Міністерство освіти і науки України
Донецький національний університет імені Василя Стуса, м. Вінниця
Київський національний університет будівництва і архітектури
Черкаський державний технологічний університет
Київський національний торговельно-економічний університет
Львівський національний університет імені Івана Франка
Тернопільський національний технічний університет імені Івана Пулюя
Громадська організація «Освітня фундація продуктового ІТ»

Матеріали

VI ВСЕУКРАЇНСЬКОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ «КОМП'ЮТЕРНІ ТЕХНОЛОГІЇ ОБРОБКИ ДАНИХ»

5 грудня 2025 року

Матеріали надруковані в авторській редакції.
Достовірність поданої інформації лежить на авторах публікацій

Вінниця
2025

УДК 004.9:[001.8:378.091.5]’06(043.2)

Рекомендовано до друку вченою радою факультету інформаційних і прикладних технологій Донецького національного університету імені Василя Стуса (протокол № 11 від 21.04.2026)

Організаційний комітет конференції:

Голова редакційної колегії:

СЕГЕДА Сергій Андрійович, доктор економічних наук, професор, в. о. декана факультету інформаційних і прикладних технологій ДонНУ імені Василя Стуса.

Заступник голови редакційної колегії:

ВЕСЕЛОВСЬКА Наталія Ростиславівна, доктор технічних наук, професор, в. о. завідувача кафедри інформаційних технологій ДонНУ імені Василя Стуса.

Відповідальний секретар оргкомітету:

БАБАКОВ Роман Маркович, доктор технічних наук, доцент, професор кафедри інформаційних технологій ДонНУ імені Василя Стуса.

Технічний секретар оргкомітету:

Москвін Олексій Михайлович, канд. техн. наук, доцент кафедри інформаційних технологій ДонНУ імені Василя Стуса;

Члени комітету:

Ніколюк П. К., д-р фіз.-мат. наук, професор, професор кафедри інформаційних технологій ДонНУ імені Василя Стуса;

Ротштейн О. П., д-р техн. наук, професор, професор кафедри інформаційних технологій ДонНУ імені Василя Стуса;

Штовба С. Д., д-р техн. наук, професор, професор кафедри інформаційних технологій ДонНУ імені Василя Стуса;

Січко Т. В., канд. техн. наук, доцент, доцент кафедри інформаційних технологій ДонНУ імені Василя Стуса;

Антонов Ю. С., канд. фіз.-мат. наук, доцент, доцент кафедри інформаційних технологій ДонНУ імені Василя Стуса;

Потапова Н. А., канд. екон. наук, доцент, доцент кафедри інформаційних технологій ДонНУ імені Василя Стуса;

Фриз І. В., канд. фіз.-мат. наук, старший викладач кафедри інформаційних технологій ДонНУ імені Василя Стуса;

Поремський Ю. В., канд. техн. наук, старший викладач кафедри інформаційних технологій ДонНУ імені Василя Стуса;

Павлюк О. А., старший викладач кафедри інформаційних технологій ДонНУ імені Василя Стуса.

Комп’ютерні технології обробки даних: матеріали VI Всеукраїнської науково-практичної конференції (м. Вінниця, 05 грудня 2025 р.). Вінниця: ДонНУ імені Василя Стуса, 2025. 124 с.

Збірник містить матеріали VI Всеукраїнської науково-практичної конференції «Комп’ютерні технології обробки даних». Тематика конференції охоплює актуальні питання досліджень у сфері інформаційних технологій, зокрема методів обробки та аналізу даних, штучного інтелекту, інтелектуальних систем та мереж, баз даних і знань, прикладних аспектів комп’ютерних технологій, кібербезпеки, Інтернету речей та математичних методів моделювання. Матеріали учасників конференції адресовано фахівцям та усім, хто цікавиться сучасним станом вивчення прикладних аспектів сучасних міждисциплінарних досліджень.

ISSN (print): 2708-5821 **УДК 004.9:[001.8:378.091.5]’06(043.2)**

© Колектив авторів, 2025

© Донецький національний університет імені Василя Стуса, 2025

ЗМІСТ

СЕКЦІЯ 1. МЕТОДИ ОБРОБКИ ТА АНАЛІЗУ ДАНИХ

Вовк В. С., Павлюк О. А.
АЛГОРИТМ ШИФРУВАННЯ ІНФОРМАЦІЇ З ВИКОРИСТАННЯМ ПСЕВДОВИПАДКОВИХ ПОСЛІДОВНОСТЕЙ 8

Бондар А. Б., Потапова Н. А.
ВИКОРИСТАННЯ АЛГОРИТМІВ ПОШУКУ У СТВОРЕННІ СТРАТЕГІЧНИХ ІГОР РЕАЛЬНОГО ЧАСУ 9

Труханська В. О., Ніколюк П. К.
ДОСЛІДЖЕННЯ ВПЛИВУ АНСАМБЛЕВИХ АЛГОРИТМІВ НА ПРОДУКТИВНІСТЬ КЛАСИФІКАЦІЙНИХ МОДЕЛЕЙ 11

Яценко В. В., Ніколюк П. К.
ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ АЛГОРИТМІВ ДЕТЕКЦІЇ ОБ'ЄКТІВ РЕАЛЬНОГО ЧАСУ 13

Вовк В. С., Потапова Н. А.
ДОСЛІДЖЕННЯ КРИПТОГРАФІЧНИХ ВЛАСТИВОСТЕЙ ХЕШ-ФУНКЦІЇ КЕССАК 16

Головня С. О., Павлюк О. А.
МАШИНА ТЮРІНГА ЯК МОДЕЛЬ ОБЧИСЛЕНЬ: ТЕОРЕТИЧНІ ОСНОВИ ТА СУЧАСНІ ІНТЕРПРЕТАЦІЇ 17

Поліщук О. С., Ніколюк П. К.
МЕТОД МОНТЕ-КАРЛО: УНІВЕРСАЛЬНИЙ ІНСТРУМЕНТ КОМП'ЮТЕРНОГО МОДЕЛЮВАННЯ 19

Житник Є. Т., Фриз І. В.
ОПТИМІЗАЦІЯ АЛГОРИТМІВ: МЕТОДИ, ПІДХОДИ ТА ЇХ ПОРІВНЯННЯ 22

Горчук К. В., Потапова Н. А.
ОСОБЛИВОСТІ ТА ПЕРЕВАГИ ВИКОРИСТАННЯ АЛГОРИТМУ БІНАРНОГО ПОШУКУ 24

Коваль В. С., Фриз І. В.
РОЛЬ АЛГОРИТМІВ ТА СТРУКТУР ДАНИХ У СУЧАСНИХ ІТ-СИСТЕМАХ 26

Головня С. О., Потапова Н. А.
СТРАТЕГІЯ «РОЗДІЛЯЙ І ВОЛОДАРЮЙ»: КОНЦЕПТУАЛЬНА ОСНОВА ШВИДКИХ АЛГОРИТМІВ 27

СЕКЦІЯ 2. ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ ТА МЕРЕЖІ

Засипка Є. В., Павлюк О. А.
АЛГОРИТМ ОБРОБКИ ДАНИХ У ШТУЧНОМУ ІНТЕЛЕКТІ 30

Олексієнко Є. О., Павлюк О. А.
АЛГОРИТМИ ВИКОРИСТАННЯ ШТУЧНОГО ІНТЕЛЕКТУ У ЧАТ-БОТАХ 32

Шендрик В. О., Потапова Н. А.
АЛГОРИТМІЧНІ СТРУКТУРИ ТА МЕТОДИ ОБРОБКИ ДАНИХ
У ПОБУДОВІ ІНТЕЛЕКТУАЛЬНИХ АГЕНТІВ 33

Грищишен О. Ю., Січко Т. В.
АНАЛІЗ ЕФЕКТИВНОСТІ ВИКОРИСТАННЯ ЗГОРТКОВИХ
НЕЙРОННИХ МЕРЕЖ ДЛЯ ЗАДАЧ КЛАСИФІКАЦІЇ ТВАРИН 35

Заболотна У. К.
ГЕНЕТИЧНІ АЛГОРИТМИ ТА ЇХ ЗАСТОСУВАННЯ 36

Носалюк О. І., Потапова Н. А.
ЗАСТОСУВАННЯ АЛГОРИТМІВ ШТУЧНОГО ІНТЕЛЕКТУ
У РОЗРОБЦІ СУЧАСНИХ ВІДЕОІГОР 39

Бєглова А. В., Веселовська Н. Р.
ЗАСТОСУВАННЯ ГТУЧНОГО ІНТЕЛЕКТУ ДЛЯ АВТОМАТИЧНОГО
ГЕНЕРУВАННЯ UI-ТЕСТІВ ВЕБДОДАТКІВ 40

Зозуля О. Р., Січко Т. В.
ІГРОВИЙ ШТУЧННИЙ ІНТЕЛЕКТ: МОДЕЛЮВАННЯ
СТРАТЕГІЧНИХ ВЗАЄМОДІЙ 43

Григоренко Я. С., Дмитрієв Д. О.
ІНТЕЛЕКТУАЛЬНІ ТЕХНОЛОГІЇ ЯКОСТІ МАТЕРІАЛІВ
В АДИТИВНОМУ ВИРОБНИЦТВІ МЕТАЛЕВИХ ВИРОБІВ 45

Зелінський О. О, Січко Т. В.
ІНТЕРПРЕТАЦІЇ РЕЗУЛЬТАТІВ ШТУЧНОГО ІНТЕЛЕКТУ
ПІД ЧАС ІДЕНТИФІКАЦІЇ ЗОБРАЖЕНЬ 48

Грибаніна А. О., Ніколюк П. К.
МОДЕЛІ НЕЙРОННИХ МЕРЕЖ 51

Русавський О. О., Ротштейн О. П.
ПРОЄКТУВАННЯ НЕЧІТКИХ КОГНІТИВНИХ КАРТ МЕТОДОМ
НАЙМЕНШОГО ВПЛИВУ 54

Бевзюк А. Ю., Ніколюк П. К.
СТАБІЛІЗАЦІЯ ВІДЕОРЯДУ, ОТРИМАНОГО
З ВІДЕОКАМЕРИ, ВСТАНОВЛЕНОЇ НА ДРОНІ 58

**СЕКЦІЯ 3. ЕКСПЕРТНІ, РЕКОМЕНДАЦІЙНІ СИСТЕМИ
ТА СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ**

Левченко М. Р., Веселовська Н. Р.
АДАПТАЦІЯ AGILE-МЕТОДОЛОГІЙ ДЛЯ РОЗРОБКИ БЕЗПЕЧНИХ
ПРОГРАМНИХ ПРОДУКТІВ 62

Афанасьєва Д. С., Веселовська Н. Р.
АЛГОРИТМІЧНІ МЕТОДИ АДАПТАЦІЇ ВЕБКОНТЕНТУ
ДЛЯ ЛЮДЕЙ З ВАДАМИ ЗОРУ 63

Дикий Д. С., Потапова Н. А.
ВЕБПОРТАЛ УПРАВЛІННЯ ЛОГІСТИКОЮ МАГАЗИНУ
КОМПЛЕКТУЮЧИХ СМАРТСИСТЕМ 66

Явгусішин Б. А., Антонов Ю. С.
ІНТЕЛЕКТУАЛЬНА РЕКОМЕНДАЦІЙНА СИСТЕМА ДІАГНОСТИКИ ПРОГРАМНИХ НЕСПРАВНОСТЕЙ ПЕРСОНАЛЬНИХ КОМП'ЮТЕРІВ: АРХІТЕКТУРА, АЛГОРИТМИ ТА РЕАЛІЗАЦІЯ 70

Рижков О. О., Потапова Н. А.
ІНТЕЛЕКТУАЛЬНА СИСТЕМА ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ ОРЕНДИ НЕРУХОМОСТІ 72

Олійник К. А., Ніколюк П. К.
ІНФОРМАЦІЙНА СИСТЕМА ДЛЯ АНАЛІЗУ ТА ПРОГНОЗУВАННЯ РОЗВИТКУ СТРУКТУРИ УНІВЕРСИТЕТІВ УКРАЇНИ 74

Плець В. В., Січко Т. В.
МЕТОДИ МАТЕМАТИЧНОГО ПРОГРАМУВАННЯ У ВІЙСЬКОВІЙ СПРАВІ 77

Шендрик В. О., Ніколюк П. К.
РОЗРОБКА ПРОСТОГО ЧАТ-БОТА НА ОСНОВІ МЕТОДІВ ОБРОБКИ ПРИРОДНОЇ МОВИ (NLP) 80

Губчакевич В. Р., Січко Т. В.
ПЕРЕВАГИ ВИКОРИСТАННЯ BI-ТЕХНОЛОГІЙ У РОБОТІ HR-МЕНЕДЖЕРА 83

СЕКЦІЯ 4. БАЗИ ДАНИХ І ЗНАНЬ. BIG DATA

Лухверчик С. А., Потапова Н. А.
АНАЛІТИЧНІ МЕТОДИ ОБРОБКИ ВЕЛИКИХ ДАНИХ У ЛОГІСТИЧНИХ СИСТЕМАХ 87

Трохимчук О. М., Січко Т. В.
БАЗИ ДАНИХ ТА ШТУЧНИЙ ІНТЕЛЕКТ 89

Байраківська В. В., Січко Т. В.
ВПРОВАДЖЕННЯ ПРОЦЕДУР ОЧИЩЕННЯ ДАНИХ (DATA CLEANSING) ДЛЯ ПІДВИЩЕННЯ ДОСТОВІРНОСТІ ІНФОРМАЦІЇ В ІНФОРМАЦІЙНОМУ ПРОСТОРІ 90

Алексюк В. В., Потапова Н. А.
ЗАСТОСУВАННЯ БАЗ ДАНИХ NOSQL У ВЕБЗАСТОСУНКАХ ДЛЯ АНАЛІТИКИ ПРОДАЖІВ 93

Федорович І. А., Осухівська Г. М.
ЗАСТОСУВАННЯ SPARK-RAPIDS ДЛЯ ЗДІЙСНЕННЯ GPU-ОБЧИСЛЕНЬ НА РУШІЇ APACHE SPARK 95

Костенко Р. О., Веселовська Н. Р.
НАУКОВА КОМУНІКАЦІЯ В УМОВАХ ЦИФРОВІЗАЦІЇ СУСПІЛЬСТВА.....97

СЕКЦІЯ 5. ПРИКЛАДНІ АСПЕКТИ ОБРОБКИ ДАНИХ В ІНФОРМАЦІЙНИХ СИСТЕМАХ

Оврамець І. В., Антонов Ю. С.
АВТОМАТИЗОВАНА СИСТЕМА ГЕНЕРАЦІЇ ВІДПОВІДЕЙ НА ПОВІДОМЛЕННЯ З ВИКОРИСТАННЯМ DOCKER КОНТЕЙНЕРІВ... 100

Кудирка М. Ю., Павлюк О. А.
АЛГОРИТМИ ВИЗНАЧЕННЯ МАКСИМАЛЬНОЇ ПРОПУСКНОЇ ЗДАТНОСТІ В ПОТОКОВИХ МЕРЕЖАХ 102

Клименко Ю. Ю., Потапова Н. А.
АЛГОРИТМІЧНІ АСПЕКТИ РЕАЛІЗАЦІЇ СИСТЕМ АНТИПЛАГІАТУ МЕТОДАМИ SHINGLING TA RABIN-KARP 104

Вершигора Р. О., Потапова Н. А.
ОПТИМІЗАЦІЯ ОБРОБКИ ДАНИХ ЗА ДОПОМОГОЮ ХЕШ-СТРУКТУР У C# 106

Хуторна Д. В., Фриз І. В.
ОСОБЛИВОСТІ ВИКОРИСТАННЯ ХЕШ-ТАБЛИЦЬ ТА ЗБАЛАНСОВАНИХ ДЕРЕВ ПОШУКУ В АЛГОРИТМІЗАЦІЇ 108

Родюк А. І., Комаров В. Ф.
ПОРІВНЯЛЬНИЙ АНАЛІЗ МЕТОДІВ ПОБУДОВИ ФРАКТАЛЬНИХ СТРУКТУР 109

Костюшин М. В., Ніколюк П. К.
ПОРІВНЯННЯ ПІДХОДІВ ДО ЗЛИТТЯ ДАНИХ У СИСТЕМАХ МУЛЬТИМОДАЛЬНОГО РОЗПІЗНАВАННЯ ОБ'ЄКТІВ 112

Замотайло В. В., Фриз І. В.
ПРОГРАМНА РЕАЛІЗАЦІЯ АПРОКСИМАЦІЇ ПЕРІОДИЧНИХ СИГНАЛІВ РЯДАМИ ФУР'Є В СИСТЕМАХ ОБРОБКИ ДАНИХ 115

Підруцький Д. А., Потапова Н. А.
ПРОЄКТ E-COMMERCE ВЕБЗАСТОСУНКУ З ВИКОРИСТАННЯМ REACT TA INTEGRATION API 117

Пацула А. О., Потапова Н. А.
РЕКУРСІЯ В ПРОГРАМУВАННІ: ПОНЯТТЯ, ПРИНЦИП РОБОТИ ТА ЗАСТОСУВАННЯ 119

Бородій І. І., Осухівська Г. М.
ФОРМУВАННЯ ВХІДНИХ ДАНИХ ДЛЯ АНАЛІТИЧНИХ СИСТЕМ НА ОСНОВІ АРХІТЕКТУР DATA LAKEHOUSE 121

СЕКЦІЯ 1
МЕТОДИ ОБРОБКИ ТА АНАЛІЗУ ДАНИХ

УДК 004.422.5

Вовк В. С., здобувачка 2 курсу
спеціальності 122 Комп'ютерні науки,
Павлюк О. А., канд. техн. наук,
старший викладач кафедри інформаційних технологій

АЛГОРИТМ ШИФРУВАННЯ ІНФОРМАЦІЇ З ВИКОРИСТАННЯМ ПСЕВДОВИПАДКОВИХ ПОСЛІДОВНОСТЕЙ

Донецький національний університет імені Василя Стуса, м. Вінниця

Алгоритм шифрування – це набір інструкцій, що перетворює дані у формат, який неможливо прочитати без спеціального ключа (зашифрований текст), використовуючи ключ шифрування для захисту інформації.

Генератор псевдовипадкових послідовностей у класичному вигляді являє собою деякий повністю математичний алгоритм, що на виході видає числа або двійкові послідовності, які схожі на дійсно випадкові числа та послідовності. Основними елементами у таких генераторах є пам'ять розміром n, яка зберігає стан генератора, і деяке перетворення, за допомогою якого обраховується наступний стан генератора та послідовність, яка подається на вихід. Такий алгоритм завжди може бути представлений як еквівалентний автомат [1].

Генератори псевдовипадкових послідовностей з хорошими статистичними властивостями застосовуються для вирішення багатьох прикладних завдань, як-от генерація криптографічних ключів, реалізація протоколів аутентифікації, створення імітаційних моделей тощо. Для формування якісних послідовностей потрібні хороші статистичні властивості, непередбачуваність, відтворюваність.

У загальному випадку схема генерування ПВП описується виразом [2]:

$$x_{n+1} = (a \cdot x_n + d) \mathrm{mod} N,$$

де x_n, x_{n+1} – значення системи на n-ій та $n+1$-ій ітерації;

$x_0, a, d \in \{0,1, \dots, \mathrm{N}-1\}$ – параметри системи;

N – натуральне число, а «mod» означає арифметичний оператор знаходження залишку від результату ділення цілих чисел.

Алгоритм шифрування базується на використанні в якості ключа генератора ПВП. Схема генерування ключа використовує два незалежні генератори ПВП (лінійний конгурентний генератор та генератор на основі логістичного відображення), що працюють із різними початковими умовами:

$$x_{n+1} = (a \cdot x_n + d) \mathrm{mod} N, x_{n+1} = \lambda \cdot x_n (1 - x_n),$$

де a_1, d_1, λ, x_0 – початкові умови для генерування послідовностей.

Питання безпеки передачі даних з давніх-давен стояло перед суспільством. Злам, перехоплення та зміна даних наносять значну шкоду для інформаційної безпеки та людської життєдіяльності. Ці загрози можуть стосуватись як окремо людини, так і суспільства загалом і призводити до високих економічних втрат та матеріальних збитків.

Для перешкоджання виникненню таких загроз застосовують алгоритми шифрування даних. Одним із таких алгоритмів є алгоритм шифрування RSA. Це криптографічний алгоритм з відкритим ключем, який базується на складності задачі розкладу на прості множники великих цілих чисел [3].

Список використаних джерел

1. Лисенко Д. В. Метод підвищення ефективності генерації псевдовипадкових послідовностей для потокового шифрування даних: магістерська дисертація. Київ, 2022.

2. Knuth D. E. The art of computer programming. Vol. 2, Seminumerical algorithms. 3rd ed. Boston: Addison-Wesley, 1997. 762 p. URL: https://www-cs-faculty.stanford.edu/~knuth/taocp.html

3. Актуальні питання забезпечення цивільного захисту: збірник тез доповідей VIII Всеукраїнської науково-практичної конференції здобувачів вищої освіти і молодих учених (м. Львів, 2021 р.) / Львівський державний університет безпеки життєдіяльності. Львів: ЛДУ БЖД, 2021.

УДК 004.422.5

Бондар А. Б., здобувачка 2 курсу
спеціальності 122 Комп'ютерні науки,
Потапова Н. А., канд. екон. наук,
доцент, доцент кафедри інформаційних технологій

ВИКОРИСТАННЯ АЛГОРИТМІВ ПОШУКУ У СТВОРЕННІ СТРАТЕГІЧНИХ ІГОР РЕАЛЬНОГО ЧАСУ

Донецький національний університет імені Василя Стуса, м. Вінниця

Стратегічні ігри реального часу (RTS) є одним із найбільш вимогливих жанрів щодо алгоритмічних рішень, оскільки у них необхідно забезпечити одночасну взаємодію великої кількості автономних юнітів. Згідно з матеріалами Game Developer, ефективність RTS напряму залежить від здатності ігрового рушія швидко виконувати пошук маршруту, обробляти запити юнітів та відповідати на динамічні зміни середовища [1]. Саме тому алгоритми пошуку стали базовим інструментом для розробників цього жанру.

Однією з ключових задач у RTS є маршрутизація юнітів, яку найчастіше реалізують за допомогою алгоритму A*. Проте класичний A* зазнає значних труднощів на великих і насичених картах. Матеріали MovingAI показують, що для одночасного переміщення сотень юнітів розробники переходять до кооперативних методів, як-от Cooperative Pathfinding та Windowed HCA*, що дають змогу уникати конфліктів між агентами та покращують узгодженість групового руху [2].

Іншим важливим напрямом є Flow Field Pathfinding, описаний у доповіді GDC Vault. Цей підхід забезпечує не окремий маршрут для кожного юніта, а глобальне поле напрямів, яке зчитують усі агенти. Таке рішення було успішно

реалізоване у грі Supreme Commander і довело свою ефективність під час управління великими арміями, мінімізуючи обчислювальні витрати та підтримуючи стабільний FPS навіть у спрощеній симуляції бою [3].

Формування поведінки штучного інтелекту в RTS також базується на алгоритмах пошуку. За даними AI and Games, сучасні ігри використовують складні дерева рішень, поведінкові дерева та методи реактивного ШІ, що дають змогу NPC не просто слідувати заданим сценаріям, а адаптуватися до змін у грі [4]. Водночас поведінкові дерева (Behaviour Trees) залишаються одним з найкращих способів структурування логіки ШІ завдяки простоті редагування та модульності, що описано на GameAIPro [5].

Альтернативні теоретичні підходи у цьому напрямі охоплюють ієрархічний пошук, евристичні методи, моделі полів потенціалів, а також груповий рух на основі «стадної» поведінки (flocking). На практиці різні методи комбінуються залежно від того, чи потрібна точність маршруту, швидкість обчислення або узгоджений рух групи. Отже, RTS-ігри є унікальним середовищем, де одночасно застосовуються класичні алгоритми пошуку, кооперативні методи та поведінкові моделі ШІ.

Як авторську пропозицію можна виділити використання гібридних навігаційних карт, де Flow Field застосовується для визначення загального напрямку, а локальні маршрути уточнюються спрощеним A* або Jump Point Search. Така комбінація дає змогу зменшити кількість конфліктів між юнітами та підвищити швидкість обробки навігаційних запитів. З практичного погляду це може значно покращити продуктивність RTS на великих картах і забезпечити більш реалістичну поведінку юнітів під час масових битв.

Отже, алгоритми пошуку є центральним механізмом у створенні RTS-ігор, забезпечуючи не лише навігацію, а й інтелектуальну поведінку, координацію групових дій та адаптивність у динамічному середовищі. Їх постійний розвиток сприяє появі більш складних та реалістичних стратегічних систем, що визначає технологічний рівень сучасних ігор цього жанру.

Список використаних джерел

1. Pathfinding in real-time strategy games. *GameDeveloper*. URL: https://www.gamedeveloper.com/programming/pathfinding-in-real-time-strategy-games (дата звернення: 20.11.2025).

2. Cooperative pathfinding for multi-agent systems. *MovingAI*. URL: https://movingai.com/ (дата звернення: 20.11.2025).

3. Flow field pathfinding in games. *GDC Vault.* URL: https://www.gdcvault.com/play/1012386/Flow-Field-Pathfinding (дата звернення: 20.11.2025).

4. Real-time strategy game AI techniques. *AI and Games.* URL: https://aiandgames.com/rts-ai (дата звернення: 20.11.2025).

5. Behaviour trees and decision-making in games. *GameAIPro.* URL: https://www.gameaipro.com (дата звернення: 20.11.2025).

УДК 519.254:004.8:681.3

Труханська В. О., здобувачка 1 курсу
ОС «Магістр» ФЗ Комп'ютерні науки,
Ніколюк П. К., д-р фіз.-мат. наук, професор
кафедри інформаційних технологій

ДОСЛІДЖЕННЯ ВПЛИВУ АНСАМБЛЕВИХ АЛГОРИТМІВ НА ПРОДУКТИВНІСТЬ КЛАСИФІКАЦІЙНИХ МОДЕЛЕЙ

Донецький національний університет імені Василя Стуса, м. Вінниця

Методи статистичного навчання відіграють фундаментальну роль у створенні моделей для класифікації, прогнозування та виявлення закономірностей у даних. Однак окремі моделі, зокрема дерева рішень чи лінійні класифікатори, часто мають обмежену узагальнювальну здатність, особливо у випадку складних або шумних вибірок. Для подолання цих обмежень застосовують ансамблеві алгоритми, що поєднують декілька моделей у єдину систему з підвищеною надійністю та точністю [1].

Ансамблеві методи формують клас моделей, продуктивність якого перевищує можливості базових алгоритмів завдяки зменшенню розбіжностей у помилках та оптимізації процесу навчання. Підходи Bagging, Boosting та Random Forest є найпоширенішими й демонструють ефективність у задачах класифікації в різних галузях – від медицини до фінансового аналізу [1]. Їх спільна ідея полягає в комбінуванні декількох слабких або сильних класифікаторів для підвищення точності та стабільності результатів [1].

Bagging (Bootstrap Aggregating) передбачає побудову незалежних моделей на різних підвибірках даних, сформованих методом випадкової вибірки з поверненням. Такий підхід знижує дисперсію класифікатора завдяки усередненню результатів. Найчастіше в цьому методі застосовують дерева рішень, оскільки вони є високовариативними щодо змін у даних [2].

Random Forest поєднує ідею Bagging із додатковою випадковістю під час вибору ознак, що зменшує кореляцію між деревами та підвищує стійкість моделі до перенавчання. Завдяки такій декореляції Random Forest зберігає баланс між точністю, інтерпретованістю та стійкістю до шуму [2].

Boosting працює за іншим принципом: моделі будуються послідовно, і кожна наступна фокусується на помилках попередніх. Це зменшує зміщення та дає змогу виявляти складні закономірності. Популярні реалізації – AdaBoost, Gradient Boosting, XGBoost – забезпечують високу ефективність, але є більш чутливими до шуму та вимагають ретельного налаштування параметрів [2].

Для об'єктивного оцінювання продуктивності ансамблевих алгоритмів аналізують точність класифікації, здатність до узагальнення та стійкість до варіативності даних [3]. Узагальнені аналітичні результати свідчать, що Bagging підвищує стабільність і знижує дисперсію, Random Forest демонструє покращену узагальнювальну здатність завдяки випадковому відбору ознак, тоді як Boosting забезпечує найвищі показники точності, але за умови коректної підготовки даних [3].



Рисунок 1 – Порівняння продуктивності ансамблевих методів

Порівняння продуктивності трьох методів наведено на рис. 1. Воно демонструє поступове зростання ефективності від Bagging до Random Forest і далі до Boosting, що узгоджується з теоретичними особливостями цих моделей. Найвищі показники Boosting пояснюються його здатністю адаптивно зменшувати зміщення, тоді як Random Forest оптимально поєднує точність та стійкість до шуму.

