечує ефективність для додатків охорони здоров'я та догляду за літніми людьми. Рівень туманних обчислень удосконалює архітектуру, забезпечуючи низьку затримку, підтримку мобільності, та безпеку. Відображення інформації відбувається на пристрої з допомогою сервіс-орієнтованого підходу [4].

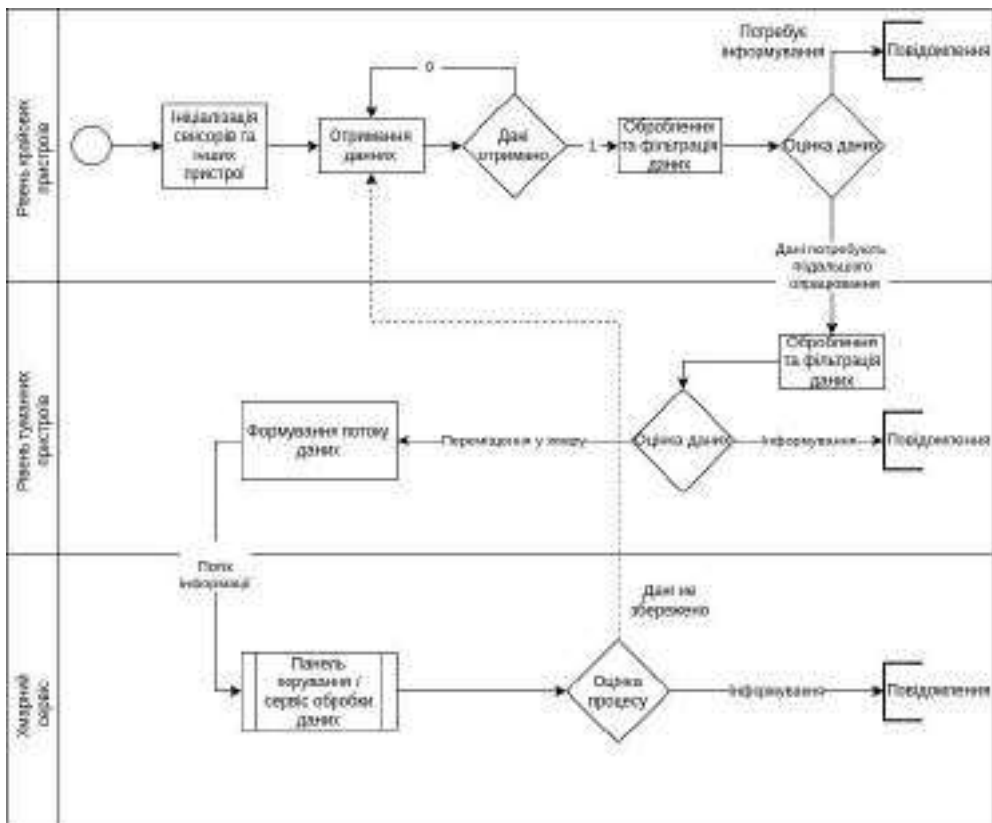


Рисунок 2. Поведінкова схема

### 3. МЕТОДИ ТА ІНСТРУМЕНТАРІЙ

#### 3.1. Моделі розгортання систем

Розробка систем крайових обчислень передбачає реалізацію різного роду методологій та принципів розгортання обчислювальних вузлів. Таким чином розглянуті нижче засоби виокремлюють та удосконалюють роботу усіх одиниць середовища.

Наприклад **Cloudlet** технологія представляє собою довірених, багатий на ресурси комп'ютер або кластер комп'ютерів, добре підключений до Інтернету і доступний для прилеглих мобільних пристроїв. Він удосконалює початкову дворівневу архітектуру "мобільний пристрій-хмара" мобільних хмарних обчислень до трирівневої архітектури "мобільний пристрій-cloudlet -хмара". При цьому Cloudlet може обслуговувати користувачів як незалежну хмару, перетворюючись на "маленьку хмару" або "центр обробки даних у коробці". Подібно до точок доступу WiFi, Cloudlet може бути розгорнутий у зручному місці (наприклад, у ресторані, кафе або бібліотеці). Декілька Cloudlet можуть утворювати розподілену обчислювальну платформу, яка може ще більше розширити доступні ресурси для мобільних пристроїв. Оскільки Cloudlet знаходиться лише в одному стрибку від мобільних пристроїв користувачів, він покращує QoS за рахунок низької затримки зв'язку та високої пропускнуєї спроможності [5].