Висновки. Ансамблеві алгоритми є одним із найефективніших інструментів сучасного статистичного навчання. Bagging знижує дисперсію моделей, Boosting покращує здатність до навчання на складних залежностях, а Random Forest забезпечує збалансований підхід до точності й стабільності. Аналітичні та теоретичні результати підтверджують їх ефективність у задачах класифікації, особливо за умов шумності чи високої складності даних. Подальші дослідження можуть бути спрямовані на оптимізацію гіперпараметрів і розробку гібридних ансамблевих підходів.

Список використаних джерел

1. Талах М. В. Ансамблеві архітектури та технології аналізу даних. Чернівці: Технодрук, 2023. 246 с.

2. Kotsiantis S. Combining bagging, boosting, rotation forest and random subspace methods. *Artificial Intelligence Review.* 2011. Vol. 35. P. 223–240. URL: https://doi.org/10.1007/s10462-010-9192-8 (дата звернення: 23.11.2025).

3. Austin P. Using ensemble-based methods for directly estimating causal effects: an investigation of tree-based g-computation. *Multivariate Behavioral Research.* 2012. Vol. 47, № 1. P. 115–135.

УДК 004.93:004.8

Яценко В. В., здобувач 4 курсу спеціальності
122 Комп'ютерні науки,
Ніколюк П. К., д-р фіз.-мат. наук, професор
кафедри інформаційних технологій

ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ АЛГОРИТМІВ ДЕТЕКЦІЇ ОБ'ЄКТІВ РЕАЛЬНОГО ЧАСУ

Донецький національний університет імені Василя Стуса, м. Вінниця

Комп'ютерний зір – галузь штучного інтелекту, що займається автоматизованим аналізом візуальної інформації. Детекція об'єктів – ключове завдання цієї галузі, що полягає у автоматичному виявленні та просторовій локалізації об'єктів різних категорій у цифрових зображеннях чи відеопотоках [1].

Сучасний етап розвитку технологій характеризується масовим впровадженням систем автоматичного розпізнавання у повсякденне життя. Системи відеоспостереження у містах, автоматизовані лінії виробництва, допоміжні системи водіння автомобілів – усі ці застосування потребують надійного виявлення об'єктів у візуальних даних. В Україні детекція об'єктів набуває особливої гостроти через використання БПЛА у військовій сфері. Безпілотники потребують алгоритмів, здатних працювати в режимі реального часу за обмежених ресурсів, водночас завдання розвідки вимагають максимальної точності. Ця двоякість вимог – швидкість проти точності – визначає необхідність комплексного аналізу наявних підходів до детекції об'єктів.

У статті «Deep Learning for Generic Object Detection: A Survey» представлено комплексний огляд сучасних архітектур для детекції об'єктів на основі глибинного навчання [2]. Дослідження показує, що розвиток галузі рухається у двох напрямах: надточні моделі для критичних застосувань та ультрашвидкі детектори для периферійних пристроїв. Окрему увагу приділено гібридним архітектурам, що намагаються об'єднати переваги обох підходів через каскадні структури та адаптивне об'єднання ознак.

Традиційно алгоритми детекції поділяють на двоетапні та одноетапні детектори. Двоетапні детектори базуються на послідовній обробці зображення. Спочатку генеруються регіони-кандидати, далі кожна область класифікується та уточнюється. Такий підхід забезпечує високу точність локалізації, особливо для дрібних об'єктів. Faster R-CNN (Region-based Convolutional Neural Networks, регіональні згорткові нейронні мережі) є типовим представником цього класу. RetinaNet вдосконалив цей підхід через запровадження focal loss (фокусної функції втрат), що дає змогу мережі під час навчання приділяти більше уваги складним прикладам, покращуючи точність без зниження швидкості [3].

One-stage-детектори, тобто одноетапні алгоритми, здійснюють виявлення та класифікацію одночасно за один прохід, що суттєво прискорює обробку [4].

YOLO (You Only Look Once, дивишся лише раз) розглядає все завдання детекції як пряму регресію від пікселів зображення до координат та класів об'єктів,

що дає значний приріст у швидкості розпізнавання. Сучасні версії – YOLOv8 та YOLOv11 – демонструють значно кращий баланс між точністю та швидкістю завдяки оптимізованим опорним мережам і передовим методам аугментації даних під час навчання [5]. SSD (Single Shot Detector, детектор одиничного кадру) використовує передбачення на різних рівнях мережі для виявлення об'єктів різного масштабу.

Метою дослідження виступає аналіз сучасних методів детекції об'єктів з акцентом на співвідношення швидкості та точності для визначення оптимальних сценаріїв застосування різних підходів.

Під час дослідження необхідно обрати репрезентативні моделі, провести над ними тестування на стандартному датасеті з вимірюванням швидкості обробки, точності детекції та розміру моделей, проаналізувати отримані результати та сформулювати практичні рекомендації щодо вибору методу залежно від специфіки застосування.

Для об'єктивного порівняння було обрано шість моделей. Швидкі моделі – YOLOv8n та YOLOv11n – є найлегшими версіями сучасних YOLO – архітектур з розміром 5–6 MB, SSD300 – класичний одноетапний детектор розміром 136 MB, та MobileNet R-CNN – оптимізована версія Faster R-CNN для мобільних пристроїв розміром 74 MB. Точні моделі представлені Faster R-CNN з ResNet50 backbone розміром 160 MB – еталоном орієнтованого на точність підходу, та RetinaNet розміром 130 MB – детектором з focal loss.

Тестування проводилося на датасеті COCO2017, що містить 5 000 зображень з різноманітними об'єктами та сценами. Вимірювалися такі показники: швидкість обробки у кадрах на секунду (FPS), точність детекції за метриками mAP@0.5 (mean Average Precision при IoU=0.5) та mAP@0.5:0.95 (середнє значення mAP за різних порогів IoU від 0.5 до 0.95) та розмір моделі. Тестування виконувалося на обладнанні з графічним прискорювачем NVIDIA.

Результати тестування представлено на рис. 1. Зелені стовпці позначають швидкі моделі, сині – точні моделі, орієнтовані на максимальну точність. Червоні пунктирні лінії позначають критичні пороги: 15 FPS для відеоаналізу в реальному часі та 50 MB для розгортання на периферійних пристроях.

Швидкі моделі демонструють 53–67 FPS за точності 42–44 % mAP@0.5 та розмірі 5–136 MB, перевищуючи поріг у 15 кадрів у 3.5–4.5 рази. Точні моделі забезпечують 56–59 % mAP@0.5 (на 30 % вище), але працюють лише на 9 FPS при розмірі 130–160 MB.

Графік trade-off демонструє фундаментальний компроміс: збільшення точності у 1.3 раза вимагає зниження швидкості у 6–7 разів. MobileNet R-CNN займає проміжну позицію (26.6 FPS, 52.6 % mAP@0.5), оптимальну для обладнання середнього класу. Аналіз використання VRAM показує, що YOLO-моделі споживають 600–700 MB під час інференсу, що дає змогу розгортання на БПЛА з 4+ GB RAM після застосування квантизації INT8 та TensorRT оптимізації.

На основі отриманих результатів можна сформулювати чіткі рекомендації щодо вибору методу. YOLO – моделі оптимальні для реального часу на БПЛА та мобільних пристроях. Швидкість 50–70 FPS та розмір 5–6 MB дають змогу використання на embedded-системах. Точність 42–44 % mAP@0.5 є достатньою для задач моніторингу та навігації.

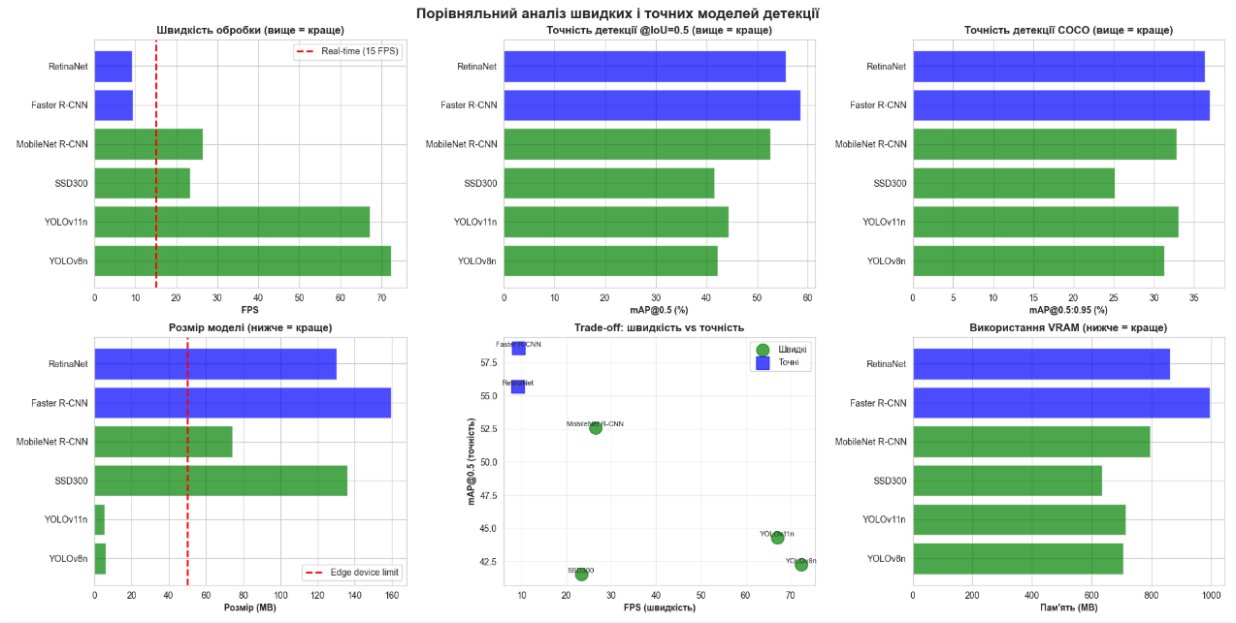


Рисунок 1 – Порівняльний аналіз продуктивності моделей детекції об'єктів

Faster R-CNN та RetinaNet доцільні для офлайн-аналізу: обробка відео, наукові дослідження, медична діагностика. Точність 56–59 % mAP@0.5 (на 30 % вище) виправдовує нижчу швидкість.

MobileNet R-CNN – проміжний варіант: 26.6 FPS, 52.6 % mAP@0.5. Підходить для відеоаналітики на обладнанні середнього класу.

Проведене дослідження підтверджує фундаментальний компроміс між швидкістю та точністю у методах детекції об'єктів. Одноетапні детектори забезпечують швидкість для реального часу, двоетапні демонструють вищу точність для критичних задач. Вибір методу визначається специфікою задачі та ресурсами.

Список використаних джерел

1. Computer Vision: Algorithms and Applications. URL: https://szeliski.org/Book/ (дата звернення: 02.12.2024).

2. Deep Learning for Generic Object Detection: A Survey. URL: https://arxiv.org/abs/1809.02165 (дата звернення: 02.12.2024).

3. Focal Loss for Dense Object Detection. URL: https://arxiv.org/abs/1708.02002 (дата звернення: 02.12.2024).

4. Object Detection in 20 Years: A Survey. URL: https://arxiv.org/abs/1905.05055 (дата звернення: 02.12.2024).

5. You Only Look Once: Unified, Real-Time Object Detection. URL: https://arxiv.org/abs/1506.02640 (дата звернення: 02.12.2024).

УДК 004.422.5

Вовк В. С., здобувачка 2 курсу спеціальності
122 Комп'ютерні науки,
Потапова Н. А., канд. екон. наук, доцент,
доцент кафедри інформаційних технологій

ДОСЛІДЖЕННЯ КРИПТОГРАФІЧНИХ ВЛАСТИВОСТЕЙ ХЕШ-ФУНКЦІЇ KECCAK

Донецький національний університет імені Василя Стуса, м. Вінниця

Keccak (вимовляється як «кетчак») – це універсальна криптографічна функція, розроблена Гвідо Бертоні, Джоан Демен, Міхаелем Пітерсом і Жилем Ван Аше. Хоча Keccak може використовуватися і для інших цілей, ця функція найбільш відома як хеш-функція, що забезпечує підвищений рівень безпеки, порівняно зі старими алгоритмами хешування, як от SHA-1 і SHA-2. Keccak-256 була розроблена як кандидат для конкурсу криптографічних хеш-функцій SHA-3, проведеного в 2007 р. Національним інститутом науки і технологій. Keccak став переможним алгоритмом, який у 2015 р. був стандартизований як Федеральний стандарт обробки інформації (FIPS) 202 [1].

Завдяки впровадженню в Keccak унікальної структури, яка продемонструвала більшу безпеку та надійність, ніж інші алгоритми, цей алгоритм був обраний NIST. Із технічного погляду, алгоритм SHA-3 заснований на функціях губки, на відміну від конструкції Меркла–Дамґарда, що використовується в SHA-1 та SHA-2.

Keccak використовується в блокчейні (наприклад, Ethereum та Solana) для забезпечення цілісності даних, а також у сферах кібербезпеки для різних криптографічних застосувань.

Версії Keccak мали $b = 1\,600$ та $c = 2n$, де $n \in \{224, 256, 384, 512\}$. Стан із 1 600 бітів можна розглядати як тривимірний масив бітів a[5][5][64], і кожен біт стану пов'язаний із 3 цілочисельними координатами, a[x][y][z], де x та y беруться за модулем 5, а z – за модулем 64 [2].

Перестановка Keccak складається з 24 раундів, які оперують 1 600 бітами стану. Кожен раунд перестановки складається з п'яти відображень $R = \iota \circ \chi \circ \pi \circ \rho \circ \theta$. Keccak використовує такі угоди про іменування, які допомагають описувати ці відображення:

– Рядок – це набір із 5 бітів із постійними координатами y та z, тобто $a[*][y][z]$.

– Стовпець – це набір із 5 бітів із постійними координатами x та z, тобто $a[x][*][z]$.

– Лінія – це набір із 64 бітів із постійними координатами x та y, тобто $a[x][y][*]$.

– Зріз – це набір із 25 бітів із постійною координатою z, тобто a[*][*][z].

Під час хешування за допомогою keccak256 будь-яка модифікація або зміна рядка призводить до значної зміни хеш-дайджесту. Заміна навіть однієї літери у вхідних даних $A > B$ призводить до значних змін хеш-дайджесту (рис. 1) [3].

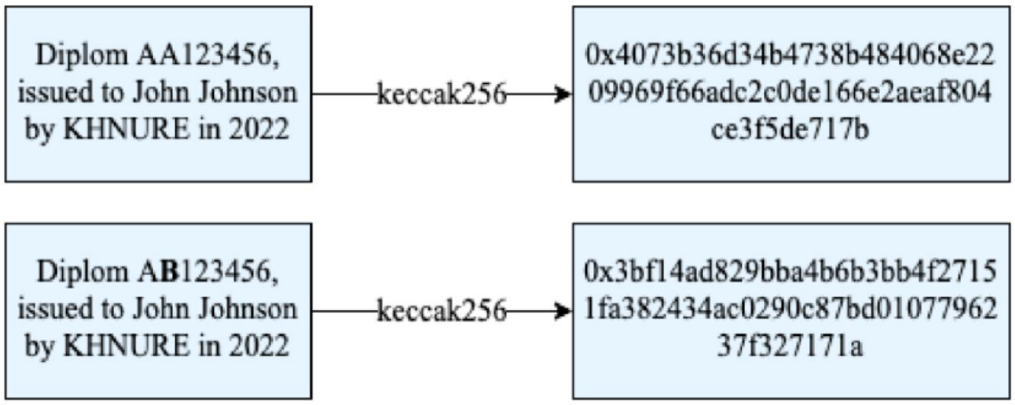


Рисунок 1 – Приклад хешування даних методом keccak256

Враховуючи безпеку та низький рівень можливості атак на SHA-3, цей алгоритм знаходить все більше застосування. Але на цьому все не закінчується. У майбутньому будуть розроблені нові та більш інноваційні хеш-алгоритми. Для криптографічної індустрії важливо постійно еволюціонувати та вирішувати численні проблеми.

Список використаних джерел

1. Antonopoulos A. M., Wood G. Mastering Ethereum: building smart contracts and DApps. Sebastopol: O'Reilly Media, 2018. 424 p.

2. Fast software encryption: 19th International Workshop, FSE 2012, Washington, DC, USA, March 19–21. 2012: revised selected papers / ed. by A. Canteaut. Berlin; Heidelberg: Springer. 2012. 476 p.

3. Group I. S. Trends in the development of science in the modern world. Primedia eLaunch LLC. 2022.

УДК 004.422.5

Головня С. О., здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,
Павлюк О. А., канд. техн. наук, старший викладач кафедри інформаційних технологій

МАШИНА ТЮРІНГА ЯК МОДЕЛЬ ОБЧИСЛЕНЬ: ТЕОРЕТИЧНІ ОСНОВИ ТА СУЧАСНІ ІНТЕРПРЕТАЦІЇ

Донецький національний університет імені Василя Стуса, м. Вінниця

Проблема формального визначення алгоритму є центральною в розвитку теорії обчислень. Щоб описати процес обчислення у строгих математичних термінах, потрібна універсальна модель, здатна відтворювати будь-які логічні та арифметичні дії. Таку модель у 1936 р. запропонував Алан Тюрінг, створивши

абстрактну машину Тюрінга. У своїй праці «On Computable Numbers…» він визначив поняття обчислюваних чисел і довів, що не існує універсального алгоритму, який може встановити істинність будь-якого математичного твердження, тобто проблема розв'язності (Entscheidungsproblem) не має загального рішення [1]. Запропонована модель стала підґрунтям тези Черча–Тюрінга, згідно з якою будь-який ефективний алгоритм може бути реалізований у межах машини Тюрінга [1], [2]. Ця ідея визначила сучасне розуміння процесу обчислень і меж можливого в алгоритмах.

Машина Тюрінга – це абстрактний автомат, що складається з нескінченної стрічки, головки читання-запису та пристрою керування зі скінченною кількістю станів. Під час роботи машина читає символ, змінює його відповідно до таблиці переходів, переміщується ліворуч або праворуч і переходить у новий стан. Такий простий механізм, попри свою абстрактність, може відтворювати будь-який обчислювальний процес [1].

Тюрінг довів існування необчислюваних задач, зокрема проблеми зупинки – неможливо створити алгоритм, який визначає, чи завершиться робота будь-якої програми для заданих вхідних даних [1]. Цей результат визначив межі обчислюваності й показав, що не всі задачі мають алгоритмічне розв'язання.

Водночас учений показав, що всі обчислювані функції можуть бути реалізовані на машині Тюрінга, а універсальна машина здатна імітувати роботу будь-якої іншої [2], [3]. Це стало основою для створення концепції універсального комп'ютера, який виконує будь-яку програму, задану послідовністю інструкцій [3].

У сучасній інформатиці машина Тюрінга залишається базовою моделлю. Вона використовується для аналізу складності обчислень, формалізації алгоритмів і класифікації задач за складністю (класи P, NP тощо) [2]. В освітній практиці ця модель допомагає студентам зрозуміти логіку побудови алгоритмів і межі їх можливостей [3].

Машина Тюрінга є фундаментом сучасної теорії обчислень. Вона формалізувала поняття алгоритму, дала змогу визначити межі розв'язності задач і стала концептуальною основою для створення універсальних комп'ютерів. Її універсальність зберігає значення як у теоретичних дослідженнях, так і в освіті, забезпечуючи розуміння принципів і меж обчислювальних процесів.

Список використаних джерел

1. Тюрінг А. М. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society.* 1937. Vol. 42. P. 230–265.

2. Новаторський М. А. Алгоритми та методи обчислень. Київ: КПІ ім. Ігоря Сікорського, 2019. 256 с.

3. Ящик О. Б. Машина Тюрінга як універсальний виконавець алгоритмів та її застосування в процесі поглибленого вивчення алгоритмізації і основ програмування. *Інформаційні технології і засоби навчання.* 2016. Т. 52, № 2. С. 10–16.

УДК 004.021

Поліщук О. С., здобувачка 4 курсу
спеціальності 122 Комп'ютерні науки,
Ніколюк П. К., д-р фіз.-мат. наук, професор
кафедри інформаційних технологій

МЕТОД МОНТЕ-КАРЛО: УНІВЕРСАЛЬНИЙ ІНСТРУМЕНТ КОМП'ЮТЕРНОГО МОДЕЛЮВАННЯ

Донецький національний університет імені Василя Стуса, м. Вінниця

Метод Монте-Карло (ММК) – це клас обчислювальних алгоритмів, які покладаються на багаторазовий випадковий відбір (семплінг) для отримання числових результатів. Це метод розв'язання математичних задач за допомогою моделювання випадкових величин.

Сучасна версія методу була розроблена в кінці 1940-х рр. у Лос-Аламоській національній лабораторії під час роботи над ядерною зброєю (Мангеттенський проєкт). Назву запропонував Ніколас Метрополіс на честь казино в Монте-Карло (Монако), де дядько Улама часто грав в азартні ігри, оскільки метод схожий на статистичну природу ігор (рулетка, кістки) [1].

Математичним фундаментом методу Монте-Карло є закон великих чисел, який виступає головним гарантом достовірності результату моделювання. Цей закон стверджує, що за достатньо великої кількості незалежних випробувань середнє арифметичне отриманих випадкових значень неминуче наближається до істинного математичного сподівання (шуканого розв'язку). Тобто чим більше експериментів ми проводимо, тим меншим стає відхилення нашого емпіричного результату від реальної фізичної або математичної величини.

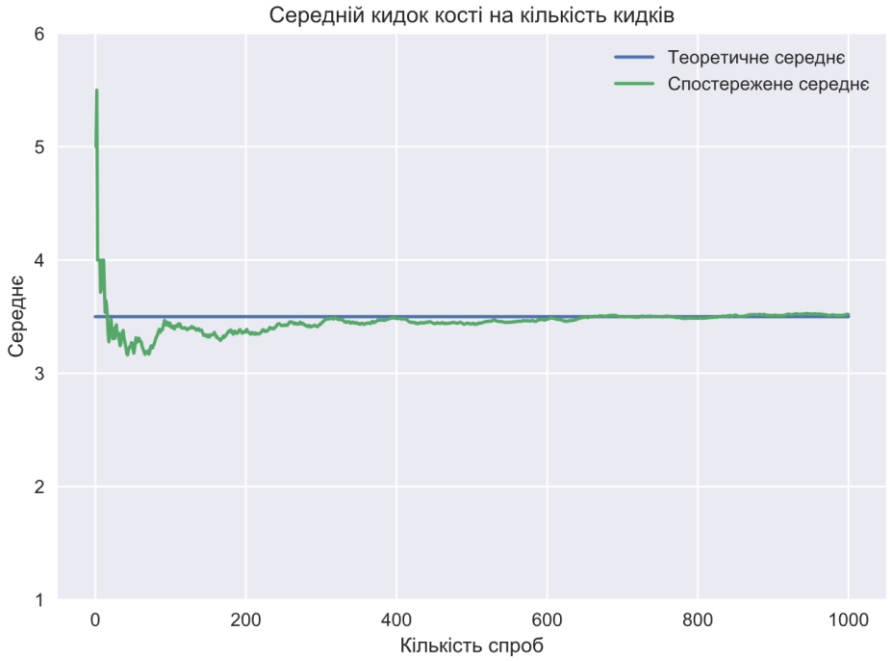


Рисунок 1 – Графічна ілюстрація Закону великих чисел

Ілюстрація закону великих чисел із використанням певного перебігу кидань кості. З тим, як кількість кидків збільшується, середнє значення всіх кидків наближується до 3.5. Різні перебіги показуватимуть різні форми кривої у частині, що відображає малу кількість кидків (ліва частина); у частині, що відображає велику кількість кидків (права частина) вони будуть дуже подібні [4].

Для глибшого аналізу точності використовується Центральна гранична теорема Тюрінга, яка дає змогу не лише отримати числову відповідь, а й оцінити її надійність шляхом побудови довірчих інтервалів. Згідно з цією теоремою, сума великої кількості незалежних випадкових величин має розподіл, близький до нормального (гаусівського). Це дає можливість досліднику чітко розуміти, з якою ймовірністю отриманий результат потрапляє в допустимі межі похибки.

Ключовою особливістю методу є специфіка оцінки похибки, яка пропорційна величині $1/\sqrt{N}$, де N – кількість проведених випробувань. Це вказує на відносно повільну збіжність алгоритму: щоб зменшити похибку в 10 разів, обсяг обчислень необхідно збільшити у 100 разів. Однак, попри таку «вартість» точності, метод Монте-Карло має вирішальну перевагу над класичними детермінованими методами: швидкість його збіжності фактично не залежить від розмірності простору, що робить його незамінним інструментом для розв’язання складних багатовимірних задач [2].

Практична реалізація методу Монте-Карло в комп’ютерному моделюванні, незалежно від специфіки задачі, завжди спирається на чітку алгоритмічну структуру. Процес починається з визначення області можливих вхідних даних. На цьому етапі дослідник має чітко окреслити межі простору, в якому відбуватиметься моделювання. Це схоже на встановлення «ігрового поля»: ми повинні знати, які параметри системи можуть змінюватися і в яких саме діапазонах, щоб уникнути генерації фізично неможливих або некоректних станів системи.

Серцем алгоритму є етап генерації випадкових чисел. Оскільки комп’ютери є детермінованими машинами, вони не можуть створити «справжню» випадковість, тому використовуються генератори псевдовипадкових чисел (PRNG).

Сучасним стандартом у цій сфері є алгоритм «Вихор Мерсенна», який забезпечує величезний період повторюваності та високу статистичну якість чисел. Важливо не просто генерувати числа, а й забезпечити їх відповідність заданому розподілу ймовірностей (наприклад, рівномірному, нормальному чи експоненціальному), що диктується фізичною природою модельованого процесу [3].

Після отримання вхідних даних настає фаза детермінованих обчислень та агрегації результатів. Для кожного згенерованого набору випадкових параметрів система виконує розрахунок за фіксованою математичною моделлю або формулою – це називається «прогоном» або «реалізацією». Коли накопичується достатня кількість таких реалізацій, отримані дані об’єднуються. Фінальна відповідь отримується шляхом статистичної обробки: результати усіх прогонів усереднюються, що дає змогу нівелювати вплив окремих випадкових відхилень і отримати стабільне, наближене до істини значення.

Щодо аналітичної оцінки ефективності, то для обґрунтування доцільності використання методу варто порівняти швидкість збіжності його похибки (ϵ) з класичними детермінованими методами (наприклад, методом Сімпсона) залежно від кількості випробувань (N) та розмірності задачі (d):

– Детерміновані методи: похибка залежить від розмірності як $\approx N^{-1/d}$. Це означає, що зі збільшенням кількості змінних (d) ефективність катастрофічно падає (так зване «прокляття розмірності»).

– Метод Монте-Карло: похибка завжди пропорційна $\epsilon \approx N^{-1/2}$ і не залежить від розмірності простору d.

Хоча для одновимірних задач ($d = 1$) класичні методи працюють швидше, для багатовимірних задач (зазвичай при $d \geq 5$) метод Монте-Карло стає безальтернативним, оскільки його точність залишається стабільною незалежно від складності системи.

Переваги:

– Можна застосовувати до задач, які не мають аналітичного розв'язку.

– Ефективність методу майже не падає зі збільшенням кількості змінних (вирішує проблему «прокляття розмірності»).

– Кожне випробування є незалежним, тому метод ідеально підходить для суперкомп'ютерів та кластерних обчислень.

Недоліки:

– Для досягнення високої точності потрібні значні ресурси.

– Отримання «точного» результату може зайняти багато часу, порівняно з детермінованими методами в задачах низької розмірності.

– Використання поганого генератора псевдовипадкових чисел може призвести до системних помилок у моделюванні.

У підсумку, метод Монте-Карло залишається одним із найпотужніших інструментів у сучасному комп'ютерному моделюванні. Зі зростанням потужності комп'ютерів сфера його застосування лише розширюється, переходячи від класичних фізичних задач до складних систем штучного інтелекту та фінансового прогнозування.

Список використаних джерел

1. Metropolis N., Ulam S. The Monte Carlo method. *Journal of the American Statistical Association.* 1949. URL: https://web.williams.edu/Mathematics/sjmiller/public_html/105Sp10/handouts/MetropolisUlam_TheMonteCarloMethod.pdf (дата звернення: 12.11.2025).

2. Rubinstein R. Y., Kroese D. P. Simulation and the Monte Carlo method. 3rd ed. Hoboken: Wiley. 2016.

3. Matsumoto M., Nishimura T. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation.* 1998. Vol. 8, № 1. P. 3–30. URL: https://dl.acm.org/doi/10.1145/272991.272995 (дата звернення: 12.11.2025).

4. Закон великих чисел. *Wikipedia, the free encyclopedia.* URL: https://uk.wikipedia.org/wiki/Закон_великих_чисел (дата звернення: 11.10.2025).

УДК 004.422.5

Житник Є. Т., здобувачка 2 курсу спеціальності
122 Комп'ютерні науки,
Фриз І. В., канд. фіз.-мат. наук,
старший викладач кафедри інформаційних технологій

ОПТИМІЗАЦІЯ АЛГОРИТМІВ: МЕТОДИ, ПІДХОДИ ТА ЇХ ПОРІВНЯННЯ

Донецький національний університет імені Василя Стуса, м. Вінниця

Оптимізація алгоритмів – це процес удосконалення алгоритмічних рішень з метою зменшення витрат ресурсів під час їх виконання. Адекватний аналіз ефективності алгоритму є критично важливим етапом розробки: правильний вибір алгоритму часто має вирішальне значення, адже навіть «теоретично оптимальний» алгоритм може виявитися на практиці надто повільним. Аналіз алгоритмів передбачає оцінку часових та пам'ятних витрат як функції розміру вхідних даних. Метою оптимізації є змінити алгоритм чи структури даних так, щоб зменшити цю складність чи покращити константи за тієї ж асимптики [1].

Визначають 3 підходи до оптимізації:

1. Зменшення асимптотичної складності. Часто це досягається застосуванням кращих алгоритмічних технік або структур даних. Наприклад, заміна квадратичного алгоритму на лінійно-логарифмічний $O(n^2) \to O(n \log n)$ дає різке підвищення продуктивності за великих n [1]. Додатково, важливою є мінімізація використання пам'яті (оптимізація пам'яті).

2. Переходи від рекурсії до ітерації. Рекурсивні алгоритми витрачають час на виклики функцій і виділення стекових кадрів. Часто можна переписати рекурсивні процедури у вигляді ітераційних циклів, що усуває накладні витрати викликів. Динамічне програмування демонструє цю ідею: рекурсивний алгоритм із мемоїзацією можна перетворити на еквівалентний bottom-up варіант. Водночас підхід «знизу вгору» працює швидше, бо відсутній накладний час рекурсії [2]. За такого переписування принцип роботи алгоритму не змінюється, але константи під час виконання коду значно зменшуються.

3. Мемоїзації і кешування результатів дає змогу уникнути повторних обчислень. Динамічне програмування – класичний приклад такого підходу: алгоритми зберігають проміжні результати в таблиці, щоб під час повторного виникнення підзадачі не виконувати обчислення заново. Динамічне програмування використовує властивість перекриття підзадач – кожну підзадачу обчислюють лише раз і потім повертають із таблиці за $O(1)$-час [2]. Так, мемоїзований (рекурсивний) та ітеративний DP-алгоритм для задачі матричного перемноження обчислюють оптимальну вартість за $O(n^3)$ однаково асимптотично, але ітераційний варіант працює швидше завдяки відсутності накладних витрат на рекурсію [2]. Порівняння методів оптимізації алгоритмів наведено у табл. 1.

Таблиця 1 – Порівняння методів оптимізації алгоритмів

Метод	Сутність	Застосування	Переваги	Недоліки
Хешування	Перетворення даних у хеш-код для швидкого доступу	Пошук, перевірка приналежності, кешування	Дуже висока швидкість ($O(1)$). Ефективне використання пам'яті за гарного хешу	Колізії. Залежність від хорошої хеш-функції
Ефективність структур даних	Підбір структури з оптимальними операціями	Сортування, пошук, маршрути, обчислення пріоритетів	Зменшення складності алгоритмів. Гнучкість	Необхідність глибокого розуміння структур. Складність реалізації
«Розділяй і володарюй»	Рекурсивний поділ задачі на дрібніші підзадачі	Сортування (MergeSort, QuickSort), пошук, обчислення	Значне прискорення обчислень. Природна паралелізація	Залежність від рекурсії. Накладні витрати на злиття та поділ
Динамічне програмування	Запам'ятовування вже обчислених результатів	Оптимізаційні задачі, графи, комбінаторика	Максимальне скорочення повторних обчислень. Мінімізує складність експоненційних задач	Вимагає багато пам'яті. Потрібно правильно визначити підзадачі
Евристики та жадібні алгоритми	Вибір часткового найкращого рішення на кожному кроці	Задачі маршрутизації, наближених рішень, великий обсяг даних	Дуже швидкі. Підходять для великих задач	Не гарантують оптимального результату. Можуть застрягати в локальних мінімумах

Оптимізація алгоритму може здійснюватися різноманітними техніками:

1. Хешування прискорює пошук у великих множинах даних: хеш-таблиця дає змогу виконувати операції вставки чи пошуку в амортизований час $O(1)$. Як зазначено у підручнику, за допомогою хешування можна побудувати структуру даних із постійним часом пошуку [1]. Використання добре спроєктованої хеш-функції та механізму розв'язання колізій забезпечує швидкий доступ до елементів.

2. Підбір ефективних структур даних – наприклад, збалансованих дерев, бінарних куп та пріоритетних черг, дає змогу оптимізувати операції вставки / видалення / пошуку. Наприклад, пріоритетна черга на основі кучи дає змогу за $O(\log n)$ дістати мінімальний елемент, тоді як із простої лінійної черги цей час був би $O(1)$ лише для вставки, але $O(n)$ для вибору мінімального.

3. «Розділяй і володарюй». Розбиття задачі на підзадачі зазвичай зменшує складність. Класичний приклад – алгоритм сортування злиттям: він розбиває масив навпіл, сортує кожну половину й потім об'єднує результати за лінійний час. Завдяки такому розбиттю та конструюванню рішення за допомогою двох часткових розв'язків, злиття забезпечує асимптотичну швидкість $O(n \log n)$. Інші приклади D&C – швидке сортування, бінарний пошук тощо – зазвичай мають кращу складність, порівняно з наївними алгоритмами.

4. Динамічне програмування та мемоїзація для задач із оптимальною підструктурою. Якщо задача дає змогу розбити її на перекриваючі підзадачі з властивістю optimal substructure, DP-алгоритм зберігає результати підзадач у таблиці,

що допомагає уникнути повторних обчислень [2]. Так складність може зменшитися з експоненційної до поліноміальної.

5. Жадібні алгоритми та евристики. Жадібні алгоритми приймають локально оптимальний вибір у кожному кроці, сподіваючись, що це приведе до глобально оптимального рішення. Евристичні підходи застосовують наближені методи, щоб швидко отримати достатньо хороший розв'язок, коли екзогенно оптимальний алгоритм є занадто повільним. Такі методи особливо важливі для задач NP-складності, де точний розв'язок може вимагати надто тривалого часу.

Отже, кожен із розглянутих методів оптимізації виконує унікальну функцію та застосовується для різних типів алгоритмічних задач і вибір конкретного методу оптимізації залежить від структури задачі, обмежень ресурсів та вимог до точності. Найвищу ефективність забезпечує комбінація декількох підходів, що дає змогу збалансувати швидкість, складність реалізації й кінцеву якість результату.

Список використаних джерел

1. Креневич А. П. Алгоритми і структури даних: підручник. Київ: ВПЦ «Київський Університет», 2021. 200 с.

2. Introduction to algorithms / T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein. 4th ed. Cambridge, MA: The MIT Press, 2022. 1312 p. URL: https://books.google.com.ua/books?id=NLngYyWFl_YC&printsec=copyright&hl=uk#v=onepage&q&f=false (дата звернення: 15.11.2025).

3. Skiena S. S. The algorithm design manual. 3rd ed. Cham: Springer, 2020. 810 p.

УДК 004.422.5

Горчук К. В., здобувач 2 курсу
спеціальності 122 Комп'ютерні науки,
Потапова Н. А., канд. екон. наук, доцент,
доцент кафедри інформаційних технологій

ОСОБЛИВОСТІ ТА ПЕРЕВАГИ ВИКОРИСТАННЯ АЛГОРИТМУ БІНАРНОГО ПОШУКУ

Донецький національний університет імені Василя Стуса, м. Вінниця

Бінарний пошук – це алгоритм пошуку індексу елементу у впорядкованому масиві. На кожній ітерації відбувається поділ масиву на дві частини, через що алгоритм називають методом ділення навпіл. Цей спосіб пошуку даних достатньо простий для розуміння, водночас він дуже ефективний, оскільки на кожному кроці кількість елементів у робочій області масиву зменшується вдвічі [1].

Алгоритм бінарного пошуку полягає у визначенні значення елементу посередині робочої області масиву даних і порівнянні його із шуканим. У разі рівності елементів повертається індекс, а якщо значення елементу посередині масиву більше від шуканого, то пошук продовжується в лівій частині масиву, якщо інакше – в

правій. Далі варто провести перевірку на сходження меж робочої області, і якщо це так, то значення не знайдене, якщо ні – перехід на перший крок [1]. Через те, що область пошуку зменшується експоненціально (поділ навпіл), кількість необхідних кроків k задовольняє $n/2^k = 1 \rightarrow 2^k = n \rightarrow k = \log_2 n$ [4]. Тобто в найгіршому й середньому випадках алгоритм потребує приблизно $\log_2 n$ порівнянь – звідси і складність $O(\log n)$.

Згідно зі стандартним описом алгоритму, бінарний пошук виконується на упорядкованому масиві, і на кожному кроці пошукова область ділиться навпіл – це дає змогу ефективно зменшувати кількість елементів для перевірки. У найгіршому та середньому випадках алгоритм виконує приблизно $\log_2 n$ порівнянь, де n – розмір масиву, що забезпечує часову складність $O(\log n)$. Водночас додаткових витрат пам'яті майже немає (просторова складність $O(1)$), оскільки алгоритм потребує лише кілька змінних для меж та середнього індексу [5].

Бінарний пошук особливо корисний для роботи з великими впорядкованими наборами даних, де перевіряти кожен елемент послідовно було б занадто повільно. Завдяки тому, що на кожному кроці область пошуку скорочується рівно вдвічі, алгоритм дає змогу значно економити час і швидко знаходити потрібний елемент навіть у масивах із десятків чи сотень тисяч значень. У таких випадках класичний лінійний пошук втрачає ефективність, тоді як бінарний демонструє стабільно хорошу продуктивність [2].

Основними перевагами бінарного пошуку є висока швидкодія, низькі витрати пам'яті та простота реалізації в системах. Він використовується для визначення індексу елементу в масиві. За умови правильного розуміння обчислення насправді доволі прості. Для цього методу оптимальними є великі набори даних. Для величезного розміру набору даних він є найбільш ефективним [3].

Варто зауважити, що існують модифікації алгоритму (наприклад, під час обчислення середнього індексу з округленням вгору), які можуть бути корисні під час роботи з масивами, що містять дублікати – тоді алгоритм може повертати, наприклад, правий або лівий з еквівалентних індексів. Водночас бінарний пошук – не універсальний: якщо дані часто модифікуються (вставки, видалення) або потрібен швидкий довільний доступ без упорядкування, доцільно використовувати інші структури (наприклад, хеш-таблиці) [5].

Отже, бінарний пошук забезпечує логарифмічну часову складність завдяки поділу масиву навпіл на кожному кроці. Алгоритм ефективний для великих впорядкованих наборів даних, має низькі витрати пам'яті та просту реалізацію. Основним обмеженням є необхідність попереднього сортування та менш ефективна робота в структурах із частими змінами. Алгоритм залишається базовим інструментом для оптимізації пошуку в сучасних обчислювальних системах.

Список використаних джерел

1. C#.Net: Бінарний пошук. URL: https://programm.top/uk/c-sharp/algorithm/search/binary-search/ (дата звернення: 28.11.2025).

2. Linear Search vs Binary Search. URL: https://www.geeksforgeeks.org/dsa/linear-search-vs-binary-search/ (дата звернення: 28.11.2025).

3. What are the advantages and disadvantages of binary search? *Quora*. URL: https://www.quora.com/What-are-the-advantages-and-disadvantages-of-binary-search (дата звернення: 28.11.2025).

4. Algorithms – How come the time complexity of Binary Search is log *n*. URL: https://math.stackexchange.com/questions/3224473/how-come-the-time-complexity-of-binary-search-is-log-n (дата звернення: 28.11.2025).

5. Binary search algorithm. URL: https://upload.wikimedia.org/wikiversity/en/e/ea/Binary_search_algorithm.pdf (дата звернення: 28.11.2025).

УДК 004.422.5

Коваль В. С., здобувач 2 курсу спеціальності
122 Комп'ютерні науки,
Фриз І. В., канд. фіз.-мат. наук,
старший викладач кафедри інформаційних технологій

РОЛЬ АЛГОРИТМІВ ТА СТРУКТУР ДАНИХ У СУЧАСНИХ ІТ-СИСТЕМАХ

Донецький національний університет імені Василя Стуса, м. Вінниця

Алгоритми та структури даних становлять основу будь-якого програмного забезпечення і визначають ефективність роботи інформаційних систем. Алгоритм є формальним описом послідовності дій, необхідних для розв'язання певної задачі, тоді як структура даних визначає спосіб організації, зберігання та обробки інформації. Саме від поєднання цих двох компонентів залежить продуктивність, масштабованість і надійність програмних рішень.

Сучасні структури даних поділяють на базові та абстрактні. До базових належать числа, логічні значення, символи, а до абстрактних – масиви, списки, стеки, черги, дерева, хеш-таблиці, графи. Масиви забезпечують швидкий доступ за індексом та часто використовуються у задачах сортування. Зв'язні списки ефективні під час частих вставок та видалень. Стек застосовується для зберігання історії викликів, реалізації рекурсії та механізмів відміни дій, тоді як черги актуальні у системах планування процесів та потокової обробки даних. Дерева та графи є основою алгоритмів пошуку, оптимізації маршрутів, роботи компіляторів та навігаційних систем.

Алгоритми, що працюють із різними структурами даних, оцінюються за допомогою нотації Big O, яка дає змогу визначити час виконання та обсяг використаної пам'яті залежно від розміру вхідних даних. Наприклад, алгоритм сортування злиттям має складність $O(n \log n)$, тоді як наївний пошук у масиві – $O(n)$. Порівняння таких характеристик дає змогу обирати найбільш ефективні методи для вирішення конкретних задач.

Важливою частиною аналізу алгоритмів є визначення компромісу між швидкодією та ресурсами пам'яті. Деякі структури даних забезпечують високу швид-

кість доступу (наприклад, хеш-таблиці), але потребують додаткової пам'яті. Інші, навпаки, є компактними, але повільнішими. Тому програміст повинен обирати структуру даних не лише за формальними характеристиками, а й з урахуванням особливостей прикладної задачі.

У сучасних мовах програмування реалізовано широкий спектр готових структур даних:

– Python – списки, словники, множини, deque, heapq;
– Java – колекції List, Set, Map, дерева, черги;
– C++ STL – вектори, списки, черги, множини, пріоритетні черги;
– C# – List, Dictionary, Stack, Queue, LinkedList.

Попри наявність готових інструментів, розробник повинен розуміти внутрішню архітектуру цих структур, щоб уникати неефективних рішень, наприклад, частого розширення масивів чи неправильного використання списків.

Алгоритми та структури даних залишаються незамінними у критичних галузях, зокрема у системах реального часу, штучному інтелекті, комп'ютерній графіці, криптографії, машинному навчанні. Від коректності їх проєктування залежить надійність та швидкодія всієї системи. Тому вивчення цих понять має велике значення для майбутніх фахівців у сфері інформаційних технологій.

Список використаних джерел

1. Introduction to algorithms / T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein. 4th ed. Cambridge, MA: The MIT Press, 2022. 1312 p.
2. Skiena S. The Algorithm Design Manual. Springer. 2020. 810 p.
3. Sedgewick R., Wayne K. Algorithms. 4th ed. Boston: Addison-Wesley. 2011. 955 p.

УДК 004.422.5

Головня С. О., здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,
Потапова Н. А., канд. екон. наук, доцент, доцент кафедри інформаційних технологій

СТРАТЕГІЯ «РОЗДІЛЯЙ І ВОЛОДАРЮЙ»: КОНЦЕПТУАЛЬНА ОСНОВА ШВИДКИХ АЛГОРИТМІВ

Донецький національний університет імені Василя Стуса, м. Вінниця

Однією з найважливіших парадигм у сучасній комп'ютерній науці є стратегія «розділяй і володарюй». Цей підхід забезпечує концептуальну основу для створення ефективних алгоритмів, наприклад, сортування злиттям та швидке сортування, даючи змогу вирішувати складні обчислювальні задачі шляхом їх декомпозиції. Розуміння цієї стратегії вимагає аналізу її теоретичної структури, формальних методів синтезу та застосування у багатовимірних просторах.

У фундаментальній теорії алгоритмів метод «розділяй і володарюй» визначається як процес, що складається з трьох ключових етапів: розподілу, підко-

рення та об’єднання. На етапі розподілу задача розбивається на кілька підзадач меншого розміру, які є подібними до початкової. Етап підкорення передбачає рекурсивне розв’язання цих підзадач; якщо ж розмір підзадачі доволі малий, вона розв’язується безпосередньо. Зрештою етап об’єднання формує підсумковий розв’язок на основі рішень усіх підзадач. Ефективність цього підходу часто оцінюється за допомогою рекурентних співвідношень, які описують час виконання алгоритму через час виконання його частин [1].

Для суворої побудови таких алгоритмів застосовуються формальні методи синтезу. Д. Р. Сміт пропонує розглядати структуру алгоритмів класу «розділяй і володарюй» через програмні схеми, де функціональність алгоритму виводиться з його структури та функціональності компонентів (операторів декомпозиції та композиції) [2]. Ключовим елементом тут є «сильний принцип редукції задачі», який стверджує: якщо вхідні дані коректно декомпозуються, підзадачі розв’язуються правильно, і результати коректно об’єднуються, то отриманий результат буде правильним розв’язком вихідної задачі [2]. Такий підхід дає змогу формально виводити специфікації для підалгоритмів, гарантуючи коректність усієї програми.

Потужність парадигми особливо яскраво проявляється під час розв’язання задач у багатовимірному просторі. Дж. Бентлі демонструє, що багатовимірний метод «розділяй і володарюй» дає змогу знизити часову складність багатьох геометричних задач (як-от пошук найближчої пари точок або побудова емпіричної функції розподілу) з наївного $O(N^2)$ до $O(N \log N)$ [3]. Алгоритм для k-вимірного простору будується рекурсивно: задача для N точок у k-вимірному просторі зводиться до двох задач для $N/2$ точок у k-вимірному просторі та однієї задачі для N точок у $(k-1)$-вимірному просторі [3]. Ця рекурсія за розмірністю дає змогу ефективно уникати комбінаторного вибуху складності.

Отже, стратегія «розділяй і володарюй» є не просто набором евристик, а строгою математичною концепцією. Вона поєднує в собі чітку алгоритмічну структуру [1], можливості формального синтезу програм [2] та високу ефективність під час обробки багатовимірних даних [3], що робить її незамінним інструментом у сучасному алгоритмічному проєктуванні.

Список використаних джерел

1. Introduction to algorithms / C. Stein et al. Cambridge, MA: The MIT Press. 2009. 1292 p.

2. Smith D. R. The design of divide and conquer algorithms. Science of computer programming. 1985. Vol. 5. P. 37–58. URL: https://doi.org/10.1016/0167-6423(85)90003-6 (дата звернення: 30.11.2025).

3. Bentley J. L. Multidimensional divide-and-conquer. Communications of the ACM. 1980. Vol. 23, № 4. P. 214–229. URL: https://doi.org/10.1145/358841.358850 (дата звернення: 30.11.2025).

СЕКЦІЯ 2
ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ ТА МЕРЕЖІ

УДК 004.422.5

Засипка Є. В., здобувач 2 курсу
спеціальності 122 Комп'ютерні науки,
Павлюк О. А., канд. техн. наук, старший
викладач кафедри інформаційних технологій

АЛГОРИТМ ОБРОБКИ ДАНИХ У ШТУЧНОМУ ІНТЕЛЕКТІ

Донецький національний університет імені Василя Стуса, м. Вінниця

Алгоритм обробки даних у штучному інтелекті – це чітко визначена послідовність інструкцій та процедур, спрямованих на перетворення сирих, неструктурованих даних у чистий, структурований та анотований формат, придатний для ефективного навчання моделей штучного інтелекту [1]. Цей алгоритм є критично важливим конвеєром попередньої підготовки, який забезпечує якість вхідної інформації, а отже, і кінцеву продуктивність і надійність інтелектуальної системи [2].

Алгоритм обробки даних у штучному інтелекті є фундаментальним процесом, що перетворює сирі неструктуровані дані на чисту та придатну для навчання моделей інформацію. Актуальність цієї теми зумовлена тим, що якість будь-якої інтелектуальної системи безпосередньо залежить від якості вхідних даних, відтак алгоритм їх обробки виступає критично важливою ланкою у конвеєрі розробки штучного інтелекту [1].

Основним завданням алгоритму обробки даних є підготовка інформаційного масиву, який дасть змогу моделі штучного інтелекту ефективно виявляти закономірності та ухвалювати точні прогнози. Цей процес можна розглядати як послідовність взаємопов'язаних етапів. Першим кроком завжди є збір та отримання даних із різноманітних джерел, серед яких можуть бути структуровані бази даних, текстові документи, зображення, аудіозаписи та потоки інформації з сенсорів. Ключовою вимогою на цьому етапі є забезпечення релевантності та достатнього обсягу вибірки [3].

Наступна фаза, попередня обробка та очищення, є найбільш трудомісткою. Вона включає в себе низку важливих процедур, спрямованих на підвищення якості даних. Необхідно вирішити проблему пропущених значень шляхом їх видалення, наприклад, заповненням середніми значеннями. Кодування категоріальних змінних перетворює текстовий опис на числовий формат за допомогою методів, як от One-Hot Encoding. Критично важливим є видалення дублікатів та ідентифікація аномальних значень (викидів), які можуть суттєво спотворити результат навчання. Також дані часто потребують нормалізації або стандартизації для приведення всіх числових ознак до єдиного масштабу, що значно пришвидшує процес навчання моделі [2].

Фінальним етапом підготовки часто виступає відбір ознак, оскільки не всі змінні однаково корисні для моделі. Надмірна кількість ознак веде до перенавчання, коли модель запам'ятовує шум замість реальних закономірностей. Методи відбору варіюються від простих фільтраційних, що оцінюють статистичну зв'язність ознаки з цільовою змінною, до складніших вкладених методів, які оціню-

ють важливість ознаки на основі продуктивності самої моделі. Варто зазначити, що специфіка алгоритму обробки значно відрізняється для різних типів даних. Для зображень це зведення до одного масштабу яскравості пікселів та аугментація; для природної мови це токенізація, видалення стоп-слів і лематизація; для часових рядів – видалення тренду та сезонності [2].

Окремої уваги заслуговують проблеми, пов'язані з якістю самих вихідних даних. Алгоритми обробки не можуть усунути системні упередження, які вже містяться в даних. Наприклад, якщо історичні дані про прийом на роботу відображають певну соціальну нерівність, модель лише відтворить і посилить це упередження. До того ж складність масштабування становить серйозну проблему – методи, що ефективно працюють на невеликих наборах даних, часто виявляються непридатними для роботи з великими даними через обмеження обчислювальних ресурсів. Отже, навіть ідеально виконаний алгоритм обробки даних не може гарантувати успіх, якщо вихідні дані містять фундаментальні вади. Це підкреслює той факт, що обробка даних залишається не лише технічною, але значною мірою методологічною та навіть філософською проблемою, де інтуїція та розуміння предметної області часто виявляються важливішими за формальні математичні процедури [3].

У висновку можна констатувати, що алгоритм обробки даних є не допоміжним, а центральним процесом у створенні ефективного штучного інтелекту. Це ітеративний і часто творчий ланцюг дій, що перетворює хаотичну інформацію на структуроване знання. Майбутній розвиток цієї галузі, ймовірно, буде пов'язаний з подальшою автоматизацією всього конвеєра обробки в межах концепції AutoML, а також із розробкою ефективніших методів роботи з великими та невпорядкованими даними. Інвестиції в якісну підготовку даних завжди є найбільш ефективною стратегією для побудови точних, надійних та робочих систем штучного інтелекту [2].

Список використаних джерел

1. ДСТУ 2860:2014. Системи штучного інтелекту. Терміни та визначення. На заміну ДСТУ 2860-94. Київ, 2014. URL: https://online.budstandart.com/ua/catalog/doc-page.html?id_doc=25034 (дата звернення: 22.11.2025).

2. Russell S. J., Norvig P. Artificial intelligence: a modern approach. 4th ed. Hoboken: Pearson. 2020.

3. Федорін І. В. Проектування та аналіз обчислювальних алгоритмів: вступ до алгоритмів: навчальний посібник. Київ: КПІ ім. Ігоря Сікорського. 2022.

УДК 004.8

Олексієнко Є. О., здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,
Павлюк О. А., канд. техн. наук, старший викладач кафедри інформаційних технологій

АЛГОРИТМИ ВИКОРИСТАННЯ ШТУЧНОГО ІНТЕЛЕКТУ У ЧАТ-БОТАХ

Донецький національний університет імені Василя Стуса, м. Вінниця

Штучний інтелект (ШІ) – це галузь комп'ютерних наук, що зосереджена на створенні систем, здатних імітувати людську інтелектуальну діяльність. Ця технологія широко застосовується у розробці чат-ботів для імітації природної людської розмови. У сучасному цифровому середовищі чат-боти на основі ШІ відіграють ключову роль, забезпечуючи автоматизацію обслуговування клієнтів, надання інформації та персоналізовану взаємодію у режимі реального часу [1; 2].

Основою інтелектуальних чат-ботів є обробка природної мови (NLP). Цей набір алгоритмів дає змогу машині «читати» та інтерпретувати людський текст. Ключовим компонентом є розуміння природної мови (NLU), яке фокусується на визначенні наміру (intent) користувача та вилученні сутностей (entities). Наприклад, у запиті «Яка погода у Львові завтра?» NLU ідентифікує намір «запит погоди», а «Львів» і «завтра» – як сутності [3].

Управління діалогом (Dialogue Management) є наступним критичним етапом. Цей компонент відповідає за підтримання контексту розмови та прийняття рішень про те, яку дію виконати або яку відповідь надати. У простих ботах це може бути реалізовано через жорсткі дерева рішень. Однак у складних ШІ-системах використовуються статистичні моделі або навчання з підкріпленням (Reinforcement Learning) для вибору оптимальної відповіді [4].

Важливим аспектом є генерація природної мови (NLG), тобто процес, за допомогою якого чат-бот формулює свою відповідь. Ранні версії ботів покладалися на заздалегідь прописані шаблони, куди просто підставлялися витягнуті сутності. Сучасні ШІ-алгоритми, зокрема рекурентні нейронні мережі (RNN) та моделі на базі LSTM, здатні генерувати текст динамічно, створюючи більш гнучкі та природні відповіді [4].

Справжній прорив у можливостях чат-ботів пов'язаний з появою архітектури Transformer та великих мовних моделей (LLM), як-от BERT або GPT. Ці моделі, навчені на величезних масивах текстових даних, здатні розуміти глибокі семантичні зв'язки, вловлювати тонкі відтінки мови та генерувати надзвичайно зв'язний і контекстуально доречний текст [5].

Сучасні ШІ-алгоритми також дають змогу чат-ботам навчатися в процесі взаємодії. Використовуючи методи машинного навчання, бот може аналізувати відгуки користувачів (наприклад, позитивні чи негативні реакції на відповіді) для коригування своєї поведінки. Це дає змогу персоналізувати спілкування та з часом покращувати якість своїх відповідей [4].

Однією з основних проблем використання ШІ у чат-ботах є висока обчислювальна складність та вартість навчання і експлуатації великих мовних моделей. Вони вимагають значних апаратних ресурсів. Іншою серйозною проблемою є схильність генеративних моделей до «галюцинацій» – генерування правдоподібної, але фактично неправильної інформації. Для боротьби з цим застосовуються новітні архітектури, як-от RAG (Retrieval-Augmented Generation), які «заземлюють» відповіді на перевірених джерелах даних [3].

Отже, алгоритми штучного інтелекту є ядром сучасних чат-ботів, що перетворює їх на потужні інструменти для взаємодії. Їх правильне застосування забезпечує реалізацію інтелектуальних, адаптивних та корисних діалогових систем, що є ключовим аспектом сучасної цифрової комунікації.

Список використаних джерел

1. Zhang Z., Wang L., Li Y. A Survey on Dialogue Systems: Recent Advances and New Frontiers. *arXiv preprint.* URL: https://surl.li/gvpqmk (дата звернення: 21.10.2025).

2. Gupta R., Sadeghian A. Artificial Intelligence Chatbots: A Survey of Classical versus Deep Machine Learning Techniques. *arXiv preprint.* URL: https://surl.li/tphmch (дата звернення: 21.10.2025).

3. Chen X., Luo M., Zhang H. Recent Advances in Deep Learning Based Dialogue Systems: A Systematic Survey. *ACM Computing Surveys.* URL: https://surl.li/myritm (дата звернення: 21.10.2025).

4. Khan S., Malik A., Ahmed N. Learning towards Conversational AI: A Survey. *IEEE Access.* URL: https://surl.li/uyxeno (дата звернення: 21.10.2025).

5. Wang J., Lee K., Park S. A Complete Survey on LLM-based AI Chatbots. *arXiv preprint.* URL: https://surl.li/onpgwy (дата звернення: 21.10.2025).