**CloudPath** – це система крайових обчислень, запропонована Університетом Торонто. У такій системі різні ресурси такі як обчислення та зберігання, надаються на шляху від пристрою до хмарного центру обробки даних. Вона підтримує розподіл на вимогу розподіл та динамічне розгортання багаторівневої архітектури. Основна ідея CloudPath полягає в реалізації так званих "обчислень по дорозі", що дозволяє скоротити час відгуку та покращити

використання пропускної спроможності. час і поліпшити використання пропускної спроможності порівняно зі звичайними хмарними обчисленнями.

**SpanEdge** – потокова обробка є одним із важливих типів додатків у крайових обчисленнях, де дані генеруються різними джерелами даних джерелами даних у різних географічних точках і безперервно передаються у вигляді потоків. Традиційно всі необроблені дані передаються глобальною мережею на сервер центру обробки даних, а системи обробки потоків, такі як Apache Spar. Системи обробки потоків, такі як Apache Spark та Flink, також розроблені та оптимізовані для одного централізованого центру обробки даних. Однак такий підхід не може ефективно обробляти величезні дані, що генеруються безліччю пристроїв на краю мережі, і ситуація стає ще гіршою, коли програми вимагають низька затримка та передбачуваність [6].

Перелічені вище технології володіють рівнозначними характеристики реалізації у трьохрівневих архітектурах крайових обчислень. Але зі спробами зменшення потоку інформації на центральні вузли хмар, з'являється проблема перевантаження крайових пристроїв, що ймовірно може спричинити відмову деяких із них. Тому було розроблено ряд програмних рішень для розвантаження та одночасно ефективного використання обчислювальних вузлів, які знаходяться між крайовими пристроями та хмарою.

### 3.2. Програмні рішення

Системи моніторингу стану пацієнтів, передбачають використання широкого спектру програмних рішень, які вирішують задачі: розпізнавання аномалій роботи організму, зберігання даних, оброблення та відображення. Але такий перелік програмного забезпечення із збереження повного його функціоналу неможливо розгорнути на малопотужних крайових чи туманних пристроях. Для рішення цієї проблеми було спроектовано безліч рішень у сфері глибокого навчання, моніторингу та т.п.

Для підтримки обробки даних за допомогою моделей глибокого навчання на периферії було випущено кілька периферійних фреймворків та інструментів глибокого навчання. Такими є: TensorFlow Lite, Caffe2, PyTorch, MXNet, CoreML та TensorRT.

TensorFlow Lite — це набір інструментів, який забезпечує машинне навчання на пристрої, допомагаючи розробникам запускати свої моделі на мобільних, вбудованих та периферійних пристроях [7].

Caffe2 — це фреймворк для глибокого навчання, який забезпечує простий і зрозумілий спосіб експериментувати з глибоким навчанням і використовувати внесок спільноти в нові моделі та алгоритми. Надає можливість масштабувати архітектуру, використовуючи потужність графічних процесорів у хмарі або в масах на мобільних пристроях за допомогою крос-платформних бібліотек Caffe2.

PyTorch — це фреймворк машинного навчання, заснований на бібліотеці Torch, що використовується для таких додатків, як комп'ютерний зір і обробка природної мови, спочатку розроблений компанією Meta AI, а тепер входить до складу Linux Foundation. Це безкоштовне програмне забезпечення з відкритим вихідним кодом, випущене під модифікованою ліцензією BSD. Хоча інтерфейс Python є більш відшліфованим і основним напрямком розвитку, PyTorch також має інтерфейс C++.

MXNet — це фреймворк глибокого навчання з відкритим вихідним кодом, який дозволяє визначати, навчати та розгортати глибокі нейронні мережі на широкому спектрі пристроїв, від хмарної інфраструктури до мобільних пристроїв. Вона має високу масштабованість, що дозволяє швидко навчати моделі, а також підтримує гнучку модель програмування і кілька мов.

TensorRT — це SDK для високопродуктивного глибокого навчання, включає в себе оптимізатор глибокого навчання і середовище виконання, яке забезпечує низьку затримку і високу пропускну здатність додатків для виводу.