УДК 004.8:004.421.2

Шендрик В. О., здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,
Потапова Н. А., канд. екон. наук, доцент, доцент кафедри інформаційних технологій

АЛГОРИТМІЧНІ СТРУКТУРИ ТА МЕТОДИ ОБРОБКИ ДАНИХ У ПОБУДОВІ ІНТЕЛЕКТУАЛЬНИХ АГЕНТІВ

Донецький національний університет імені Василя Стуса, м. Вінниця

Інтелектуальні агенти є центральною ланкою у розвитку систем штучного інтелекту. Вони поєднують алгоритмічні механізми прийняття рішень з адаптивними моделями, здатними до навчання та самоорганізації. Актуальність теми полягає у необхідності переходу від жорстких ієрархічних систем до гнучких генеративних моделей, які можуть ефективно працювати в умовах невизначеності.

Ранні дослідження у цій сфері зосереджувалися на структурованих системах, де кожен рівень виконував чітко визначені завдання. Проте сучасні фундаментальні праці, зокрема С. Рассела та П. Норвіга, вказують на необхідність використання ймовірнісних методів та машинного навчання для підвищення автономності агентів [1]. Значний внесок у розвиток адаптивних систем зробили методи навчання з підкріпленням, детально описані Р. Саттоном та Е. Барто [2]. Останні дослідження також фокусуються на використанні великих мовних моделей (LLM) у мультиагентних системах [3].

Класичні ієрархічні системи використовують послідовний розподіл рівнів: сприйняття, обробка, прийняття рішень та дія. Такі структури ефективні у стабільних середовищах, але втрачають гнучкість у динамічних умовах. Сучасні підходи базуються на генеративних та гібридних архітектурах. Алгоритмічна структура генеративного агента включає генератор дій (політика), оцінювача (функція винагороди) та механізм оновлення знань. Це дає змогу агенту не лише реагувати на стимули, а й самостійно моделювати можливі стани. Особливу увагу в сучасних розробках приділяють інтеграції когнітивних моделей, які допомагають кооперативним агентам краще взаємодіяти між собою [4]. Для навчання таких агентів використовуються методи Deep Learning та генеративні моделі (GAN, GPT). Це дає змогу реалізувати складні сценарії поведінки, що є критично важливим для створення навчальних середовищ та симуляцій мультиагентних систем [5]. Найперспективнішими є гібридні підходи, де низькорівневі модулі обробляють сенсорні дані, а високорівневі генеративні моделі формують стратегію поведінки.

Еволюція від ієрархічних до генеративних структур допомагає створити агентів, здатних до автономного навчання та адаптації. Використання генеративних алгоритмів та нейромережевих підходів значно розширює можливості інтелектуальних систем, даючи їм змогу вирішувати задачі у відкритих та непередбачуваних середовищах.

Список використаних джерел

1. Russell S. J., Norvig P. Artificial intelligence: a modern approach. 4th ed. Hoboken: Pearson, 2021.

2. Sutton R. S., Barto A. G. Reinforcement learning: an introduction. 2nd ed. Cambridge, MA: The MIT Press, 2018.

3. Nathalia N., Alencar P., Cowan D. Self-adaptive large language model (LLM)-based multiagent systems. 2023. URL: https://arxiv.org/abs/2307.06187 (дата звернення: 30.11.2025).

4. Nguyen T. N., Phan D. N., Gonzalez C. Learning in cooperative multiagent systems using cognitive and machine models. 2023. URL: https://arxiv.org/abs/2308.09219 (дата звернення: 30.11.2025).

5. Casals A., Brandão A. A. F. Hecate: an ECS-based framework for teaching and developing multi-agent systems. 2025. URL: https://arxiv.org/abs/2509.06431 (дата звернення: 30.11.2025).

УДК 004.01

Грищишен О. Ю., здобувач 1 курсу
ОС «Магістр» ФЗ Комп'ютерні науки,
Січко Т. В., канд. техн. наук, доцент,
доцент кафедри інформаційних технологій

АНАЛІЗ ЕФЕКТИВНОСТІ ВИКОРИСТАННЯ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ ДЛЯ ЗАДАЧ КЛАСИФІКАЦІЇ ТВАРИН

Донецький національний університет імені Василя Стуса, м. Вінниця

У сучасному світі технології комп'ютерного зору набувають критичного значення для автоматизації процесів у сільському господарстві, екологічному моніторингу та ветеринарії. Автоматична ідентифікація тварин дає змогу оптимізувати облік популяцій, виявляти захворювання на ранніх стадіях та забезпечувати контроль за міграцією диких видів. Враховуючи різноманітність умов зйомки (освітлення, ракурс, фон), актуальним завданням є розробка та дослідження програмних засобів, здатних забезпечити високу точність класифікації за помірних вимог до обчислювальних ресурсів.

Традиційні методи машинного навчання, що базуються на ручному виділенні ознак (HOG, SIFT), поступово витісняються методами глибокого навчання. Як зазначають Wang et al., згорткові нейронні мережі (CNN) демонструють значно вищу стійкість до варіативності вхідних даних [1]. Зокрема, архітектури типу ResNet, завдяки використанню залишкових зв'язків, дають змогу ефективно тренувати глибокі мережі, уникаючи проблеми згасання градієнта, що підтверджено у дослідженнях He et al. [2].

Мета роботи полягає в дослідженні ефективності застосування методу трансферного навчання (Transfer Learning) на базі архітектури ResNet50 для створення програмного модуля класифікації зображень тварин та оцінці точності роботи такої системи на локальних обчислювальних станціях.

Для вирішення задачі класифікації було обрано архітектуру ResNet50, попередньо навчену на наборі даних ImageNet. Цей підхід дає змогу використовувати вже сформовані фільтри для виділення низькорівневих ознак, що значно пришвидшує процес навчання та підвищує точність на спеціалізованих датасетах. Програмна реалізація виконана мовою Python з використанням бібліотек TensorFlow та Keras. Попередня обробка вхідних зображень включає масштабування до розміру 224×224 пікселі, нормалізацію та конвертацію у масиви NumPy. Для взаємодії з користувачем розроблено графічний інтерфейс на базі бібліотеки Tkinter, який дає змогу завантажувати зображення та отримувати результати прогнозування в режимі реального часу. Під час експериментального дослідження було проведене тестування розробленого модуля на наборі із 50 тестових зображень різної складності (різні види тварин, часткове перекриття об'єктів, складний фон). Середня точність класифікації становила 92 % (46 правильно класифікованих зображень із 50). Найвищі показники достовірності (понад 95 %) зафіксовано для зображень із чітким контрастом та центральним розташуванням об'єкта. Порів-

няльний аналіз показав, що використання локального модуля на базі ResNet50 є економічно вигіднішим, порівняно з хмарними API (Google Cloud Vision, Microsoft Custom Vision) для задач, що не вимагають обробки надвеликих масивів даних у хмарі, водночас забезпечуючи зіставну точність розпізнавання поширених видів тварин.

Результати дослідження підтверджують доцільність використання архітектури ResNet50 із застосуванням трансферного навчання для задач ідентифікації тварин. Розроблений програмний засіб забезпечує високу точність (92 %) та швидкодію, достатню для використання у прикладних задачах. Подальший розвиток дослідження вбачається в адаптації моделі для обробки відеопотоків та інтеграції алгоритмів детекції об'єктів (YOLO) для задач підрахунку поголів'я.

Список використаних джерел

1. Research on animal image recognition based on improved CNN model / J. Wang, Y. Sun, Y. Bao, J. Zhao. *Journal of Intelligent & Fuzzy Systems.* 2020. Vol. 38, № 3. P. 2881–2892.

2. Deep residual learning for image recognition / K. He, X. Zhang, S. Ren, J. Sun. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 2016. P. 770–778. URL: https://arxiv.org/abs/1512.03385 (дата звернення: 30.10.2025).

3. Animal recognition and identification with deep convolutional neural networks for automated wildlife monitoring / H. Nguyen, S. Maclagan, T. Nguyen. *Proceedings of the IEEE International Conference on Data Science and Advanced Analytics (DSAA).* 2017.

УДК 004.422.5

Заболотна У. К., здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,
Потапова Н. А., канд. екон. наук, доцент, доцент кафедри інформаційних технологій

ГЕНЕТИЧНІ АЛГОРИТМИ ТА ЇХ ЗАСТОСУВАННЯ

Донецький національний університет імені Василя Стуса, м. Вінниця

Генетичні алгоритми (GA) – це адаптивні евристичні алгоритми пошуку, що належать до ширшого класу еволюційних алгоритмів. Вони базуються на принципах природного добору та генетики, імітуючи процеси еволюції живих організмів. Це розумна експлуатація випадкових пошуків, наданих історичними даними для спрямування пошуку в область кращої продуктивності в просторі рішень.

Зазвичай генетичні алгоритми застосовують для розв'язання задач оптимізації та пошуку. Ці алгоритми кодують потенційне рішення певної задачі у вигляді простої структури, що нагадує хромосому, та застосовують оператори рекомбі-

нації, щоб зберегти критичну інформацію. Генетичні алгоритми часто розглядають як оптимізатори функцій, хоча спектр задач, де вони застосовуються, значно ширший.

Реалізація генетичного алгоритму починається зі створення популяції – зазвичай випадкової – хромосом. Далі кожен індивід оцінюється за певною метрикою якості, яку ще називають функцією пристосованості (fitness). Рішення, які демонструють кращу якість, отримують більше шансів «передати свої гени» наступним поколінням. Отже, добір відбувається відносно всієї поточної популяції.

Генетичні алгоритми імітують процес природного відбору, що означає, що ті види, які можуть адаптуватися до змін у своєму середовищі, можуть виживати, розмножуватися і переходити до наступного покоління. Інакше кажучи, вони імітують «виживання найбільш пристосованих» серед людей послідовних поколінь для вирішення проблеми. Кожне покоління складається з населення індивіда, і кожна людина являє собою точку в пошуковому просторі та можливе рішення. Кожна людина представлена у вигляді рядка символа / ціле число / плавання / біт.

Населення людей зберігається в пошуковому просторі. Кожна людина являє собою рішення в пошуковому просторі для даної проблеми. Кожна людина кодується як кінцевий вектор довжини (аналогічний до хромосоми) компонентів. Ці змінні компоненти аналогічні генам. Отже, хромосома (індивідуальна) складається з декількох генів (змінних компонентів).

Після створення початкового покоління алгоритм розвиває генерацію, використовуючи такі оператори:

– ***Оператор відбору:*** Функція відбору визначає, які особини з популяції будуть використовуватися для створення нового покоління. Її головна ідея – віддати перевагу особам із вищою пристосованістю, щоб саме вони передавали свої гени далі. Це імітує природний добір: найкраще пристосовані особини мають більші шанси вижити й розмножитися.

– Кожен метод має свої переваги і використовується залежно від характеру задачі та бажаного рівня селективного тиску.

– ***Оператор кросовера (схрещування):*** Кросовер створює нових особин (потомство), поєднуючи генетичну інформацію двох батьків. Мета – успадкувати корисні риси обох та покращувати різноманітність популяції.

– ***Оператор мутації:*** Мутація додає до хромосом потомства випадкові зміни, підтримуючи генетичну різноманітність популяції. Це запобігає передчасній конвергенції – ситуації, коли всі особини стають надто схожими і алгоритм перестає знаходити кращі рішення.

Мутація зазвичай застосовується з малою ймовірністю, але її роль є критичною для дослідження нових ділянок простору рішень.

Розглянемо приклад використання генетичного алгоритму (рис. 2).

У цій задачі генетичний алгоритм шукає число, найближче до 100. Популяція – це масив випадкових чисел, функція пристосованості оцінює близькість до 100, кросовер бере два найкращі числа і створює середнє потомство, а мутація додає випадкове число для різноманітності. Алгоритм повторюється 30 поколінь, поступово наближаючись до цільового числа. Користь GA в тому, що він демонструє пошук оптимальних рішень через відбір, кросовер і мутацію, ефективний для задач, де простий перебір не підходить, імітуючи еволюційні процеси.

```
using System;
using System.Linq;

0 references
class Program
{
    static Random r = new Random();
    1 reference
    static int Fitness(int x) => Math.Abs(100 - x);
    1 reference
    static int Mutate(int x) => x + r.Next(-5, 6);
    1 reference
    static int Cross(int a, int b) => (a + b) / 2;

    0 references
    static void Main()
    {
        int[] pop = Enumerable.Range(0, 10).Select(_ => r.Next(0, 200)).ToArray();

        for (int gen = 0; gen < 30; gen++)
        {
            pop = pop.OrderBy(x => Fitness(x)).ToArray();
            Console.WriteLine($"Gen {gen}: best = {pop[0]}");

            int p1 = pop[0], p2 = pop[1];
            pop = pop.Select(_ => Mutate(Cross(p1, p2))).ToArray();
        }
    }
}
```

Рисунок 1 – Приклад використання генетичного алгоритму

Отже, генетичні алгоритми – це еволюційний метод пошуку, який імітує природний добір, щоб поступово покращувати розв'язки. Вони ефективні для складних задач, де важко знайти оптимальний варіант звичайними методами. Завдяки поєднанню випадковості та відбору генетичні алгоритми здатні швидко знаходити якісні наближення оптимуму й широко застосовуються в інженерії, науці та штучному інтелекті.

Список використаних джерел

1. Генетичні алгоритми. GeeksforGeeks. URL: https://www.geeksforgeeks.org/dsa/genetic-algorithms/ (дата звернення: 25.11.2025).

2. Алгоритм: повний посібник з реалізації Python. DataCamp. URL: https://www.datacamp.com/tutorial/genetic-algorithm-python (дата звернення: 25.11.2025).

3. Огляд застосування генетичних алгоритмів до прогнозування цін на товари. URL: https://www.mdpi.com/2227-7099/9/1/6 (дата звернення: 25.11.2025).

4. A Genetic Algorithm Tutorial. URL: https://www.cs.jhu.edu/~ayuille/courses/Stat202CSpring10/ga_tutorial.pdf (дата звернення: 25.11.2025).

УДК 004.422.5

Носалюк О. І., здобувач 2 курсу
спеціальності 122 Комп'ютерні науки,
Потапова Н. А., канд. екон. наук доцент,
доцент кафедри інформаційних технологій

ЗАСТОСУВАННЯ АЛГОРИТМІВ ШТУЧНОГО ІНТЕЛЕКТУ У РОЗРОБЦІ СУЧАСНИХ ВІДЕОІГОР

Донецький національний університет імені Василя Стуса, м. Вінниця

Штучний інтелект (ШІ) є одним із найважливіших напрямів розвитку сучасних комп'ютерних технологій, зокрема у сфері відеоігор. Алгоритми ШІ дають змогу створювати складні, динамічні системи, які імітують інтелектуальну поведінку, забезпечуючи реалістичність, варіативність і адаптивність ігрового процесу.

Одним із найпоширеніших напрямів використання ШІ у відеоіграх є моделювання поведінки NPC (не гравців). Алгоритми дерева рішень, машинного навчання та підкріплення (Reinforcement Learning) дають змогу створювати персонажів, які навчаються реагувати на дії користувача, змінюють стратегію поведінки й приймають рішення в реальному часі. Це підвищує рівень занурення гравця у віртуальний світ.

У розробці відеоігор також широко використовуються алгоритми пошуку та навігації, як-от A*, Дейкстри або D* Lite. Вони забезпечують ефективне пересування персонажів і NPC складними ігровими середовищами, що особливо важливо у відкритих світах. Поєднання цих алгоритмів із нейронними мережами дає можливість реалізувати адаптивну маршрутизацію, коли персонажі самостійно коригують шляхи залежно від змін у грі.

Сучасні ігри активно застосовують алгоритми процедурної генерації контенту – рівнів, карт, квестів і навіть сюжетів. Методи генеративних нейронних мереж (GAN), або еволюційних алгоритмів, дають змогу створювати унікальний контент без прямого втручання розробника, що значно зменшує витрати часу та ресурсів.

Ще одним важливим аспектом є аналіз поведінки гравців. Алгоритми кластеризації, прогнозування та рекомендаційні системи допомагають персоналізувати ігровий досвід, пропонуючи користувачеві оптимальний рівень складності, сюжет чи партнера для гри.

Водночас застосування ШІ має свої виклики: висока обчислювальна складність, потреба у великих масивах навчальних даних та необхідність оптимізації алгоритмів для роботи в реальному часі. Однак подальший розвиток методів ШІ відкриває нові можливості для створення ще більш інтелектуальних, реалістичних і захопливих відеоігор.

Список використаних джерел

1. Artificial intelligence in game development. Medium. URL: https://medium.com/artificial-intelligence-in-games (дата звернення: 30.10.2025).

2. Artificial intelligence and games / G. N. Yannakakis, J. Togelius. Cham: Springer, 2018.

3. Game AI Pro: Collected wisdom of game AI professionals / ed. by S. Rabin. Boca Raton: CRC Press, 2013.

4. AI in modern game design. Analytics Insight. URL: https://www.analyticsinsight.net/ai-in-modern-game-design (дата звернення: 30.10.2025).

УДК 004.738.5

Бєглова А. В., здобувачка 2 курсу
ОС «Магістр» спеціальності
F3 Комп'ютерні науки,
Веселовська Н. Р., д-р техн. наук, професор,
професор кафедри інформаційних технологій

ЗАСТОСУВАННЯ ГТУЧНОГО ІНТЕЛЕКТУ ДЛЯ АВТОМАТИЧНОГО ГЕНЕРУВАННЯ UI-ТЕСТІВ ВЕБДОДАТКІВ

Донецький національний університет імені Василя Стуса, м. Вінниця

Стрімке зростання складності вебдодатків формує потребу у масштабованих та ефективних підходах до тестування їх користувацьких інтерфейсів. Традиційні методи створення UI-тестів потребують значних часових та людських ресурсів, що сповільнює розробку та ускладнює підтримку тестових сценаріїв. У таких умовах актуальним стає використання інтелектуальних систем, здатних автоматично генерувати тестові сценарії та адаптувати їх до змін в інтерфейсі.

Приблизно 30–50 % витрат на забезпечення якості у великих проєктах припадає саме на створення та підтримку UI-тестів. Класичні сценарії є крихкими, залежать від структури DOM та потребують постійного оновлення. Штучний інтелект дає змогу автоматизувати ці процеси, зменшити навантаження на команду тестування та підвищити стабільність UI-тестів. Сучасні моделі генерації коду та аналізу інтерфейсів створюють можливість адаптивного тестування без ручного опису сценаріїв.

Метою роботи є дослідити можливості застосування штучного інтелекту для автоматичного генерування UI-тестів вебдодатків та оцінити ефективність такого підходу, порівняно з традиційними методами створення тестових сценаріїв.

Сучасний розвиток інструментів автоматизації тестування призвів до появи спеціалізованих платформ, що використовують штучний інтелект саме для автоматичного генерування UI-тестів. Одним із найвідоміших рішень є Testim.io, який застосовує машинне навчання для аналізу інтерфейсу та автоматичного формування тестових сценаріїв на основі зафіксованих дій користувача. Завдяки алгоритмам ML система здатна самостійно обирати найбільш стабільні селектори та відновлювати тести після змін у DOM, що значно зменшує потребу у ручному супроводі. Інший яскравий приклад – платформа Mabl, яка комбінує аналіз DOM,

поведінкові моделі й комп'ютерний зір. Вона може генерувати UI-тести автоматично, прогнозувати потенційні зміни інтерфейсу, а також розширювати тестове покриття шляхом виявлення важливих користувацьких шляхів.

Помітну роль відіграє і Test.ai – система, що використовує нейронні мережі для розпізнавання елементів інтерфейсу на візуальному рівні. Такий підхід дає змогу платформі створювати тести, подібні до того, як би це робила людина-тестувальник: через «розуміння» зовнішнього вигляду сторінки, а не через прив'язку до селекторів, які часто змінюються. Так само Applitools розвиває напрям Autonomous Testing, що поєднує великі мовні моделі та візуальний аналіз інтерфейсу. Система може автоматично перетворювати текстові описи вимог на робочі UI-тести, формуючи як кроки сценарію, так і очікувані результати перевірок. Подібну логіку використовує і Katalon, де модулі штучного інтелекту допомагають генерувати початкові тестові сценарії на основі записаних дій користувача та оптимізують їх за допомогою автовідновлення селекторів [1].

Окрім повноцінних платформ, у сфері автоматизації набули поширення генератори, що інтегруються з наявними фреймворками та працюють на базі великих мовних моделей. Такі системи здатні автоматично створювати тести з текстових інструкцій або з аналізу HTML-структури сторінки, доповнювати їх відсутніми перевірками, та виявляти непокриті елементи інтерфейсу. Їх поява демонструє зміну парадигми: якщо раніше UI-тести вимагали повністю ручного програмування, то тепер значну частину роботи може виконувати ШІ, залишаючи інженеру роль коректора, а не автора коду.

Найбільш актуальне дослідження, що заслуговує уваги – GenIA-E2ETest. В ньому запропоновано використовувати генеративний ШІ для створення виконуваних end-to-end (E2E) UI-тестів на основі натурально-мовних описів. За результатами авторів: тестові скрипти, згенеровані системою, показали середню «completeness» і «correctness» приблизно 77 %, точність виконання («precision») – 82 %, а «recall» – 85 %. Причому потреба в ручних доробках була невеликою – в середньому лише ~10 % сценаріїв потребували підправлення після генерації. Це демонструє, що сучасні LLM + підходи до генерації можуть значно скоротити час на створення E2E-тестів, зменшити «вхідний поріг» для QA-інженерів і зробити автоматизацію більш доступною навіть для непрофільних користувачів [2].

Ще одним актуальним дослідженням є AutoQALLMs, яке комбінує LLM (наприклад, GPT-4, Claude, Grok) з аналізом HTML/DOM та регулярними виразами для автоматичного створення UI-тестів у Selenium середовищі. Система здатна «з нуля» (zero-shot) – без спеціального навчання чи шаблонів – приймати HTML сторінку, аналізувати її структуру, генерувати Selenium скрипт, а потім «очищати» / покращувати скрипт за допомогою regex-модуля, щоб підвищити стійкість до змін UI.

Згідно з оцінками на понад 30 різних веб-додатків, покриття UI-елементів досягало ~ 96 % (порівняно з ~ 98 % у ручних скриптів), а час генерації – значно менший, ніж написання вручну. Водночас експерти вказували, що хоча скрипти були робочими і їх швидко можна було запустити, вони іноді були надто «загальними» та пропускали логіку специфічних бізнес-валідацій, які зазвичай додає QA вручну. Це підкреслює важливу властивість сучасного AI-підходу: інструменти

створюють основу, але для глибокої перевірки потрібно доповнювати скрипти вручну, особливо коли йдеться про складну бізнес-логіку [3].

Огляд літератури (так званої «grey literature» + академічних праць) показує, що AI-інструменти для тестування (наприклад, mabl, Test.ai, Appvance, Testim та ін.) прийшли як спроба подолати проблеми «звичайної» автоматизації: висока вартість підтримки, громіздкість скриптів, залежність від стабільності UI та DOM, трудомісткість написання тестів. Їх переваги: спрощена генерація тестів (часто без коду), автоматичне оновлення скриптів у разі зміни UI (self-healing), інтеграція з CI/CD, можливість швидкого створення regression-suite навіть не командою QA [4].

Але існують і суттєві обмеження: для складної бізнес-логіки або нетривіальних сценаріїв (наприклад, MFA, динамічний контент, специфічні валідації) AI-згенеровані тести часто недостатні – їх треба доопрацьовувати вручну. Також – ризик «false positives / negatives», складність у підтримці, залежність від стабільності моделей, витрати на виклики AI/LLM.

Отже, впровадження штучного інтелекту в автоматичне генерування UI-тестів вебдодатків істотно підвищує ефективність і масштабованість процесу тестування. AI-орієнтовані інструменти здатні самостійно створювати та адаптувати тестові сценарії, зменшуючи залежність від ручної роботи і підвищуючи стійкість тестів до змін інтерфейсу. Результати сучасних досліджень підтверджують, що такі підходи забезпечують високу точність і скорочують витрати на підтримку автоматизації. Водночас залишаються завдання, які потребують експертного втручання, зокрема у сфері складної бізнес-логіки та нестандартних сценаріїв. Отже, найбільш ефективною стратегією є поєднання AI-генерації з професійним контролем якості з боку інженера.

Список використаних джерел

1. Ricca F., Marchetto A., Stocco A. AI-based test automation: A grey literature analysis. 2021.

2. GenIA-E2ETest: a generative AI-based approach for end-to-end test automation. 2025. URL: https://arxiv.org/abs/2510.01024 (дата звернення 20.11.2025).

3. AutoQALLMs: Automating Web Application Testing Using Large Language Models (LLMs) and Selenium. URL: https://www.mdpi.com/2073-431X/14/11/501 (дата звернення 20.11.2025).

4. Improving web element localization by using a large language model. URL: https://arxiv.org/abs/2310.02046 (дата звернення 20.11.2025).

УДК 004.8:519.83(043.2)

Зозуля О. Р., здобувачка 2 курсу
спеціальності 122 Комп'ютерні науки,
Січко Т. В., канд. техн. наук, доцент,
доцент кафедри інформаційних технологій

ІГРОВИЙ ШТУЧННИЙ ІНТЕЛЕКТ: МОДЕЛЮВАННЯ СТРАТЕГІЧНИХ ВЗАЄМОДІЙ

Донецький національний університет імені Василя Стуса, м. Вінниця

З розвитком комп'ютерної галузі, ІТ, поширенням штучного інтелекту, застосуванням нових технологій у щоденному житті у нас з'являються також нові проблеми. Сьогодні ми працюємо не з простими, відокремленими алгоритмами, а з цілими системами та надсистемами, що взаємодіють одна з одною. В таких системах рішення одного компонента впливає на всі інші, тому постає актуальне питання: як оптимально взаємодіяти в середовищі, де кожен учасник має власні цілі?

Теоретико-ігрові методи дослідження полягають в аналізі стратегічних взаємодій у ситуаціях, коли невизначеність викликана свідомими, зловмисними діями суперника.

Сучасному користувачу ПК та мережі Інтернет ця методологія, найімовірніше, знайома з відеоігор. Від звичних шахів до онлайн-шутерів у процес геймплею залучений ігровий штучний інтелект. Це набір програмних методик, який застосовується в комп'ютерних іграх для створення ілюзії інтелекту ігрових персонажів. Використання штучного інтелекту в іграх істотно впливає на ігровий процес, технічні вимоги та фінансові витрати. Тому розробникам доводиться шукати компроміс: створити цікавий та переконливий ШІ, але водночас не перевантажити систему й не збільшити бюджет. Через це ігровий ШІ суттєво відрізняється від «класичного» – у ньому часто застосовують різні спрощення й хитрощі. Наприклад, у шутерах занадто точні та миттєві реакції ботів були б неконкурентними для гравця, тому їх спеціально обмежують. Поняття чесного ШІ можна розглянути на прикладі World of Tanks на PC. ШІ для сервера повинен діяти як людина: так само підкорятись правилам, керувати технікою, взаємодіяти з гравцями. Тож перед розробниками постають певні завдання та проблеми: як навчити ШІ аналізувати карту, дії супротивників, чесно стріляти та рухатись? Для цього потрібно зібрати аналітику та створити кілька шарів статичних даних. Їх прораховують заздалегідь на карті, вони є «награним досвідом» ШІ. Потім на їх основі будують системи управління ботом [1]. Розробникам потрібно підготувати аналітичні дані, тобто яку інформацію надавати штучному інтелекту, створити апарат її використання та отримання нової аналітики, розробити компоненти для найрізноманітніших дій та сценаріїв. Створення чесного ігрового ШІ – складне ресурсомістке завдання, що вимагає великої кількості розрахунків. Рівень чесності ШІ залежить від розробників, за винятком ШІ, розробленого на нейронних мережах [1].

Теорія ігор тепер стала одним з ключових інструментів у розвитку машинного навчання. Яскравим прикладом є GAN – Generative Adversarial Networks.

Це комбінація двох нейронних мереж: дискримінатора та генератора. Генераторна нейронна мережа отримує вхідні зображення, які вона аналізує, а потім створює нові зразки зображень, зроблені так, щоб максимально точно відображати фактичні вхідні зображення. Після створення зображень їх надсилають до дискримінаторної нейронної мережі. Ця нейронна мережа оцінює надіслані їй зображення та класифікує їх як згенеровані зображення та фактичні вхідні зображення. Якщо зображення класифікується як оригінальне, глибока нейронна мережа (DNN) змінює свої параметри оцінювання. Якщо зображення класифікується як згенероване, воно відхиляється та повертається до GNN. Потім GNN змінює свої параметри, щоб покращити якість створеного зображення (рис. 1). Це конкурентний процес, який триває доти, доки обидві нейронні мережі не потребуватимуть жодних змін у своїх параметрах, і подальше покращення в обох нейронних мережах не стане можливим. Стан відсутності подальшого покращення – це рівновага Неша [2].