Таблиця 1. Порівняння бібліотек машинного навчання

Назва	Підтримка мов	Типи завдань	Характеристика	Розподіленість	Багатопроекторність	Складність у розгортанні	Операційні системи	Цільовий пристрій
TensorFlow Lite	Java, Swift, Objective-C, C++, Python	Прийняття рішень	Латентність	Наявна	Середня	Легко	Android та iOS, вбудовані Linux та мікроконтролери	Мобільні та вбудовані пристрої
Caffe2	C++, Python	Прийняття рішень, тренування	Легкість, модульність, масштабованість	Наявна	Мінімальна	Складна	MacOS; Ubuntu; CentOS; Windows; Android; Raspberry Pi	Багатоформатні
PyTorch	C++ Python	Прийняття рішень, тренування	Легкість у дослідженні	Наявна	Середня	Легко	Linux MacOS Windows	Багатоформатні
MXNet	C++, JavaScript, Python, R, Matlab, Julia, Scala, Clojure, Perl	Прийняття рішень, тренування	Великий вибір модулів	Наявна	Повна	Середня	Linux MacOS Windows мікроконтролери	Багатоформатні
CoreML	-	Прийняття рішень	Використання пам'яті та потужність	Невідомо	Повна	Легка	iOS	Apple
TensorRT	C++ Python	Прийняття рішень	Латентність та пропускну здатність	Відсутня	Повна	Середня	Linux Windows	NVIDIA GPU

### 3.3. Рекомендації щодо процесу розгортання

При розгортанні подібних систем варто виконати наступні кроки:

- Об'єм інтелекту IoT-пристроїв.

Чим більше аналітичних дій виконується на пристрої, тим менше кількість операцій виконується на хмарних серверах. Адже дані вже фільтруються в джерелі — пристрої IoT. Інтелектуальні, стандартизовані пристрої IoT забезпечать менші обсяги даних в більш легко керованих форматах.

- Групування IoT пристроїв

Об'єднування в групи виконується за декількома критеріями. Поділ на області можна виконати, в залежності, які властивості були надані пристроям у першому кроці. Або ж у

відповідності до географічного положення, це значно зменшить затримки та навантаження на мережу.

- Визначення результируючих даних

Розробка хорошого механізму правил та оновлення його за необхідності, щоб відобразити, як змінюються пріоритети і потреби у виконанні тих чи інших задач. Пріоритетом також є інвентаризація середовища в режимі реального часу, щоб гарантувати, що туманні сервери працюють в поточному середовищі.

- Використання hub-spoke підходу

Для управління потоком необхідних даних необхідно мати крайову інфраструктуру, яка складається з різних серверів, розміщених в мережі з ієрархічним способом роботи з даними між ними. Оптимальним способом роботи з такою складною системою є використання найдешевших, найменш інтелектуальних крайових серверів - це умовне використання термінології, ці системи можуть бути досить інтелектуальними і дорогими самі по собі - якомога ближче до пристроїв IoT.

- Використовувати розширену аналітику даних та звітність.

Необхідно уникати помилкових спрацьовувань і негативних сигналів, а можливі шляхи виправлення ситуації повинні бути показані будь-якій людині, яка бере участь в цьому процесі. Тому не варто економити на аналітичних інструментах, що використовуються, і переконайтесь, що звітність ведеться чітко і змістовно.

#### 4. ВИСНОВКИ

Екосистема крайових обчислень дуже динамічна, у ній з'являється безліч нових ініціатив від різних організацій та компаній. Тільки в області відкритих кодів реалізується не менше 20 ініціатив. Поки що немає узгодженого галузевого стандарту, що охоплює всі аспекти крайових обчислень, навіть після багаторічних зусиль деяких органів зі стандартизації.

Навіть на стадії розвитку крайових обчислень, виникає велика кількість програмних та архітектурних рішень, що в повній мірі використовують усі переваги середовища, у якому їх розгорнуто. Так чудовим прикладом слугують системи "розумного" міста, моніторингу роботи сонячних панелей, система управління робочими вузлами на виробництвах, модернізується сфера охорони здоров'я.

У роботі розглядається архітектурні та програмні рішення, які є ефективним у медицині. Так на основі роботи кінцевих пристроїв, якими являються: медичне обладнання, різного роду сенсори та датчики; було реалізовано трьохрівнену архітектуру з опрацюванням показників життєдіяльності та їх подальшу передачу у центральні вузли як туманного рівня так і хмари. Така методологія, організовує швидку реакцію системи на події, про які сигналізують кінцеві пристрої, навіть в умовах недостатнього доступу до інтернет-мережі.

#### ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Edge Computing. URL: [https://en.wikipedia.org/wiki/Edge\\_computing](https://en.wikipedia.org/wiki/Edge_computing) (дата звернення: 11.11.2022)
2. OpenFog Reference Architecture for Fog Computing. URL: [https://www.iiconsortium.org/pdf/OpenFog\\_Reference\\_Architecture\\_2\\_09\\_17.pdf](https://www.iiconsortium.org/pdf/OpenFog_Reference_Architecture_2_09_17.pdf) (дата звернення: 11.11.2022)
3. Smart Items, Fog and Cloud Computing as Enablers of Servitization in Healthcare URL: [https://www.researchgate.net/publication/284430697\\_Smart\\_Items\\_Fog\\_and\\_Cloud\\_Computing\\_as\\_Enablers\\_of\\_Servitization\\_in\\_Healthcare](https://www.researchgate.net/publication/284430697_Smart_Items_Fog_and_Cloud_Computing_as_Enablers_of_Servitization_in_Healthcare) (дата звернення: 12.11.2022)

4. A. Krolczyk, V. Stantchev, C. Senf, Service-oriented approaches for e-government, in Proceedings of the 11th International Conference on Information Integration and Web-based Applications & Services, 2009, pp. 441–446.
5. Cloudlet. URL: <https://en.wikipedia.org/wiki/Cloudlet> (дата звернення: 12.11.2022)
6. SpanEdge: Towards Unifying Stream Processing over Central and Near-the-Edge Data Centers. URL: <https://ieeexplore.ieee.org/document/7774704> (дата звернення: 12.11.2022)
7. TensorFlow Lite. URL: <https://www.tensorflow.org/lite/guide> (дата звернення: 12.11.2022)

# ВИРІШЕННЯ ЗАДАЧІ ГРАФЕМНО-ФОНЕМНОГО ПЕРЕТВОРЕННЯ ТЕКСТІВ

Самвелян А.Р.<sup>1</sup>, Кисельов Г.Д.<sup>2</sup>

Національний технічний університет України  
«Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна

<sup>1</sup>yourcamelcase@gmail.com [0000-0001-8364-1900], <sup>2</sup>g.kyselov@gmail.com

**Розглядається методи вирішення задачі графемно-фонемного перетворення. Буде визначена математична модель задачі, наведені методи вирішення даної задачі та їх опис.**

**Ключові слова: графема, фонема, нейромережева модель, синтез мовлення.**

## 1. ВСТУП

Процес графемно-фонемного (G2P) перетворення генерує фонетичну транскрипцію з письмової форми слів. Написання слова називається послідовністю графем, фонетична форма - послідовністю фонем (або фонем). Розвиток фонемної лексики має важливе значення в системах перетворення тексту в мову (TTS) та автоматичного розпізнавання мови (ASR). Для цього використовуються методи G2P, і від точності G2P-перетворення залежить отримання сучасних характеристик цих систем. Так, наприклад, в акустичних моделях АСР критично важливими компонентами є вимовні лексикони та мовні моделі. Акустичні та мовні моделі будуються автоматично з великих корпусів. Лексикони вимови є середнім шаром між акустичною та мовною моделями. Для нової задачі розпізнавання мови продуктивність всієї системи залежить від якості вимовної складової. Іншими словами, продуктивність системи залежить від точності G2P. Наприклад, G2P перетворення слова "диктор" - це "S P IY K ER". У системах TTS якісна модель G2P також є невід'ємною частиною і має великий вплив на загальну якість. Неточне перетворення G2P призводить до неприродної вимови або навіть до незрозумілої синтетичної мови.

## 2. МАТЕМАТИЧНА МОДЕЛЬ ЗАДАЧІ ГРАФЕМНО-ФОНЕМНОГО ПЕРЕТВОРЕННЯ ТЕКСТІВ

Нехай  $G = \{a, b, v, \dots, y\}$  - множина графем української мови,  $P = \{A0, \dots, Y1\}$  - множина фонем української мови,  $W$  - множина слів української мови. Також:

$$\forall w \in W \exists! \underline{g} = (g_1, g_2, \dots, g_k), g_{1..k} \in G$$

Задача графемно-фонемного відображення полягає в тому щоб знайти таке відображення  $f: G_n \rightarrow P_m$  щоб:

$$\forall g \in G, p \in P, f((g_1, g_2, \dots, g_n)) = (p_1, p_2, \dots, p_m)$$

## 3. ТИПИ МОДЕЛЕЙ ДЛЯ ВИРІШЕННЯ ЗАДАЧ

**На основі правил:**

Даний підхід базується на написанні правил (регулярних виразів) для перетворення слів у набір фонем, в якому мається на увазі розміщення букв та відповідний.