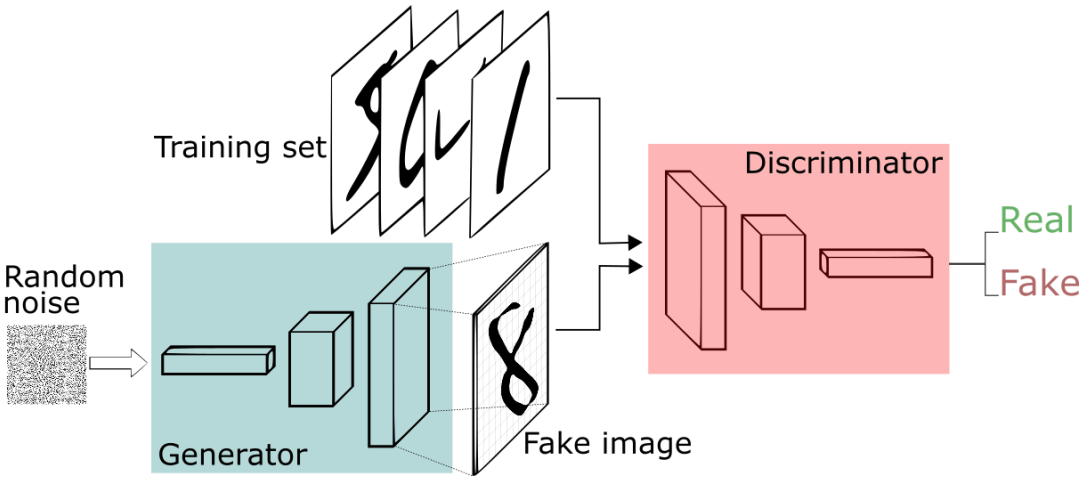


Рисунок 1 – Алгоритм роботи GAN

Розвиток машинного навчання та штучного інтелекту почав набирати обертів останнім часом, адже задачі конфлікту інтересів у реальному житті є складнішими, а умови та стратегії можуть змінюватися непередбачувано. Гравець часто мотивований не лише чистим егоїзмом, а й соціальними уподобаннями, як-от взаємність: гравець відчуває провину, коли вибирає невдалу дію, навіть якщо помітив хороший сигнал, тоді як інший гравець дратується, коли вибирає кооперативну дію, навіть якщо помітив поганий сигнал. Навіть більше, гравець часто стає наївним і вибирає дію у випадковий спосіб, незалежно від свого чистого егоїзму та взаємних мотивів, через свої когнітивні обмеження та невизначений психологічний настрій, а також стратегічну складність, спричинену моніторингом недосконалості та особистим спостереженням [3]. Тож забезпечення адаптивності та гнучкості поведінки ШІ є цілями в галузях, що потребують високої ефективності виконання роботи та мінімальних втрат. Теорія ігор, що почала застосовуватися вперше під час Другої світової війни для прорахунку дій супротивника, а потім знайшла застосування в розвитку ШІ, зараз знову може повернутися на «поле бою». У сучасних умовах саме поєднання ігрових моделей, машинного навчання та багатоагентних систем дає змогу розробляти більш ефек-

тивні стратегії й адаптивні алгоритми. Тому розвиток ШІ та методів машинного навчання стає одним із ключових пріоритетів, адже вони забезпечують новий рівень як для цивільних цифрових систем, так і для оборонних технологій.

Отже, у сучасному світі теорія ігор перестала бути суто математичною дисципліною і перетворилась на важливий інструмент для побудови складних моделей взаємодії – від економіки до комп'ютерних технологій і штучного інтелекту. Сьогодні, коли ШІ дедалі частіше бере участь у прийнятті рішень і у військових застосуваннях, методи теорії ігор знову набувають стратегічного значення. Саме тому подальший розвиток штучного інтелекту та моделей раціональної поведінки залишається одним із ключових напрямів сучасних досліджень.

Список використаних джерел

1. Штучний інтелект в іграх: коли краще робити його чесним, а коли дозволяти читерство. URL: https://gamedev.dou.ua/blogs/artificial-intelligence-and-cheating-in-games/ (дата звернення: 23.11.2025).

2. Game Theory in AI. URL: https://www.geeksforgeeks.org/machine-learning/game-theory-in-ai/ (дата звернення: 16.11.2025).

3. Matsushima H. Behavioral theory of repeated prisoner's dilemma: generous tit-for-tat strategy: working paper. CARF-F-452. Tokyo: University of Tokyo, 2019. 21 p.

4. Січко Т. В., Нескородєва Т. В. Методичні вказівки щодо виконання лабораторних робіт з дисципліни «Методи оптимізації та дослідження операцій» для студентів СО «Бакалавр» денної та заочної форм навчання спеціальностей 122 «Комп'ютерні науки», 113 «Прикладна математика». Вінниця: ДонНУ імені Василя Стуса, 2020.

5. Експертні та рекомендаційні системи: навч. посіб. / Т. В. Нескородєва, Є. Є. Федоров, Т. В. Січко, А. Р. Нескородєва. Вінниця: ДонНУ імені Василя Стуса, 2022. 208 с.

УДК 620.18:004.9

Григоренко Я. С., аспірант,
Дмитрієв Д. О., д-р техн. наук, професор,
професор кафедри автоматизації,
робототехніки і мехатроніки

ІНТЕЛЕКТУАЛЬНІ ТЕХНОЛОГІЇ ЯКОСТІ МАТЕРІАЛІВ В АДИТИВНОМУ ВИРОБНИЦТВІ МЕТАЛЕВИХ ВИРОБІВ

Херсонський національний технічний університет, м. Херсон

Вступ. Адитивне виготовлення металевих виробів перетворилося на одну з базових технологій сучасного високоточного машинобудування. Локалізоване введення енергії в шар порошкового матеріалу та пошарове нарощування геометрії деталі дають змогу створювати складні внутрішні канали, решітчасті структури

та топологічно оптимізовані форми, що важко або неможливо реалізувати традиційними методами. Водночас металеві процеси на основі лазерного або електронного променя характеризуються значною кількістю змінних параметрів і складною фізикою тепломасообміну. У роботі [1] досліджено інтелектуальні технології визначення характеристик якості металевих виробів, виготовлених методами пошарового утворення.

Актуальність. За даними оглядових робіт, присвячених оцінюванню адитивного виробництва [2; 3], найпоширенішими дефектами є газова та ключова пористість, зони неповного плавлення, мікротріщини, локальні перегріви й сильна анізотропія структури. Їх природа зумовлена не лише вибраним матеріалом, а й морфологією порошку, якістю укладання шару, стабільністю розплавленої ванни, стратегією сканування й режимами охолодження. Традиційні методи контролю – макро- і мікроструктурний аналіз вибіркових зразків, неруйнівні випробування лише наприкінці технологічного ланцюга – не дають змоги своєчасно виявляти ділянки підвищеного ризику та прогнозувати властивості деталей до завершення їх виготовлення, що є критичним для відповідальних машинобудівних елементів.

Аналіз останніх досліджень і публікацій. Адитивне виготовлення металевих виробів сьогодні розглядають як одну з ключових технологій високоточного машинобудування. Роботи [2; 3] показують, що завдяки локалізованому введенню енергії в шар порошкового матеріалу та пошаровому нарощуванню геометрії стає можливим створення топологічно оптимізованих деталей зі складними внутрішніми каналами й решітчастими структурами. Розвиток сенсорних систем, комп’ютерного зору та методів машинного навчання привів до появи концепції інтелектуального контролю якості, коли рішення приймаються на основі масивів даних, що надходять із різних рівнів процесу [4]. Паралельно розвивається напрям цифрових двійників, у межах якого для кожного виробу формується модель, здатна відтворювати теплові та механічні процеси в зоні побудови [5]. Поєднання цих підходів дає змогу перейти від постфактум-оцінювання до прогнозного керування якістю, однак на практиці відсутня цілісна система, яка б інтегрувала аналіз порошку, цифрове моделювання, in-situ-моніторинг та постпроцесну валідацію в єдиний інтелектуальний контур [1; 4; 5].

Мета – обґрунтувати доцільність інтеграції даних цифрового моделювання, сигналів процесного моніторингу та результатів мікроструктурного аналізу в єдину інтелектуальну систему контролю якості, орієнтовану на відповідальні машинобудівні деталі, та сформувати структурну схему такої системи [1].

Постановка задачі. Саме необхідність розроблення інтегрованої системи визначення характеристик якості металевих виробів у процесах пошарового утворення, яка б поєднувала аналіз порошку, цифрове моделювання, in-situ-моніторинг і постпроцесну валідацію, і становить основну наукову проблему даної роботи [1]. Для її розв’язання необхідно узагальнити основні напрями застосування машинного навчання, комп’ютерного зору та цифрових двійників у металевому адитивному виробництві та інтегрувати їх у єдину архітектуру інтелектуальної системи якості.

Виклад основного матеріалу. На рівні сировини критичними є розподіл розмірів частинок, форма та стан поверхні порошку. Надто широкий грануло-

метричний спектр або значна кількість деформованих частинок призводять до погіршення реології шару, утворення локальних зон пониженої щільності, що змінює поглинання енергії та режим плавлення [2; 3]. Застосування алгоритмів комп'ютерного зору та згорткових нейронних мереж для аналізу зображень порошкових розсипів дає змогу сегментувати частинки, класифікувати їх за формою, обчислювати показники сферичності, ексцентриситет та шорсткість поверхні, будувати інтегральні критерії придатності партії порошку й оцінювати пористість і міцність надрукованих зразків ще до початку друку [1; 4].

В установках LPBF застосовують комплекси оптичних і тепловізійних камер, фотодіодів та іноді акустичних сенсорів, на основі яких оцінюють геометрію melt-pool, розподіл температури, стабільність режиму плавлення та виявляють перегріви й зони неповного проплавлення [4]. Глибокі згорткові нейронні мережі, що працюють із термограмами, картами інтенсивності та часовими рядами інтенсивності випромінювання, забезпечують прогноз пористості й структурних аномалій із точністю, достатньою для практичного використання, а також розрізняють стабільний режим, режим зі схильністю до утворення газових пор та режим неповного плавлення й прогнозують майбутній стан melt-pool до формування дефектів [4].

Перший напрям пов'язаний із застосуванням цифрових двійників адитивних процесів, що розглядають як сукупність взаємопов'язаних термічних, механічних та мікроструктурних моделей, які відтворюють перебіг процесу побудови деталі в часі [5]. Такі моделі дають змогу оцінювати чутливість якості виробів до зміни параметрів друку, визначати зони потенційних перегрівів, накопичення напружень та зниженої щільності, формувати «віртуальне процесне вікно», у межах якого ймовірність виникнення критичних дефектів є мінімальною [5]. У роботі [1] цифровий двійник поєднується з моделями машинного навчання для швидкого прогнозування температурних полів, напружень та показників якості в режимі, наближеному до реального часу.

З урахуванням наведених підходів запропоновано інтегровану архітектуру інтелектуальної системи визначення характеристик якості металевих виробів, що охоплює всі етапи життєвого циклу адитивного виробу – від аналізу сировини до постпроцесної валідації [1]. У складі системи виділено модулі аналізу сировини, планування процесу та цифрового моделювання, in-situ-моніторингу й первинної аналітики, прогнозування дефектів і оцінювання властивостей, прийняття рішень та адаптивного керування, а також постпроцесної валідації [1]. Така система об'єднує дані про порошок, параметри процесу, результати in-situ – моніторингу, моделювання та постпроцесні вимірювання в єдиному інформаційному просторі, створюючи підґрунтя для самонавчальних адитивних виробничих комплексів і підвищення відтворюваності механічних властивостей виробів [1; 5].

Висновки. На основі аналізу сучасних публікацій сформовано цілісне уявлення про інтелектуальні технології визначення характеристик якості металевих виробів, виготовлених методами пошарового утворення [1–5]. Показано, що застосування машинного навчання та комп'ютерного зору на етапах аналізу порошку й in-situ-моніторингу melt-pool дає змогу автоматизувати виявлення аномалій процесу та оцінювати ризики формування дефектів. Цифрові двійники адитивних

процесів створюють можливість виконувати багатоваріантний аналіз і оптимізацію параметрів друку до початку експериментів, а їх інтеграція з ML-моделями формує основу замкненого контуру керування якістю. Запропонована модель інтелектуальної системи визначення характеристик якості охоплює всі етапи – від сировини до постпроцесної обробки – й дає змогу розглядати металеве адитивне виробництво як елемент кіберфізичної виробничої системи Індустрії 4.0 [1].

Список використаних джерел

1. Григоренко Я. С., Кузмічов М. Ф., Дмитрієв Д. О. Інтелектуальні технології визначення характеристик якості металевих виробів в процесах пошарового утворення. *Вісник Херсонського національного технічного університету.* 2025. № 4(95). С. 84–91.

2. Yeshiwas T. A., Tiruneh A. B., Sisay M. A. A review article on the assessment of additive manufacturing. *Journal of Materials Science: Materials in Engineering.* 2025. Vol. 20, № 85.

3. Gao B., Wu D., Wang Y. A review of research progress in selective laser melting (SLM). *Micromachines.* 2023. Vol. 14(1). P. 57.

4. Machine learning assisted quality control in metal additive manufacturing: A review / Z. Hu, X. Gao, Y. Li et al. *Advanced Powder Materials.* 2025. URL: https://www.sciencedirect.com/science/article/pii/S2772834X25000788 (дата звернення: 16.11.2025).

5. Ben Amor S., Brissaud D., Rivette M. Digital twin implementation in additive manufacturing: A comprehensive review. *Processes.* 2024. Vol. 12, № 6. P. 1062. URL: https://www.mdpi.com/2227-9717/12/6/1062 (дата звернення: 16.11.2025).

УДК 004.001

Зелінський О. О, здобувач 1 курсу
ОС «Магістр» ФЗ Комп'ютерні науки,
Січко Т. В., канд. техн. наук, доцент
кафедри інформаційних технологій

ІНТЕРПРЕТАЦІЇ РЕЗУЛЬТАТІВ ШТУЧНОГО ІНТЕЛЕКТУ ПІД ЧАС ІДЕНТИФІКАЦІЇ ЗОБРАЖЕНЬ

Донецький національний університет імені Василя Стуса, м. Вінниця

Ідентифікація зображень є одним із найбільш динамічних напрямів штучного інтелекту, у якому глибинні нейронні мережі демонструють здатність до високоточної класифікації та розпізнавання візуальних об'єктів [1]. Проте разом із ефективністю ці методи приносять і суттєву проблему: користувачі не можуть безпосередньо оцінити логіку прийняття рішення моделлю. Більшість сучасних архітектур – CNN, ResNet, Vision Transformers – працюють за принципом «чорної скриньки», коли рішення є точним, але його причини залишаються непрозорими [1; 4].

Така непрозорість створює нові виклики: помилки в ідентифікації можуть залишатися непоміченими, модель може навчитися на випадкових, а не релевантних ознаках, а в критично важливих сферах (медицина, безпека, промисловість) відсутність пояснень може становити прямий ризик. Саме тому зростає значення пояснюваних систем – Explainable Artificial Intelligence (XAI), що забезпечують інтерпретацію процесу прийняття рішень [5].

Глибинні моделі працюють із багатовимірними представленнями, тому їх внутрішні обчислення недоступні для прямої інтерпретації [5]. Результати класифікації можуть бути арифметично правильними, але водночас базуватися на нерелевантних патернах – наприклад, на фоновому шумі, кольорі зображення або сторонніх артефактах [1].

У реальних задачах це призводить до проблем:

– у медичних зображеннях модель може фокусуватися не на патології, а на текстових позначках на знімку [1];

– у системах відеоспостереження алгоритм іноді орієнтується на фон, а не на обличчя;

– у промислових системах контролю якості модель може «навчитися» реагувати на освітлення, а не на структуру поверхні.

Отже, інтерпретованість стає критично важливою, оскільки без неї неможливо оцінити коректність логіки моделі [5].

Методологія XAI включає низку підходів, які надають користувачам можливість отримувати докладні пояснення до кожного рішення моделі [5]. Серед основних методів:

1. Візуальні методи:

– Grad-CAM – будує теплові карти важливих областей зображення, підсвічуючи ті ділянки, що найбільше вплинули на класифікацію [2];

– CAM-варіації (Score-CAM, Guided Backpropagation, SmoothGrad) – забезпечують уточнені або згладжені візуальні пояснення [2];

– Attention Maps у Vision Transformer – дають змогу безпосередньо візуалізувати механізм уваги, що є природною особливістю трансформерів [4].

Ці методи дають змогу побачити, чи дійсно модель «дивиться» на необхідний об’єкт.

2. Локальні пояснення:

– LIME – створює локальні апроксимації навколо конкретного прогнозу, даючи змогу зрозуміти, які регіони зображення вплинули на рішення [3];

– SHAP – оцінює внесок кожної ознаки, побудований на ігровій теорії [3].

Ці методи допомагають пояснювати окремі рішення, а не модель загалом.

3. Глобальні пояснення:

– аналіз шарів нейронних мереж;

– побудова інтерпретованих сурогатних моделей;

– дослідження чутливості моделі до змін у даних [5].

Глобальні пояснення допомагають оцінити загальну логіку моделі та її надійність.

Попри значний розвиток XAI, існує низка проблем [5]:

– пост-хок-пояснення не гарантують істинності. Деякі методи лише наближають пояснення, не розкриваючи справжньої логіки моделі [3];

– нечіткість теплових карт. Інколи карти активностей не дають змоги однозначно визначити, на які саме області спиралася модель [2];

– висока обчислювальна складність. Деякі методи, наприклад, SHAP, потребують великої кількості обчислень [3];

– відсутність універсальних метрик оцінки якості пояснень. Сьогодні не існує загальноприйнятого способу виміряти, наскільки інтерпретація «правильна» [5].

Ці недоліки зумовлюють потребу у вдосконаленні інтерпретованих моделей та формуванні більш стандартизованих підходів.

Інтерпретація результатів не є лише теоретичною задачею – вона має прикладне значення у багатьох галузях [5]:

– медицина: пояснюваність дає змогу лікарю переконатися, що модель аналізує саме клінічно значущі області;

– промисловість: полегшує виявлення причин помилкових рішень у системах виявлення дефектів;

– транспорт: дає змогу аналізувати рішення систем автономного керування та виявляти помилкові тригери;

– безпека: пояснюваність допомагає уникнути помилкових спрацювань та знижує ризик упередженості;

– агросектор: дає змогу зрозуміти, які візуальні патерни використовуються під час діагностики стану рослин чи ґрунтів.

У кожній із цих сфер інтерпретація допомагає підвищити довіру до системи та мінімізувати вплив помилок.

Сучасні тенденції свідчать про рух у напрямі створення моделей, які є інтерпретованими за своєю природою, а не лише за допомогою пост-хок-методів [5]. До перспектив належать:

– розробка архітектур, що поєднують точність і прозорість;

– використання генеративних моделей для кращого аналізу важливих ознак [3];

– поєднання XAI із мультимодальними системами, де текстові пояснення можуть супроводжувати візуальні;

– автоматична діагностика упередженості моделей на основі пояснень;

– розвиток стандартизованих метрик для оцінювання якості інтерпретацій.

Отже, XAI перетворюється на самостійний напрям, що визначає майбутнє систем комп’ютерного зору.

Інтерпретація результатів штучного інтелекту під час ідентифікації зображень є фундаментальним складником побудови надійних, прозорих і безпечних систем комп’ютерного зору. Вона дає змогу не лише покращити якість моделей, а й забезпечити довіру користувачів, зрозуміти логіку роботи алгоритмів та уникнути критичних помилок. Розвиток методології Explainable AI сприятиме створенню моделей, здатних не лише правильно класифікувати об’єкти, а й аргументувати власні рішення. Це відкриває нові можливості для впровадження ШІ у відповідальні сфери та формує підґрунтя для подальших наукових досліджень.

Список використаних джерел

1. LeCun Y., Bengio Y., Hinton G. Deep learning. *Nature*. 2015. Vol. 521, № 7553. P. 436–444.

2. Grad-CAM: visual explanations from deep networks via gradient-based localization / R. R. Selvaraju, M. Cogswell, A. Das et al. *Proceedings of the IEEE International Conference on Computer Vision (ICCV).* 2017. P. 18–26.

3. Ribeiro M., Singh S., Guestrin C. Why should I trust you?: explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* 2016. P. 115–124.

4. An image is worth 16×16 words: transformers for image recognition at scale / A. Dosovitskiy, L. Beyer, A. Kolesnikov et al. *Advances in Neural Information Processing Systems.* 2020. Vol. 33. P. 58–68.

5. Кульчицька О. Ю., Січко Т. В. Цифрова обробка зображень та відео. *Прикладні інформаційні технології:* тези доповідей Всеукраїнської науково-практичної конференції (м. Вінниця, 29 квітня 2020 р.). Вінниця, 2020. С. 110–111.

УДК 004.8

Грибаніна А. О., студентка 4 курсу спеціальності 122 «Комп'ютерні науки»,
Ніколюк П. К., д-р фіз.-мат. наук, професор кафедри інформаційних технологій

МОДЕЛІ НЕЙРОННИХ МЕРЕЖ

Донецький національний університет імені Василя Стуса, м. Вінниця

Стрімкий розвиток технологій штучного інтелекту (ШІ) зумовлює зростання попиту на адаптивні моделі, здатні обробляти складні структури даних у режимі реального часу. У сучасних дослідженнях центральне місце займають нейронні мережі – обчислювальні системи, що успішно вирішують задачі класифікації, прогнозування, аналізу образів, керування автономними системами та моделювання складних процесів.

Нейронні мережі, на відміну від класичних алгоритмів, дають змогу виявляти приховані залежності, працювати з великими обсягами даних і формувати узагальнені представлення, що підвищує їх практичну ефективність. Мета цієї роботи – проаналізувати сучасні архітектури нейронних мереж (2017–2024) та продемонструвати апроксимацію функції в MATLAB як приклад їх застосування. Особливу увагу приділено автоенкодерам, згортковим мережам (CNN), рекурентним структурам (RNN) та трансформерам.

Сучасний етап розвитку нейронних мереж характеризується переходом від простих багатошарових моделей до архітектур, які здатні опрацьовувати просторові, часові та гібридні дані (рис. 1).

До ключових ознак сучасних ШНМ належать:

– ієрархічність структури, що дає змогу поетапно формувати ознаки;
– механізми уваги, які підсилюють релевантні елементи даних;
– паралельна обробка інформації, характерна для трансформерних моделей;

– ефективне стискання та реконструкція даних, реалізовані в автоенкодерах;
– стійкість до шумів та нелінійностей, важлива для технічних систем і БПЛА.

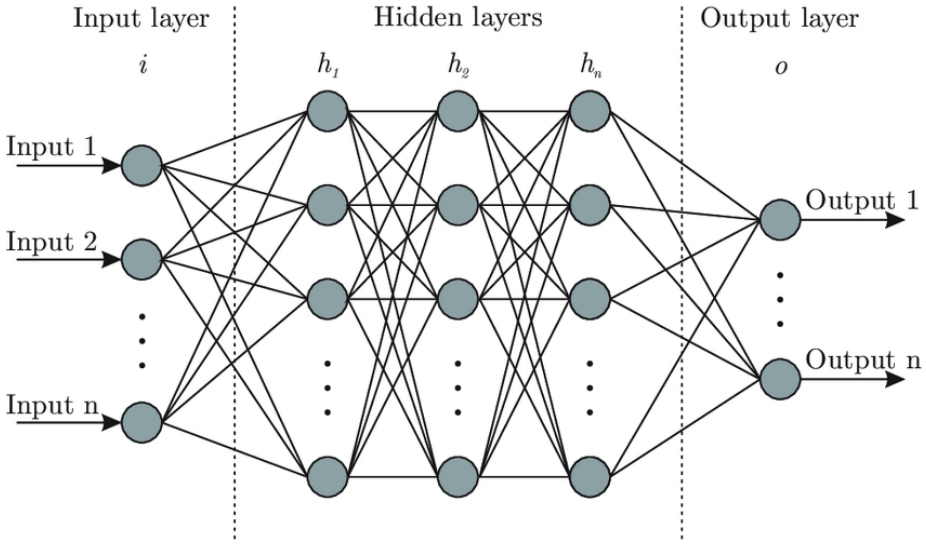


Рисунок 1 – Проста багатошарова модель.

Сучасні архітектури забезпечують глибоке семантичне розуміння даних, виходячи за межі класичної регресії чи класифікації.

Сучасні нейронні архітектури з високою науковою цінністю. Автоенкодери як засіб виявлення латентних структур. Автоенкодери застосовують для стискання та реконструкції даних, виявлення аномалій і підвищення якості зображень. Структура включає: енкодер (формує латентне представлення), декодер (відновлює дані) та латентний простір. Варіаційні автоенкодери (VAE) дають змогу моделювати розподіли даних і генерувати нові приклади.

Архітектури зі згортковими шарами (CNN). CNN ефективні для обробки просторових даних (ResNet, MobileNet, EfficientNet):

– розпізнавання та сегментація зображень;
– виявлення об’єктів на відео;
– оцінка стану поверхонь, дефектів та медичних аномалій;
– аналіз даних сенсорів у робототехніці та навігації БПЛА.

Перевага CNN – автоматичне виділення ознак без ручного проєктування фіч.

Трансформери та механізм самоуваги. Transformer із механізмом self-attention дає змогу:

– паралельну обробку великих даних;
– виявлення глобальних закономірностей;
– інтеграцію різних джерел інформації.

Застосування: Vision Transformer, GPT, BERT, класифікація аерофотознімків, автономне управління.

Гібридні нейромережеві системи. Поєднання CNN, LSTM та attention-механізмів дає змогу інтегрувати просторові та часові ознаки, підвищуючи точність прогнозування та адаптивність систем. Використовуються у робототехніці та автономних апаратах.

Експериментальне моделювання НМ у MATLAB. Щоб продемонструвати здатність нейронної мережі до навчання, було обрано завдання апроксимації заданої математичної функції. Функція для наближення: $y = \sin(x)$,

```
clc; clear; close all;

% Генерація вхідних даних та цільової функції (синус)
x = linspace(-2*pi, 2*pi, 200);
t = sin(x);

% Ініціалізація та конфігурація MLP з [10, 10] нейронами в шарах
net = feedforwardnet([10 10]);
net.trainParam.epochs = 500; % Кількість ітерацій навчання
net.trainParam.goal = 1e-5; % Цільова похибка

% Тренування моделі
net = train(net, x, t);

% Перевірка моделі (отримання виходу мережі)
y = net(x);

% Візуалізація результатів
figure;
plot(x, t, 'LineWidth', 2); hold on;
plot(x, y, '--', 'LineWidth', 2);
legend('Оригінальна функція', 'Наближення MLP');
title('Моделювання функції sin(x) за допомогою багатошарової мережі');
grid on;
```

Аналіз результатів. Розроблена модель продемонструвала високу точність наближення. Після 500 епох навчання середньоквадратична похибка знизилася до значення менше 10^{-5}. Графічне порівняння підтвердило високу точність моделі.

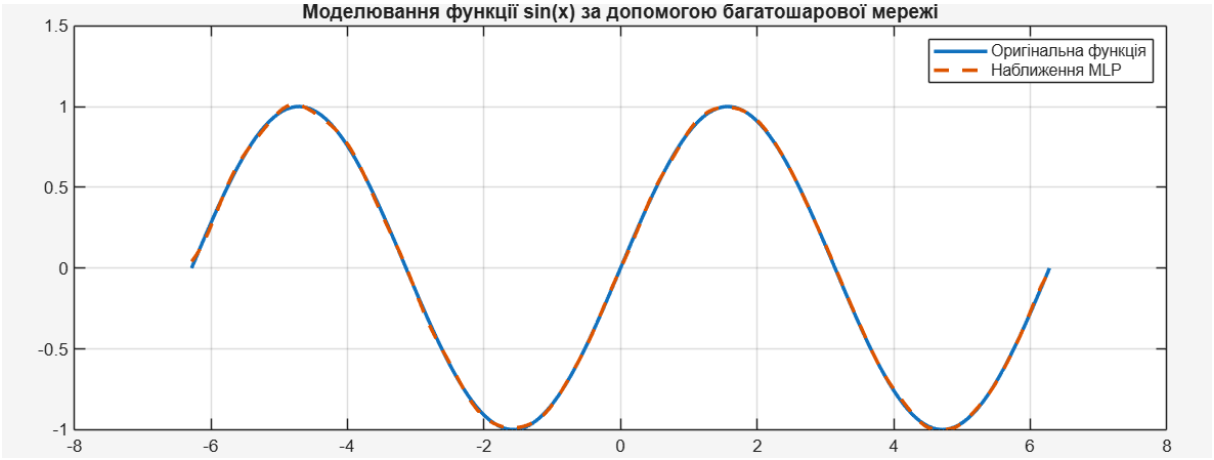


Рисунок 2 – Моделювання функції $\sin(x)$ за допомогою багатошарової мережі

Висновки. У роботі проведено поглиблений огляд сучасних архітектур нейронних мереж, що формують наукову основу сучасних систем штучного інтелекту. Особливу увагу приділено автоенкодерам, згортковим мережам, рекурентним структурам та трансформерам – моделям, що домінують у сучасних дослідженнях завдяки високій здатності до узагальнення та інтелектуального аналізу даних.

Експериментальне моделювання підтвердило здатність нейронних мереж ефективно апроксимувати складні функції, що підкреслює їх актуальність для задач прогнозування, керування та аналізу сигналів. Нейронні мережі залишаються ключовою технологією, яка визначає напрям розвитку ШІ та робототехніки.

Список використаних джерел

1. Kingma D., Welling M. Auto-encoding variational Bayes. 2014. URL: https://arxiv.org/abs/1312.6114 (дата звернення: 26.11.2025).

2. Deep residual learning for image recognition / K. He, X. Zhang, S. Ren, J. Sun. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 2016. P. 770–778.

3. Hochreiter S., Schmidhuber J. Long short-term memory. *Neural Computation.* 1997. Vol. 9, № 8. P. 1735–1780.

4. Attention is all you need / A. Vaswani, N. Shazeer, N. Parmar et al. *Advances in Neural Information Processing Systems.* 2017. Vol. 30. P. 5998–6008.

5. Baltrusaitis T., Ahuja C., Morency L. Multimodal machine learning: a survey and taxonomy. *IEEE Signal Processing Magazine.* 2019. Vol. 36, № 1. P. 27–45.

УДК 004.8:330.4

Русавський О. О., аспірант,
Роштейн О. П., д-р техн. наук, професор
кафедри інформаційних технологій

ПРОЄКТУВАННЯ НЕЧІТКИХ КОГНІТИВНИХ КАРТ МЕТОДОМ НАЙМЕНШОГО ВПЛИВУ

Донецький національний університет імені Василя Стуса, м. Вінниця

Вступ. Нечіткі когнітивні карти (НКК) [1; 2] є зручним інструментом для моделювання узагальнених динамічних систем, до яких належать промислове підприємство, людський організм, бойові дії та інші системи, що розглядаються як взаємопов'язана сукупність об'єктів і процесів, які змінюються з часом [3]. Важливим етапом побудови НКК є визначення сили впливу (w_{ij}) між концептами C_i та C_j, які відповідають змінним стану системи. Пряме призначення числових значень сил впливу w_{ij} пов'язане з великою часткою суб'єктивізму. Експерту важко дати чітку числову оцінку величини w_{ij}, але він легко користується якісними порівняннями, як-от: однакова, трохи більше (менше), набагато більше (менше) тощо. Парні порівняння реалізовані в методі ієрархії Сааті [4]. Складність цього методу полягає в необхідності побудови повної матриці парних порівнянь і перевірки її узгодженості.

Для зменшення кількості опитувань експертів і обчислень пропоновано скористатись методом «найгіршого випадку», який представлено у роботі [5].

Метод. Передбачається наявність експерта, який здатен:

– визначити множину концептів НКК;

– визначити множини позитивних і негативних впливів між концептами;

– в множинах позитивних і негативних впливів визначити найменші впливи;

– порівняти кожен із впливів з найменшим впливом за 9-бальною шкалою Сааті (табл. 1).

Зв'язки поділяються на дві групи: *позитивні* ($w_{ij} > 0$) та *негативні* ($w_{ij} < 0$).

У кожній групі експерт обирає зв'язок із *найменшою силою впливу* (w_{min}), якому присвоюється ранг r_{min}.

Усі інші зв'язки порівнюються з найменшим. Ранг r_i характеризує силу впливу так, що виконується співвідношення:

$$\frac{w_1}{r_1} = \frac{w_2}{r_2} = \cdots = \frac{w_{min}}{r_{min}} = \cdots = \frac{w_n}{r_n}.$$

Відношення рангів оцінюється за шкалою Сааті (табл. 1).

Таблиця 1 – Шкала порівнянь Сааті

r_i відносно r_{min}	$\frac{r_i}{r_{min}}$
Такий самий	1
Трохи сильніший	3
Сильніший	5
Значно сильніший	7
Абсолютно сильніший	9

З огляду на умову нормування $w_1 + w_2 + \cdots + w_n = 1$ сила найменшого впливу визначається на такою формулою:

$$w_{min} = \frac{1}{\frac{r_1}{r_{min}} + \frac{r_2}{r_{min}} + \cdots + \frac{r_n}{r_{min}}}. \quad (1)$$

Решта ваг визначаються через найменшу силу впливу:

$$w_1 = w_{min} \cdot \frac{r_1}{r_{min}}, w_2 = w_{min} \cdot \frac{r_2}{r_{min}}, \ldots, w_n = w_{min} \cdot \frac{r_n}{r_{min}}.$$

Отримані значення сил впливу нормалізуються шляхом ділення на найбільшу силу впливу.

Приклад. Розглянемо абстрактну НКК, що складається з 7 концептів ($C_1 \ldots C_7$), структура якої наведена на рис. 1.

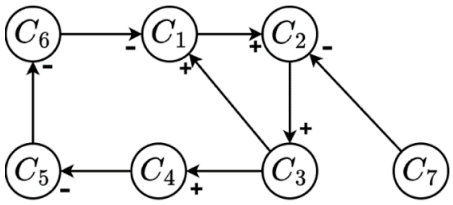


Рисунок 1 – Граф НКК

Ваги позитивних впливів визначаються множиною: $\{w_{1,2}, w_{2,3}, w_{3,1}, w_{3,4}\}$. Припускаємо, що вага найменшого позитивного впливу: $w_{1,2} = w(C_1 \to C_2)$.

Таблиця 2 – Порівняння позитивних впливів

Сила впливу	**Порівняння з найменшим впливом**	**Відношення рангу**
$w_{1,2}$	Такий самий	$r_{1,2}/r_{1,2} = 1$
$w_{2,3}$	Трохи сильніший	$r_{2,3}/r_{1,2} = 3$
$w_{3,1}$	Сильніший	$r_{3,1}/r_{1,2} = 5$
$w_{3,4}$	Значно сильніший	$r_{3,4}/r_{1,2} = 7$

Підставивши відношення рангів з табл. 2 у формулу (2), отримаємо:

$$w_{1,2} = \frac{1}{r_{1,2}/r_{1,2} + r_{2,3}/r_{1,2} + r_{3,1}/r_{1,2} + r_{3,4}/r_{1,2}} = \frac{1}{1+3+5+7} = \frac{1}{16}.$$

Решта сил впливу розраховується за формулами пропорції, наведеними в табл. 3. Для нормалізації значення діляться на максимум $(w_{3,4})$.

Таблиця 3 – Сили позитивних впливів

Сила впливу	$\boldsymbol{w_{1,2}}$	$\boldsymbol{w_{2,3}}$	$\boldsymbol{w_{3,4}}$	$\boldsymbol{w_{3,1}}$
Перед нормалізацією	$\frac{1}{16}$	$\frac{3}{16}$	$\frac{5}{16}$	$\frac{7}{16}$
Після нормалізації	$\frac{1}{7} \approx 0.14$	$\frac{3}{7} \approx 0.43$	$\frac{5}{7} \approx 0.71$	$\frac{7}{7} = 1$

Ваги негативних впливів визначаються множиною: $\{w_{4,5}, w_{5,6}, w_{6,1}, w_{7,2}\}$. Припускаємо, що вага найменшого позитивного впливу: $w_{4,5} = w(C_4 \to C_5)$.

Таблиця 4 – Порівняння негативних ваг

Сила впливу	**Порівняння з найменшим впливом**	**Відношення рангу**
$w_{4,5}$	Такий самий	$r_{4,5}/r_{4,5} = 1$
$w_{6,1}$	Трохи сильніший	$r_{6,1}/r_{4,5} = 3$
$w_{7,2}$	Трохи сильніший	$r_{7,2}/r_{4,5} = 5$
$w_{5,6}$	Сильніший	$r_{5,6}/r_{4,5} = 7$

Підставивши відношення рангів з табл. 4 у формулу (2), отримаємо:

$$w_{4,5} = \frac{1}{r_{4,5}/r_{4,5} + r_{6,1}/r_{4,5} + r_{7,2}/r_{4,5} + r_{5,6}/r_{4,5}} = \frac{1}{1+3+3+5} = \frac{1}{12}.$$

Решту сил впливу розраховують за формулами пропорції, наведеними в табл. 5. Для нормалізації значення ділять на максимум $(w_{5,6})$.

Таблиця 5 – Сили негативних впливів

Сила впливу	$w_{4,5}$	$w_{6,1}$	$w_{7,2}$	$w_{5,6}$
Перед нормалізацією	$\frac{1}{12}$	$\frac{3}{12}$	$\frac{3}{12}$	$\frac{5}{12}$
Після нормалізації	$\frac{1}{5} = 0.20$	$\frac{3}{5} = 0.60$	$\frac{3}{5} = 0.60$	$\frac{5}{5} = 1$

Результати табл. 3 та 5 зводяться у матрицю суміжності W, що наведена у табл. 6.

Таблиця 6 – Матриця ваг дуг НКК

	C_1	C_2	C_3	C_4	C_5	C_6	C_7
C_1	0	0.14	0	0	0	0	0
C_2	0	0	0.43	0	0	0	0
C_3	1.00	0	0	0.71	0	0	0
C_4	0	0	0	0	–0.20	0	0
C_5	0	0	0	0	0	–1.00	0
C_6	–0.60	0	0	0	0	0	0
C_7	0	–0.60	0	0	0	0	0

Висновки. Результатом роботи є метод найменшого впливу для оцінки ваг дуг нечіткої когнітивної карти. Сутність методу полягає у визначенні найменших ваг дуг серед позитивних і негативних впливів та порівнянні з найменшою вагою останніх ваг дуг за 9-бальною шкалою Сааті. Застосування методу забезпечує зменшення рівня суб'єктивізму експерта в порівнянні з прямим числовим оцінюванням ваг дуг, а також значно меншу кількість обчислень, порівняно з методом ієрархій Сааті.

Список використаних джерел

1. Kosko B. Fuzzy cognitive maps. *International Journal of Man-Machine Studies.* 1986. Vol. 24. P. 65–75.

2. Нечіткий сценарний аналіз узагальнених динамічних систем / О. П. Ротштейн, А. А. Кашканов, О. В. Зелінська, Д. І. Катєльніков. *Кібернетика і системний аналіз.* 2026. № 1.

3. Glushkov V. M. Introduction to ASU. Kiev: Technika, 1974.

4. Saaty T. L. Decision making with the analytic hierarchy process. *International Journal of Services Sciences.* 2008. Vol. 1, № 1. P. 83–98.

5. Fuzzy multicriteria selection of alternatives: the worst-case method / A. Rotshtein, E. Shnaider, M. Schneider, A. Kandel. *International Journal of Intelligent Systems.* 2010. Vol. 25. P. 948–957.

УДК 004.93

Бевзюк А. Ю., здобувачка 4 курсу
спеціальності 122 Комп'ютерні науки,
Ніколюк П. К., д-р фіз.-мат. наук, професор
кафедри інформаційних технологій

СТАБІЛІЗАЦІЯ ВІДЕОРЯДУ, ОТРИМАНОГО З ВІДЕОКАМЕРИ, ВСТАНОВЛЕНОЇ НА ДРОНІ

Донецький національний університет імені Василя Стуса, м. Вінниця

Стрімкий розвиток безпілотних літальних апаратів (БПЛА) привів до їх широкого використання в різних сферах: від кінематографії та журналістики до геодезії, сільського господарства, пошуково-рятувальних операцій і оборони. Ключовим компонентом, що забезпечує функціональність БПЛА, є відеокамера, яка надає оператору візуальний зв'язок із землею та дає змогу фіксувати події. Проте якість відеоряду, отриманого під час польоту, часто незадовільна через фізичні збурення, пов'язані з вібрацією двигуна, турбулентністю повітря та різкими змінами орієнтації під час маневрування [1]. Внаслідок цього відео стає розмитим, нестабільним і малопридатним для професійного аналізу чи комфортного перегляду, що знижує його інформативність.

Стабілізація відеоряду дронів є актуальною проблемою, оскільки у багатьох галузях якісний відеоконтент є критично важливим. У військовій сфері це забезпечує точність розвідданих, у цивільній – високу якість кінозйомки та журналістських матеріалів, а в науковій – достовірність даних моніторингу. Тому застосування методів стабілізації є необхідним етапом обробки, який підвищує точність, інформативність та придатність відео для подальшого комп'ютерного аналізу.

Існує кілька основних підходів до стабілізації відео, які застосовуються у системах комп'ютерного зору та відеозйомки з дронів:

– методи на основі оптичного потоку, які оцінюють поле зміщень між сусідніми кадрами та визначають глобальний зсув камери (методи Лукаса–Канаде, Хорна–Шенка, Farneback тощо) [2];

– методи, що використовують ключові точки, де визначаються характерні ознаки зображення (SIFT, ORB, SURF), встановлюються відповідності між ними та обчислюється матриця трансформації для вирівнювання кадрів;

– методи фільтрації траєкторії руху, коли оцінюється послідовність зміщень камери, після чого вона згладжується за допомогою фільтрів (Калмана, середнього ковзного, Gaussian smoothing), а зображення коригуються відповідно до отриманої стабілізованої траєкторії.

У цьому дослідженні застосовано підхід, що базується на оцінці міжкадрового руху за допомогою оптичного потоку Farneback. Отримані значення зміщення використовувалися для побудови афінного перетворення, яке компенсувало глобальний рух камери та забезпечувало стабілізацію відеоряду.

Для експериментальної перевірки методу було використано набір даних UAV123_10fps [3], що містить реальні відеопослідовності польоту дронів у різних

умовах. Реалізацію алгоритму виконано мовою C# з використанням бібліотеки OpenCvSharp, що надала необхідні засоби для роботи з оптичним потоком та геометричними перетвореннями.

Для реалізації стабілізації було завантажено 50 кадрів із набору даних та попередньо перетворено у формат градацій сірого. Між кожною парою сусідніх кадрів обчислювався оптичний потік за алгоритмом Farneback, який дає змогу визначити напрям і величину локальних зміщень у зображенні:

```
Mat flow = new Mat();
Cv2.CalcOpticalFlowFarneback(prevGray, gray, flow, 0.5, 3, 15, 3, 5, 1.2,
0);
```

На основі отриманої карти руху визначався середній вектор глобального зміщення, що характеризує загальний дрейф камери між кадрами. Цей вектор послідовно акумулювався, формуючи оцінку кумулятивного зміщення дронах.

Для компенсації цього руху застосовувалося афінне перетворення типу зсуву, що дає змогу змістити кадр у протилежному напрямку та вирівняти відеоряд. Зсув описувався матрицею:

```
Mat shiftMatrix = Mat.Eye(2, 3, MatType.CV_32F);
shiftMatrix.Set(0, 2, -accumulatedShift.X);
shiftMatrix.Set(1, 2, -accumulatedShift.Y);
```

Оскільки стабілізація відеоряду компенсує загальний дрейф камери, зміни між окремими кадрами можуть бути незначними та малопомітними на статичних зображеннях. Однак у динаміці, під час програвання відео, сумарний ефект вирівнювання руху стає чітко видимим.

На рис. 1 наведено приклад кадру до та після стабілізації.

Для кількісної оцінки ефективності алгоритму було використано метрику суми абсолютних різниць (SAD) між послідовними кадрами:

$$SAD = \sum_{x,y} |I_t(x,y) - I_{t-1}(x,y)|,$$

де $I_t(x,y)$ та $I_{t-1}(x,y)$ – це інтенсивності пікселів поточного та попереднього кадру.

На графіку (рис. 2) представлено динаміку змін цієї метрики для оригінального та стабілізованого відео. Як видно з результатів, значення SAD після стабілізації (зелена лінія) знаходяться суттєво нижче за значення оригінальної послідовності (червона лінія) на переважній більшості кадрів. Це свідчить про значне зменшення міжкадрових відмінностей, що безпосередньо підтверджує ефективність запропонованого методу стабілізації.

Проведений аналіз і реалізація алгоритму показали, що стабілізація на основі оптичного потоку дає змогу значно зменшити розмиття і коливання кадрів. Після обробки відео стає більш стабільним, інформативним та придатним для подальшого комп’ютерного аналізу. Це підвищує точність оцінки руху об’єктів, полегшує візуальне сприйняття та розширює можливості практичного використання даних дронів.



Рисунок 1 – Приклад зображення до та після стабілізації

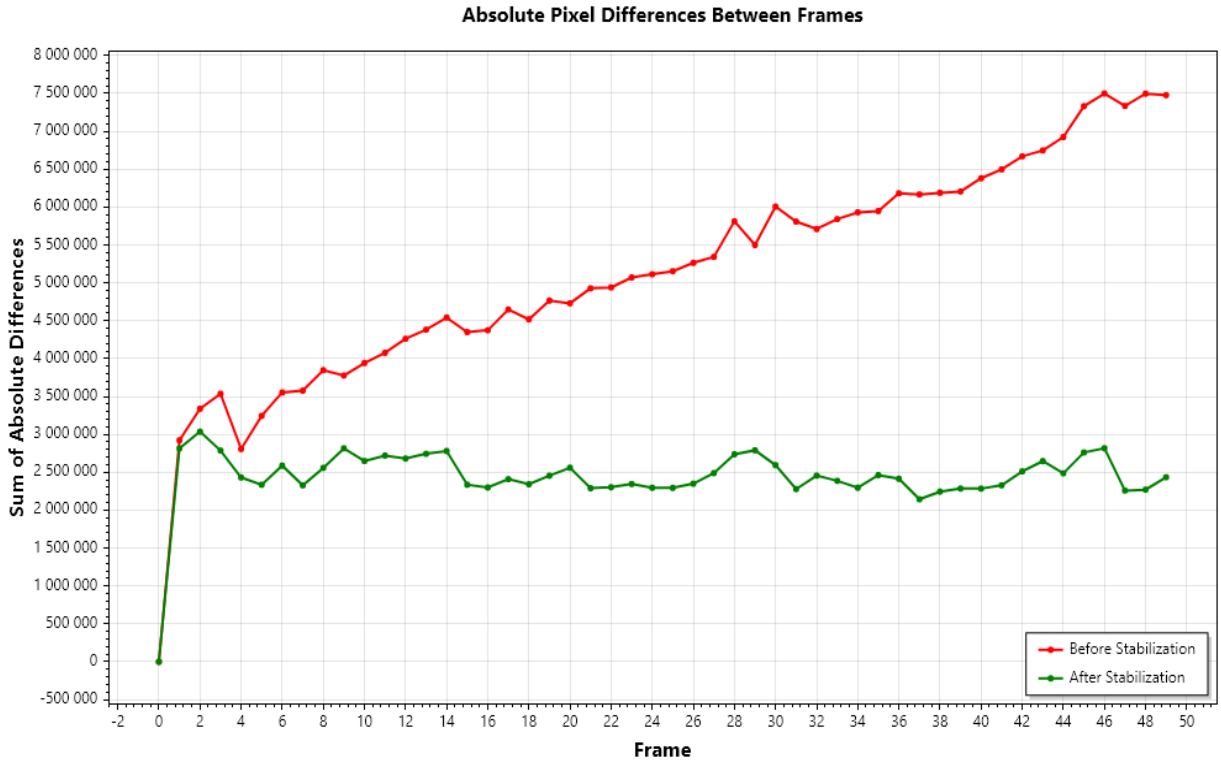


Рисунок 2 – Аналіз стабілізації за метрикою SAD

Список використаних джерел

1. Cross N. S. Onboard video stabilization for unmanned air vehicles. 2011. URL: https://doi.org/10.15368/theses.2011.67 (дата звернення: 27.11.2025).

2. Real-time optical flow-based video stabilization for unmanned aerial vehicles / A. Lim et al. Journal of real-time image processing. 2017. Т. 16, № 6. P. 1975–1985. URL: https://doi.org/10.1007/s11554-017-0699-y (дата звернення: 28.11.2025).

3. Mueller M., Smith N., Ghanem B. A benchmark and simulator for UAV tracking. *Computer vision – ECCV 2016.* Cham, 2016. P. 445–461. URL: https://doi.org/10.1007/978-3-319-46448-0_27 (дата звернення: 28.11.2025).

СЕКЦІЯ 3
ЕКСПЕРТНІ, РЕКОМЕНДАЦІЙНІ СИСТЕМИ ТА СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ

УДК 004.01

Левченко М. Р., здобувачка 4 курсу
спеціальності 122 Комп'ютерні науки,
Веселовська Н. Р., д-р техн. наук, професор,
професор кафедри інформаційних технологій

АДАПТАЦІЯ AGILE-МЕТОДОЛОГІЙ ДЛЯ РОЗРОБКИ БЕЗПЕЧНИХ ПРОГРАМНИХ ПРОДУКТІВ

Донецький національний університет імені Василя Стуса, м. Вінниця

Гнучкі підходи давно стали частиною роботи команд, які створюють програми. Вони дають можливість швидко реагувати на зміни, але водночас відкривають простір для помилок, пов'язаних з безпекою. Саме тому постає питання, як зробити гнучку розробку не тільки швидкою, а й захищеною [1].

Сьогодні будь-який софт може опинитися під загрозою через неправильну перевірку даних або невдало написаний фрагмент коду. Кількість атак зростає, і це робить питання безпеки особливо важливим. Гнучкі методи приваблюють тим, що їх легко адаптувати. Завдяки цьому виникає можливість не просто додати окремий етап перевірки, а вплести безпеку у сам процес створення [2].

У багатьох працях підкреслено, що успіх у безпеці залежить від звичок команди. Одні автори пишуть про необхідність постійних перевірок [2]. Інші наголошують на ролі взаємодії між учасниками і на тому, що кожен має розуміти типові ризики [3]. Є роботи, де пропонують не просто додавати перевірку, а й змінювати спосіб мислення так, щоб кожне рішення оцінювалося з погляду безпечності.

Мета цієї роботи полягає у тому, щоб розкрити, як гнучкий підхід можна природно поєднати з регулярною увагою до захисту, і показати власні ідеї, які можуть підсилити цей процес.

Необхідно визначити, які елементи гнучкої роботи можна покращити. Потрібно знайти практичні інструменти, які не ускладнюють роботу команди, але допомагають будувати більш надійний продукт [4; 5].

Під час розгляду питання стало зрозуміло, що гнучка розробка дає змогу робити малі, але регулярні кроки в напрямі безпеки. Одним із власних практичних пропозицій є введення короткого етапу перевірки перед початком кожної задачі [3]. Його суть у тому, щоб розробник перед стартом на хвилину замислився, які саме частини коду можуть бути ризикованими. Це простий прийом, але він переключає увагу на важливі моменти і зменшує кількість типових помилок. Інша ідея полягає у тому, щоб після завершення кожного фрагмента робити невеличкий журнал помилок та рішень [1]. Це не повинно мати вигляд складної документації. Достатньо кількох речень, що описують, що саме перевірялось і чому ця перевірка була важливою. Такий журнал працює як нагадування для самої команди і допомагає бачити повторювані моменти, що підлягають удосконаленню [4]. Також можна додавати до щоденних зустрічей короткий блок, де кожен учасник може назвати одну дрібницю, яка покращує безпеку [3]. Це може бути простий приклад, наприклад, уважніше перевіряти вхідні дані або не залишати тимчасові фрагменти коду. Такий підхід формує культуру, де безпека перестає

бути окремим завданням і стає природною частиною обговорень [2]. Ще одним практичним моментом є створення маленьких навчальних мінісесій усередині команди. Наприклад, один учасник може показати просту типову помилку і коротко пояснити, як її уникати. Це не займає багато часу, але сприяє тому, що команда росте у цьому напрямі разом [5].

Результати аналізу показали, що такі практичні кроки працюють найбільш ефективно тоді, коли повторюються регулярно. Маленькі, але постійні дії створюють відчутну різницю у якості коду і в стійкості продукту до можливих загроз [1].

Поєднання гнучких методів і безпеки можливе, якщо додавати до процесу маленькі, але регулярні практичні дії. Авторські ідеї у вигляді короткої підготовчої перевірки, створення журналу помилок, обговорення простих порад під час зустрічей і маленьких внутрішніх навчань допомагають сформувати культуру уважності. Такий підхід робить продукт надійнішим і не сповільнює роботу команди, що дає змогу зберегти переваги гнучкої методології.

Список використаних джерел

1. Williams L. Smith Br. Integrating Security Practices into Agile Development Processes. *IEEE Software.* 2023. URL: https://doi.org/10.1109/MS.2023.0145

2. Baca D., Boldt M. Security Training in Agile Software Development Teams. *Information and Software Technology.* 2024. URL: https://www.sciencedirect.com/science/article/pii/S0950584924001123 (дата звернення: 01.12.2025).

3. McGraw G. Software Security: Building Security. *Addison-Wesley Professional.* 2020. URL: https://www.informit.com/store/software-security-building-security-in-9780321356703 (дата звернення: 01.12.2025).

4. Alshammari B., Singh H. Secure Agile Development Framework for Modern Software Projects. *International Journal of Secure Software Engineering.* 2022. DOI: 10.4018/IJSSE.20220701.

5. Ghanbari H., Shahriari H. Embedding Security Activities into Agile Methods: A Systematic Review. *Journal of Systems and Software.* 2025. URL: https://www.journals.elsevier.com/journal-of-systems-and-software (дата звернення: 01.12.2025).

УДК 004.5

Афанасьєва Д. С., здобувачка 4 курсу спеціальності 122 Комп'ютерні науки,
Веселовська Н. Р., д-р техн. наук, професор, професор кафедри інформаційних технологій

АЛГОРИТМІЧНІ МЕТОДИ АДАПТАЦІЇ ВЕБКОНТЕНТУ ДЛЯ ЛЮДЕЙ З ВАДАМИ ЗОРУ

Донецький національний університет імені Василя Стуса, м. Вінниця

Зі зростанням складності вебінтерфейсів і перенесенням ключових життєвих процесів у цифровий простір питання доступності сайтів для людей з порушен-

нями зору перестає бути вузькоспеціальним. Стандарти на кшталт WCAG 2.2 задають чіткі вимоги до контенту [1], проте значна частина вебресурсів і надалі не відповідає навіть базовим критеріям: аналіз одного мільйона головних сторінок WebAIM у 2025 р. показав, що понад 94 % з них мають автоматично виявлені порушення WCAG із середнім значенням понад 50 помилок на сторінку [2]. Це означає, що покладатися лише на правильну розробку не можна – потрібні алгоритмічні рішення, які здатні аналізувати вже наявний вебконтент і адаптувати його під потреби користувача з вадами зору. Мета роботи – показати, як принципи доступності можуть бути реалізовані через алгоритмічні методи, що працюють у режимі реального часу: від евристичного аналізу DOM і CSS до використання моделей машинного навчання для персональної адаптації.

Для користувача з повноцінним зором поганий контраст чи дрібний шрифт – лише незручність. Для людини зі слабким зором це може бути фактичною відсутністю доступу до послуги: неможливістю прочитати інструкцію, заповнити форму, підтвердити дію. У таких умовах логічним кроком є перехід від ручного забезпечення доступності до алгоритмічних рішень. Ідеться не тільки про автоматизоване тестування, яке виявляє порушення стандартів, а й про адаптацію інтерфейсу в режимі реального часу: коли програмний модуль аналізує вже розміщену сторінку й автоматично підлаштовує її під потреби конкретного користувача з вадами зору.

Якщо розглядати адаптивну систему з практичного боку, її реалізація складається з кількох логічних шарів. На першому етапі здійснюється збір даних: модуль отримує доступ до DOM-дерева сторінки, аналізує атрибути елементів та їх стилі з урахуванням правил наслідування. Система ідентифікує текстові вузли, заголовки, посилання, кнопки, поля форм, зображення та інші значущі для користувача сутності. Далі формується аналітичний рівень, на якому обчислюється контрастність тексту і фону, аналізуються розміри шрифту, міжрядкові інтервали й відстані між блоками, перевіряється наявність альтернативного тексту в зображення, структура заголовків, доступність елементів з клавіатури та видимість фокуса, а також враховуються попередні користувацькі налаштування. На трансформаційному рівні система реалізує результати аналізу, динамічно змінюючи CSS-властивості, додаючи або коригуючи ARIA-атрибути, перерозподіляючи блоки та змінюючи порядок фокусування. Технічно це часто виконується через підключення додаткового стильового шару з вищим пріоритетом без втручання у вихідний код сайта. Завершальний рівень пов’язаний із взаємодією з користувачем і передбачає панель налаштувань, де задаються параметри шрифту, кольорової схеми, контрастності й анімації; ці значення зберігаються локально або в профілі користувача та надалі використовуються для персоналізованої адаптації.

Найпростішим практичним варіантом реалізації такої системи є правилоорієнтований підхід, у межах якого адаптація інтерфейсу здійснюється на основі заздалегідь визначених перевірок і трансформацій. Це виглядає як набір функцій, кожна з яких відповідає за конкретний аспект доступності: нормалізацію шрифтів, перевірку контрастності кольорів, підвищення видимості фокуса, корекцію структури заголовків або доступності елементів керування. Такі алгоритми не потребують складних обчислень, легко піддаються тестуванню та верифікації,

а їх результат є передбачуваним. У практичному застосуванні цього вже достатньо, щоб суттєво покращити зручність користування сайтами для людей із низьким зором, особливо на тих ресурсах, де вимоги WCAG взагалі не були враховані на етапі розробки.