### Моделі спільних послідовностей (JSM):

Фундаментальна ідея JSM полягає в тому, що послідовності графем і фонем можуть бути згенеровані спільно за допомогою послідовності спільних одиниць (графонів), які несуть як символи графем, так і символи фонем. Завданням СПМ є знаходження послідовності  $Y$  фонем, що задана послідовністю  $X$  графем. Цю задачу можна сформулювати як визначення оптимальної послідовності фонем  $Q$ , що максимізує їх умовну ймовірність  $Q$ , при заданій послідовності графем  $G$ :

$$\hat{Q} = \arg \max_Q P(Q|G).$$

Обчислення для всіх можливих послідовностей  $Q$  безпосередньо з  $P(Q|G)$  є складним і ми можемо виразити його за допомогою правила Байєса наступним чином:

$$\hat{Q} = \arg \max_Q P(Q|G) = \arg \max_Q \{P(G|Q) \cdot P(Q)/P(G)\}$$

Тут  $P(G)$  є спільною для всіх послідовностей  $Q$ . Наведене вище рівняння можна спростити наступним чином:

$$\hat{Q} = \arg \max_Q P(G|Q) \cdot P(Q)$$

### Використання нейромереж

Методи seq2seq засновані на генерації з непрямо обумовленою мовною моделлю, останнім часом показали багатообіцяючі результати в ряді завдань. У машинному перекладі моделі, обумовлені вихідними словами, використовувалися для створення тексту цільовою мовою, а в підписі до зображень - моделі, обумовлені зображенням, для генерації тексту підписів. Минулі роботи з цим підходом були зосереджені на завданнях з великими словниками та вимірюваною якістю в термінах BLEU.

Так як графемно-фонемне перетворення по своїй суті є перетворення однієї послідовності в іншу, то для вирішення даної задачі можна використати seq2seq нейромережеві моделі (Рис. 1).

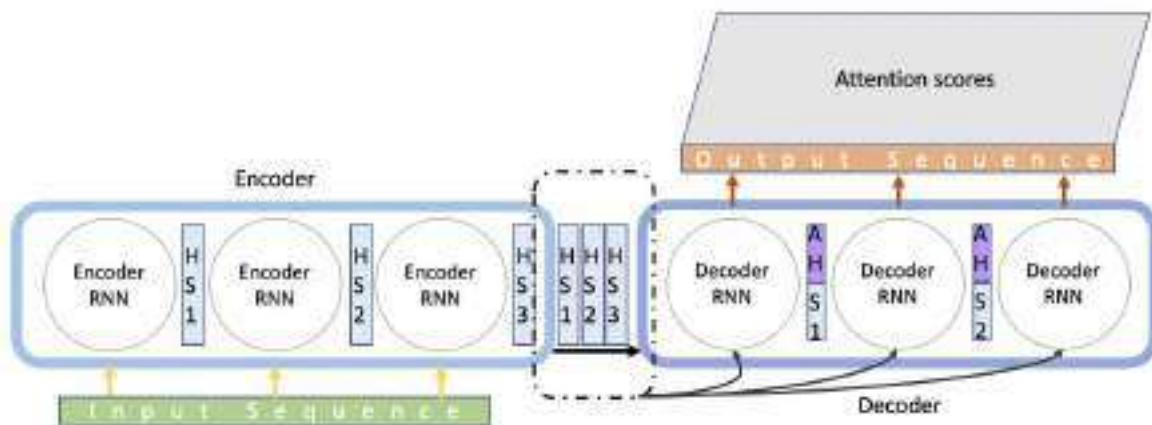


Рисунок 1. Використання seq2seq нейромережевої моделі

## 4. ВИСНОВКИ

Моделі перетворення графем у фонем (G2P) є важливими для обробки природної мови (NLP), автоматичного розпізнавання мови (ASR) та перетворення тексту в мову (TTS).

Хоча багато підходів машинного навчання застосовні для перетворення G2P, більшість з них є підходами керованого навчання, і в якості передумови ми повинні підготувати чисті анотовані навчальні дані, а це дорого коштує. На практиці нам потрібно здійснювати бутстрапінг або активне навчання за допомогою невеликого анотованого вручну словника G2P для ефективної розробки G2P-перетворювачів.

### **ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Comparison of Grapheme-to-Phoneme Conversion Methods on a Myanmar Pronunciation Dictionary URL: <https://aclanthology.org/W16-3702.pdf>
2. Sequence-to-Sequence Neural Net Models for Grapheme-to-Phoneme Conversion URL: <https://arxiv.org/pdf/1506.00196.pdf>
3. A Finite State and Data-Oriented Method for Grapheme to Phoneme Conversion URL: <https://aclanthology.org/A00-2040.pdf>