Водночас правилоорієнтований підхід має природні обмеження, адже він працює переважно з формальними параметрами інтерфейсу і майже не враховує реальну поведінку користувача та контекст використання. Саме тому останнім часом поширення набувають гібридні методи, у яких базові правила поєднуються з аналізом даних [3]. Зібрані в процесі взаємодії агреговані поведінкові характеристики, як-от швидкість навігації, частота помилкових кліків, затримка фокуса на окремих елементах, частота зміни масштабу або режиму контрасту, дають змогу уточнювати початкові налаштування інтерфейсу. На основі цих даних система може поступово адаптувати базові алгоритми під індивідуальні потреби користувача, переходячи від універсальної адаптації до персоналізованої.

У реальному використанні це виглядає як поетапний процес. Під час першого відвідування сайт адаптується відповідно до стандартного набору правил, забезпечуючи мінімально необхідний рівень зручності: збільшений шрифт, підвищений контраст, чіткий фокус елементів керування. Паралельно система фіксує узагальнені дані про характер взаємодії користувача з інтерфейсом, не втручаючись у його роботу. Надалі, використовуючи ці спостереження, алгоритмічний модуль може коригувати первинні налаштування, наприклад, збільшувати базовий розмір тексту, зменшувати кількість одночасно відображених елементів або змінювати логіку відображення меню, роблячи її простішою для конкретного користувача.

Окремим напрямом розвитку є використання глибинного навчання для аналізу візуальної структури вебсторінок і складніших патернів взаємодії. У цьому випадку система працює не лише з числовими характеристиками стилів, а й із цілісним образом сторінки, отриманим, наприклад, зі скриншота. Нейронна мережа може виявляти проблемні зони з погляду сприйняття, як-от перевантажені інформацією блоки, дрібний текст або надмірно контрастні декоративні елементи, після чого формуються рекомендації щодо переформатування інтерфейсу [4]. Хоча повноцінна інтеграція таких моделей безпосередньо у браузер є технічно складною, на практиці можливе використання серверної обробки з передачею клієнту вже готових результатів адаптації, що є цілком прийнятним рішенням для навчальних і прикладних проєктів.

Отже, алгоритмічні методи адаптації вебконтенту для людей з порушеннями зору забезпечують перехід від формального декларування доступності до практичного аналізу та динамічної зміни інтерфейсу під конкретного користувача. На технічному рівні це реалізується через багатошарові програмні модулі, які аналізують структуру та стилі сторінки, застосовують правила WCAG 2.2 і за потреби доповнюють їх моделями машинного навчання для врахування поведінкових даних. Поєднання правилоорієнтованих і ML-підходів дає змогу створювати гібридні системи персоналізованої доступності, що можуть функціонувати як браузерні розширення, вбудовані модулі або інструменти для аудиту й покращення інтерфейсів, а з урахуванням посилення регуляторних вимог і стабільної

поширеності порушень доступності цей напрям набуває чітко вираженої практичної значущості.

Список використаних джерел

1. Web Content Accessibility Guidelines (WCAG) 2.2. *W3C Recommendation.* URL: https://www.w3.org/TR/WCAG22/ (дата звернення: 29.11.2025).

2. The 2025 Report on the Accessibility of the Top 1,000,000 Home Pages. *The WebAIM Million.* URL: https://webaim.org/projects/million/ (дата звернення: 30.11.2025).

3. Machine Learning Driven Adaptive Accessibility Interfaces for Enhanced User Experience / S. Patel, R. Mendoza, I. Cruz, N. Kim, P. Singh, A. Martinez. URL: https://doi.org/10.13140/RG.2.2.12545.36965 (дата звернення: 27.04.2026).

4. Deep Learning-Enhanced Adaptive Interface for Improved Accessibility in E-Government Platforms / X. Li; Y. Wang et al. *Proceedings of the 6th International Conference on Frontier Technologies of Information and Computer (ICFTIC 2024).* China, 2024. P. 1692–1695. URL: https://doi.org/10.1109/ICFTIC64248.2024.10912984 (дата звернення: 27.04.2026).

УДК 004.428.2

Дикий Д. С., здобувач 2 курсу
ОС «Магістр» спеціальності
F3 Комп'ютерні науки,
Потапова Н. А., канд. екон. наук, доцент,
доцент кафедри інформаційних технологій

ВЕБПОРТАЛ УПРАВЛІННЯ ЛОГІСТИКОЮ МАГАЗИНУ КОМПЛЕКТУЮЧИХ СМАРТСИСТЕМ

Донецький національний університет імені Василя Стуса, м. Вінниця

Сучасні інформаційні системи дедалі частіше будуються на основі принципів гнучкості, масштабованості та інтегрованості, оскільки традиційні програмні рішення вже не здатні ефективно задовольняти потреби бізнесу в умовах постійного зростання обсягів даних та високої варіативності користувацьких сценаріїв. Особливо це актуально для систем, що забезпечують підтримку логістичних процесів, де швидкість обробки інформації, стабільність роботи та можливість адаптації до змін є критично важливими.

Створення вебпорталу для автоматизації логістики магазину комплектуючих для смартсистем потребувало формування архітектури, яка б поєднувала високу продуктивність, простоту супроводу та можливість гнучкого масштабування. Важливо було забезпечити чітке структурування компонентів, стабільність обміну даними, а також здатність до інтеграції зі сторонніми сервісами та інструментами.

Архітектура розробленого вебпорталу ґрунтується на принципах клієнт-серверної взаємодії та патерні Model-View-Controller (MVC), що дає змогу чітко

розмежувати рівні даних, бізнес-логіки та інтерфейсу користувача. Такий підхід суттєво підвищує керованість системи, спрощує тестування і забезпечує можливість швидкої модернізації окремих її компонентів. На рис. 1 зображено загальний підхід до MVC-моделі.

Серверна частина порталу реалізована на Java з використанням фреймворка Spring, який надає розвинені засоби інверсії керування, гнучку конфігурацію та ефективну роботу з HTTP-запитами. Керування залежностями здійснюється за допомогою Gradle, що пришвидшує процес складання та полегшує інтеграцію сторонніх модулів. Для роботи з даними застосовано ORM-технологію Hibernate, яка забезпечує абстракцію над реляційною моделлю та мінімізує кількість ручних SQL-операцій.

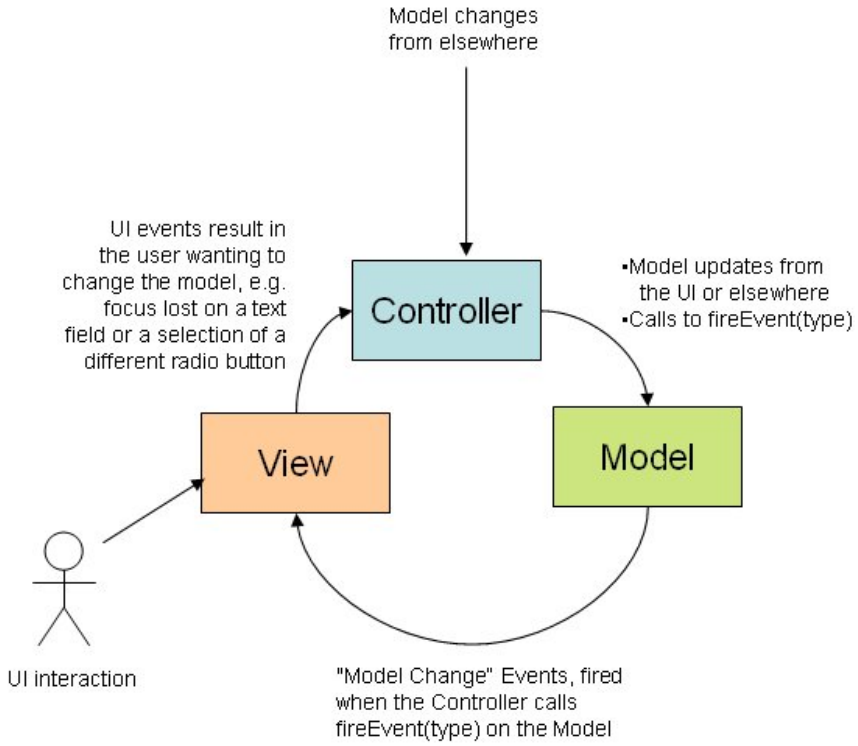


Рисунок 1 – Загальний підхід до побудови MVC-моделі

Контроль версій реалізовано через Liquibase, що гарантує узгодженість змін між середовищами розробки, тестування й продакшену. Основою для зберігання даних обрано MySQL як продуктивну та масштабовану СУБД. На рис. 2 зображено загальну архітектуру системи з поєднанням усіх її компонентів у єдине ціле.

Клієнтська частина побудована з використанням HTML, CSS та JavaScript, що дає змогу створити адаптивний та інтуїтивно зрозумілий інтерфейс. Така фронтенд-архітектура забезпечує зручну взаємодію користувача із системою, швидку обробку подій та мінімальні затримки під час навігації вебпорталом. Для підвищення продуктивності реалізовано кешування даних за допомогою Caffeine Cache, що значно скорочує час відповіді сервера та зменшує навантаження на базу даних.

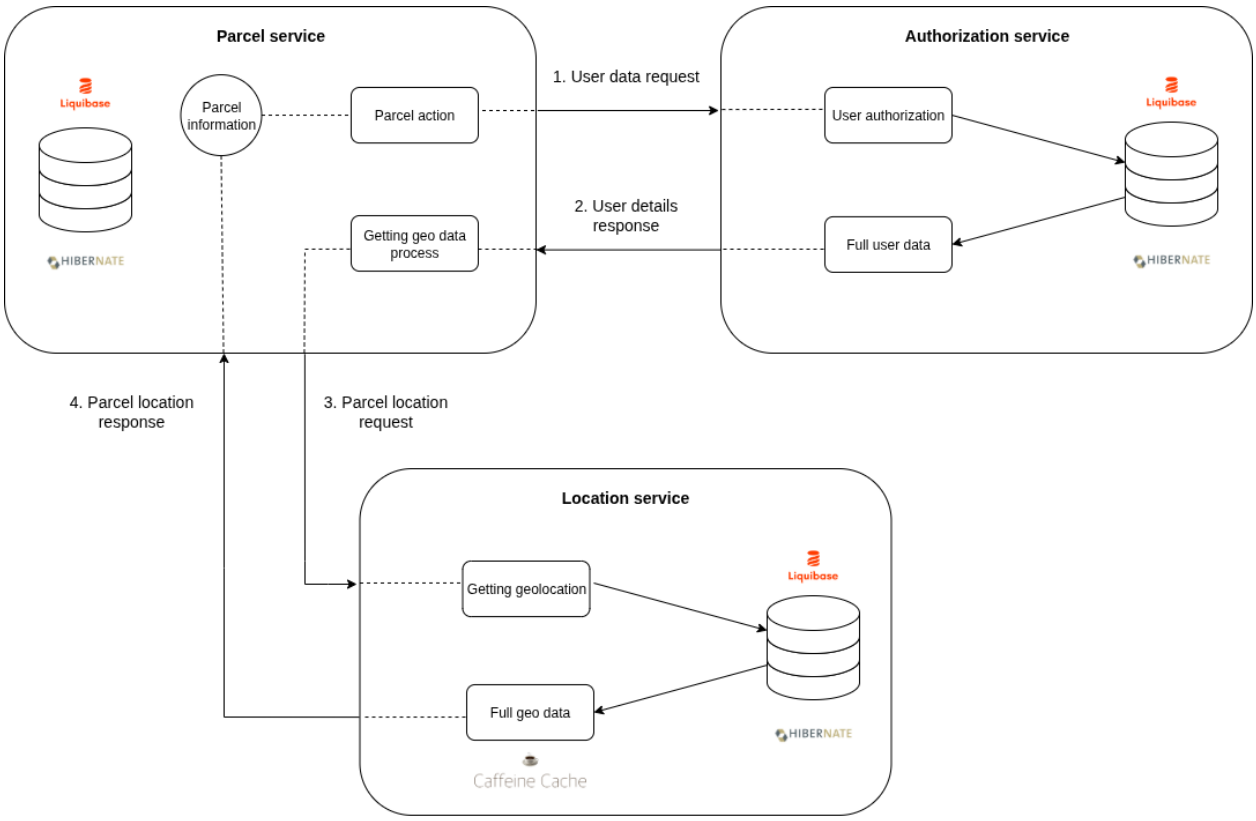


Рисунок 2 – Загальна архітектура системи

Опрацювання операцій із посилками в межах сервісу охоплює повний цикл взаємодії з логістичними даними – від формування нового відправлення до його моніторингу, редагування параметрів та можливого скасування. Система забезпечує узгоджену обробку всіх етапів руху посилок, що є критично важливим для підтримання безперебійної логістики компонентів смартсистем. Для наочного подання структури та взаємодій між основними елементами сервісу на рис. 3 представлено UML-діаграму, яка демонструє логіку виконання дій з посилками та описи відповідних операцій.

Реалізація серверної частини на базі Java та Spring, використання Hibernate і Liquibase для роботи з даними, а також інтеграція MySQL як основної СУБД, дали змогу сформувати стабільну платформу для обробки та зберігання інформації. Застосування кешування Caffeine забезпечило швидший доступ до даних і зниження навантаження на систему.

Клієнтська частина, розроблена з використанням HTML, CSS та JavaScript, забезпечила інтуїтивно зрозумілий та зручний для користувача інтерфейс, що суттєво підвищує ефективність роботи з системою. Модель обробки дій із посилками охоплює повний цикл логістичних операцій, забезпечуючи коректність та узгодженість усіх етапів переміщення товарів.

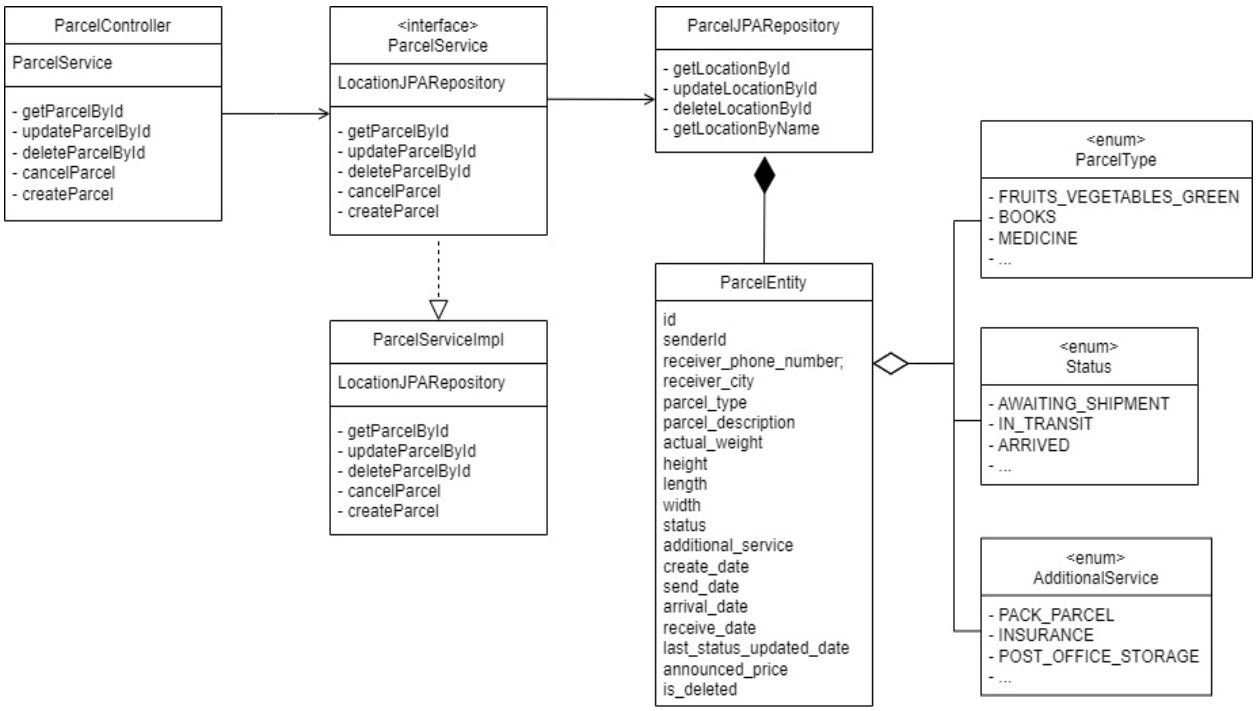


Рисунок 3 – Логіка виконання дій з посилками

Отже, за результатами проведеного аналізу та розроблення вебпорталу для автоматизації логістики компонентів смартсистем було створено функціональну, масштабовану та технологічно сучасну інформаційну систему. Запропонована архітектура, заснована на принципах клієнт-серверної взаємодії та патерні MVC, довела свою ефективність у контексті забезпечення чіткого розподілу відповідальностей, підвищення продуктивності та спрощення супроводу програмного забезпечення.

Список використаних джерел

1. Freeman E., Robson E. Head First Design Patterns. 2nd edition. O'Reilly, 2020. P. 312–333.

2. MySQL Indexes. URL: https://www.geeksforgeeks.org/mysql/mysql-indexes/ (дата звернення: 01.12.2025).

3. MVC. URL: https://developer.mozilla.org/en-US/docs/Glossary/MVC (дата звернення: 01.12.2025).

4. Guide to Caching in Java. URL: https://www.baeldung.com/caching-series (дата звернення: 01.12.2025).

5. Як будувати UML-діаграми. URL: https://dou.ua/forums/topic/40575/ (дата звернення: 01.12.2025).

УДК: 004.05

Явгусішин Б. А., здобувач 2 курсу
ОС «Магістр» спеціальності
F3 Комп'ютерні науки,
Антонов Ю. С., канд. фіз.-мат. наук, доцент,
доцент кафедри інформаційних технологій

ІНТЕЛЕКТУАЛЬНА РЕКОМЕНДАЦІЙНА СИСТЕМА ДІАГНОСТИКИ ПРОГРАМНИХ НЕСПРАВНОСТЕЙ ПЕРСОНАЛЬНИХ КОМП'ЮТЕРІВ: АРХІТЕКТУРА, АЛГОРИТМИ ТА РЕАЛІЗАЦІЯ

Донецький національний університет імені Василя Стуса, м. Вінниця

Вступ. Сучасні персональні комп'ютери є багаторівневими системами, що поєднують апаратні компоненти, операційну систему та програмне забезпечення. Зростання складності цих систем та зниження середнього рівня технічної знань користувачів призвело до збільшення кількості програмних несправностей, які складно ідентифікувати без спеціалізованих інструментів. Діагностика таких несправностей ускладнюється неоднозначністю симптомів, різноманітністю конфігурацій обладнання та наявністю «шуму» в даних, які можуть містити інформацію щодо несправностей.

У зв'язку з цим зростає потреба в інтелектуальних системах підтримки діагностики, здатних поєднувати експертні знання, аналіз технічних параметрів, машинне навчання та підхід обробки природної мови. Розроблена рекомендаційна система спрямована на подолання класичних обмежень діагностики, забезпечення пояснювальності результатів та можливості самостійного усунення несправностей без залучення технічних спеціалістів. Її архітектура та алгоритми створені так, щоб забезпечити гнучкість, масштабованість і точність роботи за умов неповних або суперечливих вхідних даних.

Основний текст. Для збору інформації, необхідної для діагностки, реалізована рекомендаційна система використовує два незалежні джерела. Першим є адаптивний опитувальник, де запитання формуються залежно від попередніх відповідей користувача. Такий підхід дає змогу врахувати спостережувану поведінку комп'ютера, включно з проявами, які не можуть бути автоматично зафіксовані системою. Другим джерелом є автоматичний технічний аналіз, що охоплює зчитування параметрів операційної системи, стану служб, системних журналів, температурних показників та інших характеристик, які є потенційно релевантними до несправності.

Отримані дані інтерпретуються у вхідний вектор симптомів, де кожен симптом представлено уніфікованою та структурованою ознакою, яка визначена в експертній базі знань. Це допомагає перетворити неструктуровані тексти журналів подій і суб'єктивні відповіді користувача на формалізований набір діагностичних параметрів. Вектор симптомів далі передається на опрацювання трирівневим діагностичним алгоритмом, що інтегрує експертні правила, ймовірнісні методи та машинне навчання.

Перший рівень діагностики є правило-орієнтованим, що спирається на набір залежностей між симптомами та потенційними несправностями. Кожне правило описується набором умов, вагових коефіцієнтів та несправністю, пов'язаною з ним. Алгоритм визначає ступінь відповідності правила наявним симптомам і формує перелік кандидатів несправностей. Цей рівень забезпечує високу інтерпретованість та швидку реакцію на типові та добре описані сценарії несправностей.

Другим рівнем є ймовірнісні методи, що уточнюють результати попереднього етапу на основі апріорних та умовних ймовірностей. Для кожної потенційної несправності обчислюється постеріорна оцінка, яка відображає узгодженість отриманого набору симптомів зі статистичними характеристиками несправностей. Система враховує як наявні симптоми, так і відсутні, якщо вони є типовими для певної несправності. Завдяки цьому цей рівень демонструє стійкість до часткових, шумових або суперечливих даних.

Третій і останній рівень – це машинне навчання. Він використовує накопичені історичні дані про попередні діагностичні сесії, що дає змогу виявляти складні патерни у взаємозв'язку симптомів і реальних причин несправностей. Модель ML.NET тренується на сервері за даними користувачів, які надали зворотний зв'язок про корисність наданих рекомендацій. Після навчання модель експортується та застосовується локально на пристроях користувачів. Результат машинного аналізу комбінується з оцінками двох попередніх рівнів, створюючи інтегровану діагностичну оцінку.

Для підвищення зрозумілості результатів реалізовано адаптер для роботи з великими мовними моделями. Він формує пояснювальні текстові інструкції на основі структурованих діагностичних даних і за бажанням користувача може проводити додатковий незалежний аналіз симптомів. Мовна модель не впливає на основний результат діагностики, але виконує роль інформаційного посередника, що забезпечує адаптивність системи до користувачів без спеціалізованих знань.

Під час проєктування системи особлива увага була приділена організації програмної архітектури, здатної забезпечити масштабованість, розширюваність та чітке розмежування відповідальностей. Для цього застосовано багаторівневий підхід до формування архітектури у поєднанні з патернами MVVM, Strategy, Adapter і Dependency Injection. Це забезпечило чітке розділення відповідальностей між інтерфейсом, бізнес-логікою та роботою з даними. Реалізація клієнтської частини створена за допомогою фреймворків .NET MAUI, що забезпечує кросплатформність і єдину структуру застосунку на різних операційних системах.

Для зберігання інформації використано базу даних PostgreSQL, розгорнуту у середовищі Docker, що дає змогу зробити симуляцію підключення застосунку до бази даних на сервері та створити дві логічно розділені бази даних: експертну та користувацьку. Експертна база даних містить несправності, симптоми, правила, ймовірності та рекомендації, тоді як користувацька зберігає історію діагностичних сесій, дані технічного аналізу, відповіді користувача та результати виконання алгоритму діагностики.

Висновки. Інтелектуальна рекомендаційна система, реалізована за допомогою поєднання адаптивного опитувальника, автоматичного технічного аналізу та трихрівневого алгоритму діагностики забезпечує високу якість визначення

програмних несправностей персонального комп’ютера навіть за умов неповних або суперечливих даних. Її багаторівнева архітектура дає змогу інтегрувати експертні правила, статистичні оцінки та машинне навчання у єдиний діагностичний процес, що підвищує точність і стійкість системи. Застосування мовних моделей забезпечує зрозумілість і доступність рекомендацій для користувачів з будь-яким рівнем технічних знань.

Список використаних джерел

1. Nield T. Essential Math for Data Science: Take Control of Your Data with Fundamental Linear Algebra, Probability, and Statistics. 1st ed. Sebastopol: O’Reilly Media, 2022. 349 p.

2. Introduction to Algorithms / Th. H. Cormen, Ch. E. Leiserson, R. L. Rivest, C. Stein. 4th ed. Cambridge: The MIT Press, 2022. 1312 p.

3. Silberschatz A., Galvin P. B., Gagne G. Operating System Concepts. 10th ed. Hoboken: Wiley, 2021. 1040 p.

4. Ryan D. Expert Systems: Design, Applications and Technology. New York: Nova Science Publishers, 2017. 138 p.

5. Capellman J. Hands-On Machine Learning with ML.NET: Getting Started with Microsoft ML.NET to Implement Popular Machine Learning Algorithms in C#. Birmingham: Packt Publishing, 2020. 451 p.

УДК 004.9

Рижков О. О., здобувач 2 курсу
ОС «Магістр» спеціальності
F3 Комп’ютерні науки,
Потапова Н. А., канд. екон. наук, доцент,
доцент кафедри інформаційних технологій

ІНТЕЛЕКТУАЛЬНА СИСТЕМА ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ ОРЕНДИ НЕРУХОМОСТІ

Донецький національний університет імені Василя Стуса, м. Вінниця

Сучасний світ бізнесу стрімко трансформується під впливом гібридних моделей роботи та цифровізації корпоративних процесів. Ефективне управління офісними просторами стало критичною необхідністю для організацій, які прагнуть оптимізувати використання ресурсів та забезпечити комфортні умови праці. Традиційні системи бронювання вимагають від користувачів складного процесу заповнення форм, що займає 15–20 хвилин та часто призводить до помилок. У цьому контексті розробка інтелектуальних систем з природномовним інтерфейсом набуває стратегічної значущості.

З появою великих мовних моделей (Large Language Models, LLM) та технологій обробки природної мови (Natural Language Processing, NLP) з’явилися нові

можливості для створення інтуїтивних систем управління бронюваннями. Інтеграція штучного інтелекту дає змогу користувачам формулювати запити природною мовою, наприклад: «Забронюй переговорну на 10 осіб на завтра з 14:00 до 16:00» або «Мені потрібен тихий кабінет для роботи щопонеділка з 9:00 до 12:00 протягом місяця». Це принципово змінює користувацький досвід та підвищує продуктивність співробітників.

Ця робота присвячена розробці повнофункціональної інтелектуальної системи бронювання офісних приміщень з інтегрованим AI-асистентом на основі моделі GPT-4o-mini від OpenAI. Вибір цієї моделі обґрунтовується оптимальним співвідношенням якості розпізнавання природномовних запитів (точність >90 %), швидкості відгуку (до 2 секунд) та економічності API-викликів. Система забезпечує автоматичне розпізнавання параметрів бронювання через технології Named Entity Recognition (NER), визначення намірів користувача через Intent Detection та екстракцію параметрів через Slot Filling.

Ключовим науковим результатом є розробка комплексних алгоритмів виявлення конфліктів бронювань для одноразових, багатоденних та рекурентних резервацій. Алгоритм підтримує складні рекурентні паттерни (щоденні, щотижневі, щомісячні, за вибраними днями тижня) та забезпечує перевірку доступності офісних приміщень у режимі реального часу. Математична умова перетину часових інтервалів (start1 < end2 AND start2 < end1) реалізована з оптимізацією через B-tree індексування бази даних, що знижує складність з $O(N)$ до $O(logN + K)$. Гібридний підхід до зберігання рекурентних бронювань забезпечує компактність даних (збереження правила замість всіх входжень) та можливість централізованого редагування серії бронювань.

Архітектура платформи спроєктована з використанням сучасного патерну Feature-Sliced Design, що забезпечує модульність, масштабованість та підтримуваність системи. Технологічний стек включає Next.js 16 з Server Actions та React Server Components для серверного рендерингу, TypeScript для типобезпеки, PostgreSQL 15 для зберігання даних, Prisma ORM для type-safe доступу до бази, Vercel AI SDK для streaming-відповідей від AI-асистента та Tailwind CSS для адаптивного інтерфейсу. Інтеграція з OpenAI API забезпечує обробку природномовних запитів через структурований вихід у форматі JSON, що дає змогу уникнути етапу парсингу текстових відповідей.

Проведено комплексне тестування розробленої системи за метриками продуктивності (час відгуку AI-асистента 1.8 с, перевірки доступності 450 мс), точності AI (екстракція дат 96 %, часу 94 %, місткості 91 %), якості планування (0 % конфліктів, 0 % пропущених перетинів) та користувацького досвіду (85 % успішних завершень з першої спроби). Система забезпечує скорочення часу на оформлення бронювання на 70–85 %, порівняно з традиційними формами, підтримує обробку 100+ одночасних користувачів та понад 10,000 бронювань зі збереженням високої продуктивності.

Подальший розвиток системи може включати інтеграцію з корпоративними календарями (Google Calendar, Microsoft Outlook), системами контролю доступу, платформами відеоконференцій (Zoom, Teams), впровадження аналітики використання офісних просторів з machine learning для прогнозування попиту, мульти-

мовну підтримку та голосовий інтерфейс для взаємодії з AI-асистентом. Результати дослідження демонструють перспективність інтеграції великих мовних моделей у корпоративні системи управління ресурсами та відкривають нові можливості для підвищення продуктивності праці через природномовні інтерфейси.

Список використаних джерел

1. Attention is All You Need / A. Vaswani, N. Shazeer, N. Parmar et al. *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS 2017).* Long Beach, California, USA, 2017. P. 6000–6010.

2. Language Models are Few-Shot Learners / T. Brown, B. Mann, N. Ryder et al. *Advances in Neural Information Processing Systems.* Vol. 33 (NeurIPS 2020). 2020. URL: https://papers.nips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html (дата звернення: 21.11.2025).

3. Emergent Abilities of Large Language Models / J. Wei, Y. Tay, R. Bommasani et al. *Transactions on Machine Learning Research.* 2022. URL: https://openreview.net/forum?id=yzkSU5zdwD (дата звернення: 21.11.2025).

4. OpenAI GPT-4o-mini: advancing cost-efficient intelligence. URL: https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/ (дата звернення дата звернення: 21.11.2025).

5. Scheduling Algorithms: A Comprehensive Guide to Resource Allocation and Time Management. URL: https://www.geeksforgeeks.org/scheduling-algorithms/ (дата звернення дата звернення: 21.11.2025).

УДК 37.091.2:004.9

Олійник К. А., здобувач 1 курсу ОС «Магістр»
ФЗ Комп'ютерні науки,
Ніколюк П. К., д-р фіз.-мат. наук, професор
кафедри інформаційних технологій

ІНФОРМАЦІЙНА СИСТЕМА ДЛЯ АНАЛІЗУ ТА ПРОГНОЗУВАННЯ РОЗВИТКУ СТРУКТУРИ УНІВЕРСИТЕТІВ УКРАЇНИ

Донецький національний університет імені Василя Стуса, м. Вінниця

У сучасних умовах система вищої освіти України стикається зі скороченням контингенту студентів, обмеженим фінансуванням, нестачею кадрів та нерівномірним розвитком університетів. За таких обставин традиційні методи аналізу вже не забезпечують достатньої точності та швидкості прийняття управлінських рішень. Водночас зростає потреба у цифрових інструментах, які здатні забезпечити оперативне опрацювання освітніх даних, підтримку управлінських рішень та підвищення прозорості процесів у сфері вищої освіти. Саме тому впровадження сучасних аналітичних систем стає критичним для ефективного функціонування університетської мережі.

Метою роботи є розробка інформаційної системи, яка дає змогу автоматизувати процес збору, опрацювання та візуалізації даних про структуру університетів України, а також надає можливість формування прогнозних оцінок розвитку на основі доступних статистичних показників.

Запропонований підхід забезпечує зменшення ручної обробки, уніфікацію даних та створення аналітичного середовища для підтримки стратегічного планування.

Під час дослідження було використано декілька груп джерел даних, що забезпечують комплексний огляд стану та тенденцій у вищій освіті:

1. Єдина державна електронна база з питань освіти (ЄДЕБО) [1]: контингент студентів, вступ, випуск, ліцензовані спеціальності.

2. World Bank Open Data [2]: демографічна динаміка, економічні індикатори, дані щодо участі молоді в освіті.

Система, розроблена у межах роботи, включає модулі збору даних, їх попередньої обробки, інтеграції з різних джерел, а також аналітичний модуль, який забезпечує побудову графіків, порівнянь, динамічних звітів та виявлення ключових тенденцій у структурі вищої освіти України. Особливу увагу приділено створенню єдиного інтерактивного інтерфейсу, який дає змогу працювати з даними без необхідності використання сторонніх інструментів чи трудомісткого ручного опрацювання. На рис. 1 відображено діаграму компонентів.



Рисунок 1 – Діаграма компонентів

Практична цінність розробленої платформи полягає у можливості її застосування для оперативного аналізу, порівняння університетів за ключовими показниками та підготовки управлінських рішень.

За результатами дослідження отримано:

1. Розроблено повноцінну архітектуру інформаційної системи, що включає модулі збору, обробки та уніфікації даних, аналітичний модуль та інтерфейс візуалізації. Реалізована структура забезпечує інтеграцію різних джерел даних та їх подальшу обробку у єдиному середовищі, зображеному на рис. 1.

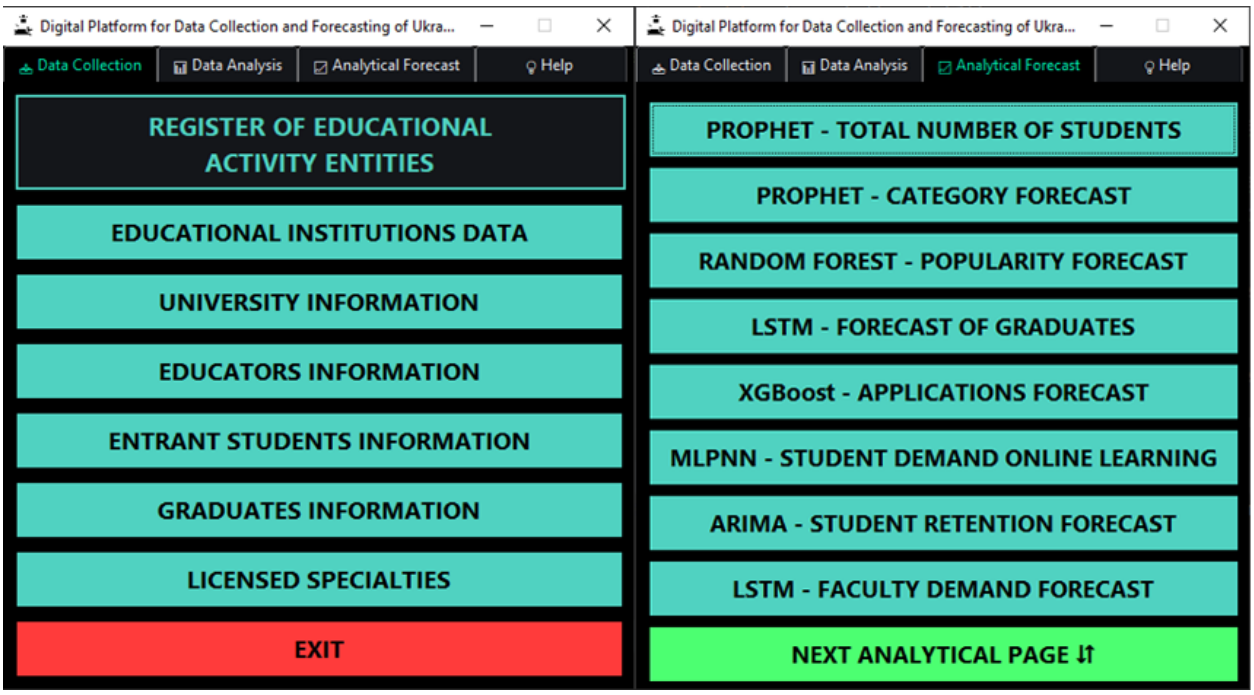


Рисунок 2 – Функціональні вкладки інтерфейсу

2. Створено та протестовано прогнозні моделі, які дають змогу аналізувати динаміку студентського контингенту, вступних потоків та результатів випуску. Моделі забезпечують основу для оцінки ефективності освітніх програм та формування прогнозів розвитку університетів.

3. Розроблено комплексні інструменти аналітики, що дають змогу порівнювати університети за актуальними освітніми показниками, регіонами та спеціальностями, а також відстежувати зміни популярності освітніх програм у часі.

4. Проведено тестування програмної платформи, яке підтвердило стабільність роботи системи, зручність користувацького інтерфейсу та можливість подальшого масштабування функціоналу. На рис. 3 наведено приклад аналітичного модуля системи, що відображає динаміку вступників і випускників за спеціальністю 122 та їх десятирічний прогноз, сформований моделлю Prophet. Прогнозування здійснено на основі навчальних даних ЄДЕБО та макроекономічних індикаторів World Bank, що забезпечує врахування соціально-економічних чинників, які впливають на попит на освітні програми.

Подальші дослідження будуть спрямовані на розширення набору даних, зокрема інтеграцію показників ринку праці, міжнародних рейтингів університетів та індикаторів студентської мобільності. Важливим напрямом розвитку є також інтеграція додаткових міжнародних джерел статистики, що дасть змогу підвищити точність оцінки тенденцій і розширити можливості системи для підтримки стратегічного планування у сфері вищої освіти.

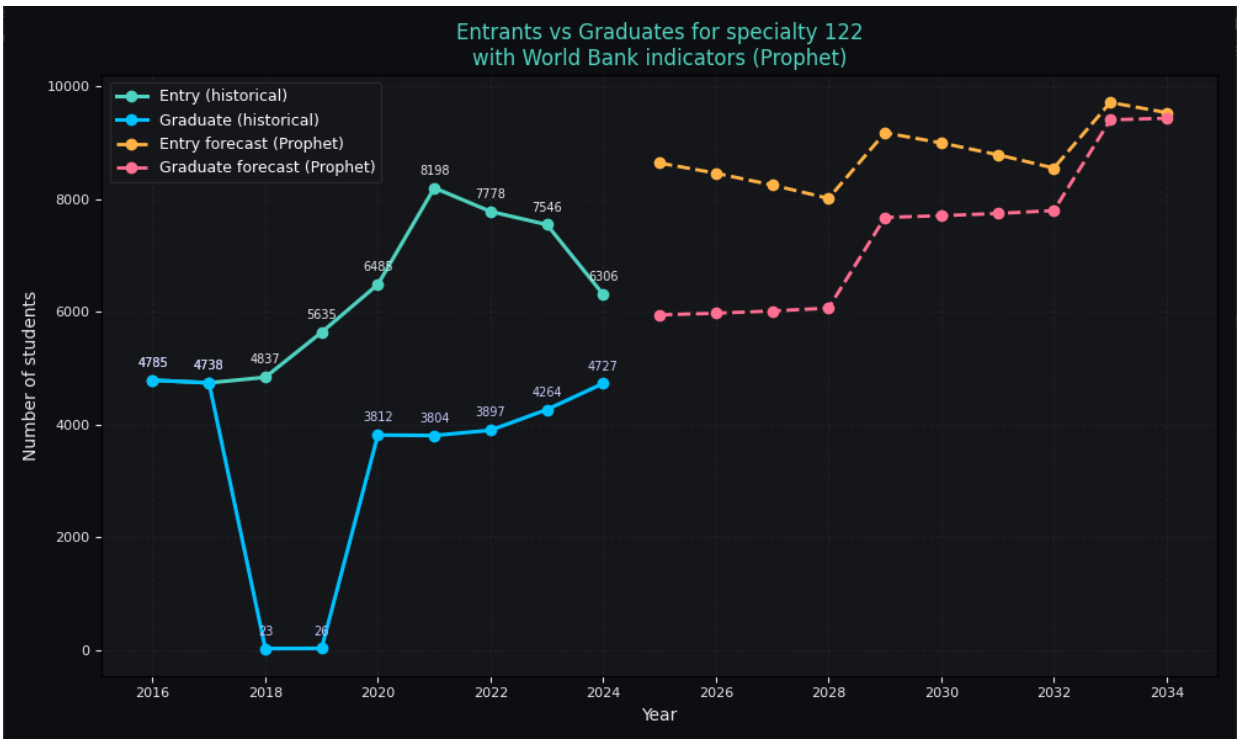


Рисунок 3 – Прогнозування динаміки кількості вступників та випускників

Отримані результати підтверджують актуальність створення подібних інформаційних систем та їх практичну значущість для розвитку вищої освіти України.

Список використаних джерел

1. Єдина державна електронна база з питань освіти (ЄДЕБО). URL: https://info.edbo.gov.ua (дата звернення 14.10.2025).

2. World Bank Open Data. URL: https://data.worldbank.org/ (дата звернення 18.10.2025).

УДК 519.85:355

Плець В. В., здобувачка 3 курсу
спеціальності 122 Комп'ютерні науки,
Січко Т. В., канд. техн. наук, доцент
кафедри інформаційних технологій

МЕТОДИ МАТЕМАТИЧНОГО ПРОГРАМУВАННЯ У ВІЙСЬКОВІЙ СПРАВІ

Донецький національний університет імені Василя Стуса, м. Вінниця

Вступ. У сучасних умовах ведення війни ефективність воєнних дій значною мірою залежить від здатності командування приймати швидкі та обґрунтовані рішення на основі об'єктивних даних. У таких умовах інтуїтивні рішення, що

ґрунтуються виключно на досвіді командира, часто виявляються неефективними або навіть катастрофічними. На допомогу приходить математичне програмування – розділ прикладної математики, який досліджує методи знаходження оптимальних рішень в умовах обмежених ресурсів.

Мета застосування цих методів у військовій справі полягає у забезпеченні максимальної ефективності бойових та забезпечувальних операцій за мінімальних витрат. Це може виражатися в мінімізації часу виконання завдання, зниженні паливно-матеріальних витрат, скороченні бойових втрат або максимізації шкоди противнику. У реальних бойових умовах, де кожна хвилина, кожен літр палива та кожна одиниця боєприпасів можуть мати вирішальне значення, науково обґрунтований підхід до планування стає не просто перевагою, а життєвою необхідністю [1; 2].

Математичні моделі дають змогу оцінити наслідки рішень, моделювати сценарії та вибирати оптимальний варіант дій.

Методи математичного програмування є різнорідними і застосовуються залежно від природи задачі, що вирішується.

1. Лінійне програмування є найбільш розповсюдженим методом. Воно застосовується там, де цільова функція й обмеження є лінійними. У військовій сфері це розподіл ресурсів: планування харчування, оптимізація маршрутів постачання палива й боєприпасів, розподіл завдань між підрозділами. Класичний приклад – транспортна задача мінімізації витрат.

2. Нелінійне програмування застосовується, коли залежності між змінними нелінійні. Це характерно для технічних систем: розрахунок траєкторії ракети, оптимізація радарів, налаштування систем наведення.

3. Динамічне програмування оптимізує багатокрокові процеси, де кожне рішення впливає на наступні. У військовій справі це планування наступу чи відступу, введення резервів, вибір послідовності операцій.

4. Стохастичне програмування (СП) враховує невизначеність. Війна – стохастичний процес, тому СП дає змогу створювати плани, стійкі до ризиків: втрати конвою, погодні зміни, збої техніки.

5. Цілочисельне програмування застосовується, коли змінні можуть бути лише дискретними. Це формування бойових груп, вибір місць розташування баз, визначення оптимальної кількості техніки.

6. Багатокритеріальна оптимізація (БКО) відображає реальні умови, коли існує кілька суперечливих цілей: мінімізація втрат, витрат, часу та одночасна максимізація шкоди противнику. Методи БКО дають змогу знайти збалансовані рішення [3].

Сфера застосування методів математичного програмування охоплює практично всі аспекти військової діяльності.

Військова логістика. Це класична галузь застосування. Оптимізаційні моделі використовуються для визначення найкоротших і найбезпечніших маршрутів постачання, розрахунку необхідної кількості транспортних засобів, планування розміщення складів тилу та управління запасами. Дослідження показують, що використання логістичних моделей дозволяє знизити витрати палива на 15–20 % [4]. Стохастичні моделі дає змогу визначити оптимальний рівень запасів,

щоб уникнути як дефіциту (ризик для операції), так і надлишку (неефективне використання ресурсів).

Планування бойових операцій. Системи планування операцій, як-от «OPLAN» (Operational Planning), що використовуються в країнах НАТО, активно інтегрують алгоритми динамічного та цілочисельного програмування. Вони допомагають розробити послідовність дій для підрозділів, визначити точки застосування основних зусиль, оптимально розподілити сили та засоби по фронту для досягнення оперативних цілей з мінімальними втратами.

Розвідка та моніторинг. Обмеженість розвідувальних ресурсів (дрони, супутники, агентурні мережі) вимагає їх оптимального розподілу. Стохастичні моделі та методи оптимізації (наприклад, теорія ігор для передбачення дій противника) застосовуються для прогнозування ймовірних напрямків руху ворожих підрозділів, визначення пріоритетних цілей для зйомки та оптимального маршруту патрулювання БПЛА.

Кібербезпека. У віртуальному просторі математичне програмування використовується для розподілу обчислювальних ресурсів і навантаження між серверами, оптимізації часу реагування на кібератаки, побудови стійких до збоїв мережевих топологій та розробки стратегій проактивного захисту інформаційних систем.

Медичне забезпечення. Евакуація поранених – це процес, де кожна хвилина коштує життя. Оптимізаційні алгоритми застосовуються для планування маршрутів медичної евакуації, визначення оптимальних місць розгортання польових госпіталів (задача розміщення) та розподілу медикаментів між ними. Такі системи дають змогу значно зменшити середній час надання допомоги.

Моделювання бойових дій. Математичні моделі, як-от рівняння Ланчестера, побудовані на системах диференціальних рівнянь, дають змогу кількісно оцінити співвідношення сил у конфлікті. Вони дають змогу прогнозувати результат зіткнення залежно від чисельності сторін, їх вогневої потужності, тактики та інших факторів, допомагаючи оцінити ефективність різних стратегій до їх реалізації на полі бою.

Штучний інтелект (ШІ) та автономні системи. Сучасні бойові системи, як-от автономні дрони або системи підтримки прийняття рішень (СППР), базуються на складних алгоритмах оптимізації. ШІ використовує методи машинного навчання для розпізнавання цілей, побудови оптимальних траєкторій польоту, уникнення засобів ППО та прийняття рішень у реальному часі в умовах неповної інформації.

Висновки. Методи математичного програмування перетворилися з абстрактного академічного інструменту на потужний практичний механізм підвищення ефективності військового управління на всіх рівнях. Вони дають змогу перевести планування з мистецтва на основі інтуїції та досвіду на наукову основу, що забезпечує оптимізацію ресурсів, мінімізацію ризиків, скорочення втрат, підвищення точності та швидкості прийняття рішень.

Системне використання цих методів сприяє створенню так званої «розумної армії» – гнучкої, адаптивної, здатної швидко реагувати на зміни обстановки та ефективно діяти в умовах сучасних гібридних воєн. Майбутній розвиток тех-

нологій, зокрема поширення штучного інтелекту нового покоління та поява квантових обчислень, які зможуть за лічені хвилини вирішувати задачі, непідвладні сьогоднішнім комп'ютерам, ще більше розширить горизонти застосування математичного програмування, зробивши його абсолютно невід'ємним складником стратегічної переваги на полі бою.

Список використаних джерел

1. Вітлінський В. В., Верченко П. І. Математичне програмування: теорія, методи, застосування. Київ: КНЕУ, 2020. С. 250.

2. Taha H. A. Operations Research: An Introduction. 10th ed. Pearson, 2017. P. 125.

3. Бондаренко М. Ф., Кузьменко О. В. Оптимізаційні методи та моделі у військовій логістиці. Львів: НАСВ, 2021. С. 110.

4. Hillier F. S., Lieberman G. J. Introduction to Operations Research. New York: McGraw-Hill, 2020. P. 86.

5. Січко Т. В., Нескородєва Т. В. Методичні вказівки щодо виконання лабораторних робіт з дисципліни «Методи оптимізації та дослідження операцій» для студентів СО «Бакалавр» денної та заочної форм навчання спеціальностей 122 «Комп'ютерні науки», 113 «Прикладна математика». Вінниця: ДонНУ імені Василя Стуса. 2020. С. 50.

6. Експертні та рекомендаційні системи: навч. посіб. / Т. В. Нескородєва, Є. Є. Федоров, Т. В. Січко, А. Р. Нескородєва. Вінниця: ДонНУ імені Василя Стуса, 2022. 208 с.

УДК 004.89:004.912

Шендрик В. О., здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,
Ніколюк П. К., д-р фіз.-мат. наук, професор кафедри інформаційних технологій

РОЗРОБКА ПРОСТОГО ЧАТ-БОТА НА ОСНОВІ МЕТОДІВ ОБРОБКИ ПРИРОДНОЇ МОВИ (NLP)

Донецький національний університет імені Василя Стуса, м. Вінниця

Чат-боти стали невід'ємною частиною сучасних інформаційних систем, забезпечуючи автоматизовану взаємодію з користувачами у сферах електронної комерції, підтримки клієнтів та освіти. Традиційні боти, що базуються на жорстких правилах (rule-based), мають обмежену гнучкість і не здатні ефективно обробляти неструктуровані запити. Використання методів обробки природної мови (Natural Language Processing, NLP) дає змогу створювати інтелектуальних агентів, здатних розуміти контекст і наміри користувача.

Метою роботи є розробка архітектури та програмна реалізація простого чат-бота, що використовує методи NLP для класифікації намірів користувача та

генерації відповідей. В основі розробки лежить використання мови програмування Python та бібліотек NLTK (Natural Language Toolkit) і Scikit-learn.

Процес створення чат-бота складається з кількох ключових етапів попередньої обробки даних, які є критичними для точності моделі:

1. Токенізація: розбиття вхідного текстового потоку на окремі елементи (слова, символи).

2. Очищення та нормалізація: видалення розділових знаків, переведення тексту в нижній регістр та лематизація (зведення слів до їх словникової форми) для зменшення варіативності даних.

3. Векторизація: перетворення текстових даних у числовий формат, зрозумілий для алгоритмів машинного навчання. Для цього доцільно використовувати метод «мішка слів» (Bag of Words) або TF-IDF, який оцінює важливість слова в контексті документа.

Для розпізнавання намірів (intents) користувача використано алгоритм логістичної регресії або наївний байєсівський класифікатор. Ці методи є ефективними для задач класифікації тексту з невеликим обсягом навчальних даних. Навчальний набір даних формується у вигляді JSON-структури, що містить категорії намірів (наприклад, «вітання», «запит ціни», «прощання»), приклади фраз для кожного наміру та відповідні шаблони відповідей.

Важливим аспектом є архітектура прийняття рішень: після отримання запиту система обчислює ймовірність приналежності вхідного тексту до кожного з відомих намірів. Якщо максимальна ймовірність перевищує встановлений поріг (наприклад, 0.7), бот обирає відповідь із відповідної категорії; в іншому випадку – генерує уточнювальне запитання або повідомляє про нерозуміння. Загальну схему функціонування розробленого чат-бота наведено на рис. 1.

Експериментальні результати. Для оцінки ефективності розробленої системи було проведено тестування на відкладеній вибірці даних. Логістична регресія продемонструвала загальну точність (Accuracy) на рівні 92 %, що виявилося вищим показником, порівняно з наївним байєсівським класифікатором (89 %). Також встановлено, що використання лематизації підвищує F1-меру моделі на 4.5 %, що підтверджує критичну важливість етапу попередньої обробки даних, описаного вище.

Практична реалізація підтвердила, що навіть прості моделі NLP значно перевершують сценарні підходи за гнучкістю. Вони дають змогу обробляти запити з помилками, синонімами та різними формулюваннями однієї й тієї ж думки. Однак основною проблемою залишається обробка контексту діалогу, що вимагає застосування більш складних рекурентних нейронних мереж (RNN) або трансформерів у майбутніх ітераціях розробки.

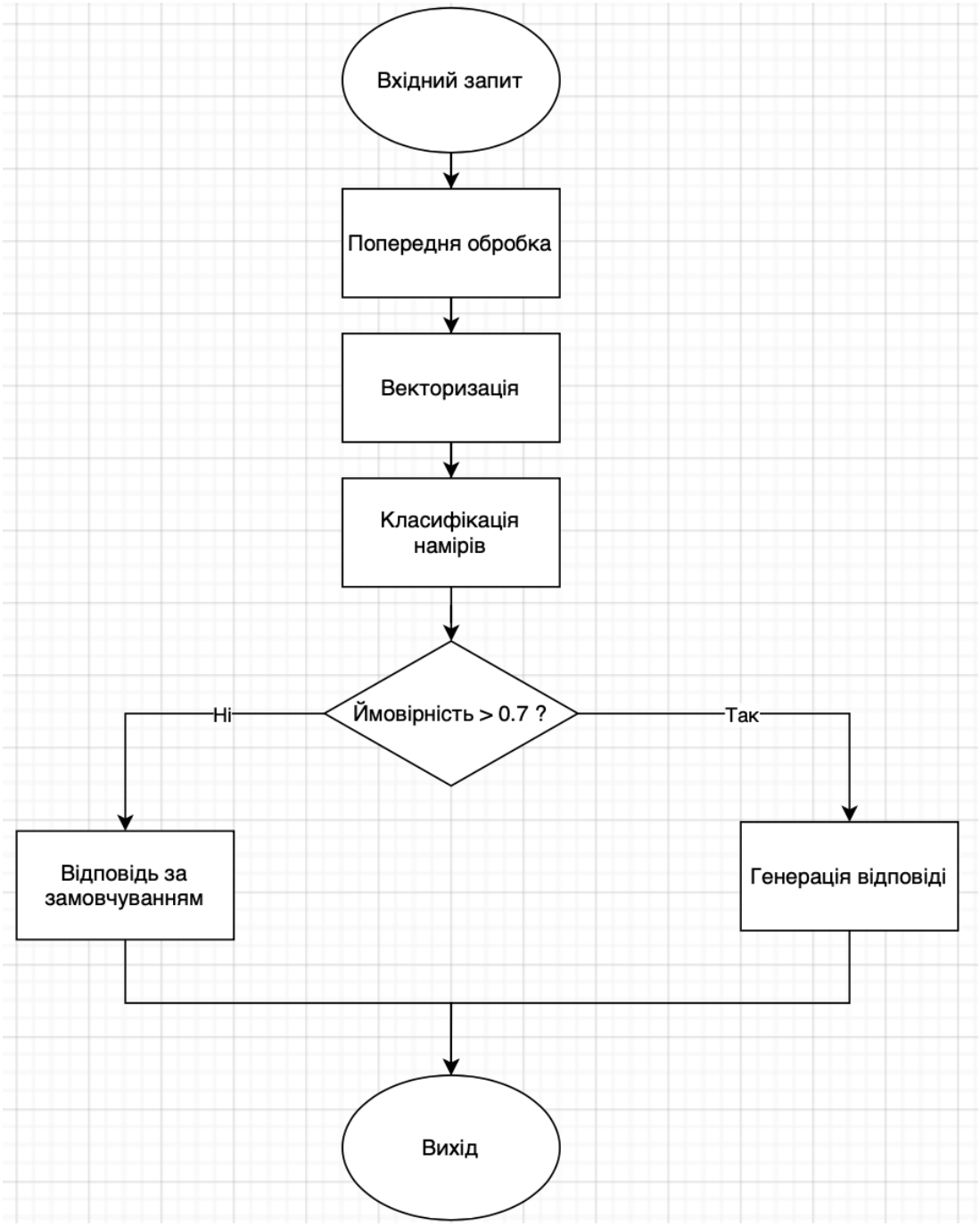


Рисунок 1 – Структурна схема алгоритму обробки запиту розробленим чат-ботом

Отже, використання базових методів NLP, як-от токенізація, лематизація та векторна класифікація, дає змогу створювати ефективні чат-боти з мінімальними обчислювальними витратами. Такий підхід є оптимальним для вирішення типових задач консультаційного характеру.

Список використаних джерел

1. Jurafsky D., Martin J. H. Speech and Language Processing. 3rd ed. draft. 2025. URL: https://web.stanford.edu/~jurafsky/slp3/ (дата звернення: 21.11.2025).

2. Селін Я. Ю. Аналіз інструментів для створення NLP чат-бота. *Радіоелектроніка та молодь у XXI ст.* 2024. Т. 6. С. 7–9.

3. Singh R. P., Singh K. Natural Language Processing in Chatbots. *IOSR Journal of Multidisciplinary Research.* 2025. Vol. 2, № 4. P. 1–7.

УДК 004.001

Губчакевич В. Р., здобувач 2 курсу
ОС «Магістр» ФЗ Комп'ютерні науки,
Січко Т. В., канд. техн. наук, доцент кафедри
інформаційних технологій

ПЕРЕВАГИ ВИКОРИСТАННЯ BI-ТЕХНОЛОГІЙ У РОБОТІ HR-МЕНЕДЖЕРА

Донецький національний університет імені Василя Стуса, м. Вінниця

У сучасних умовах цифрової трансформації бізнесу ефективне управління персоналом стає ключовим чинником конкурентоспроможності підприємств. Людський капітал є основним стратегічним ресурсом організації, тому здатність своєчасно приймати обґрунтовані кадрові рішення має вирішальне значення для її розвитку. Одним із найважливіших інструментів, що дають змогу досягти цього, є технології бізнес-аналітики (Business Intelligence, BI), які забезпечують комплексну обробку, аналіз і візуалізацію даних про працівників.

Використання BI у сфері HR допомагає не лише підвищити якість управлінських рішень, а й сформувати новий підхід до роботи з персоналом – data-driven HR-менеджмент, заснований на аналітичних показниках. Як зазначає Anna Karmańska у дослідженні «The Benefits of HR Analytics», аналітика в HR забезпечує компаніям перехід від інтуїтивного до доказового управління персоналом, що значно підвищує ефективність бізнес-процесів. Натомість у статті «Implementation of Business Intelligence and Analytics in Human Resource Management as a Core Competency for Sustainability» (MDPI) наголошується, що BI-аналітика стає невід'ємним складником сталого розвитку компаній, адже забезпечує стратегічне планування та контроль соціальних показників [1; 2].

Інструменти бізнес-аналітики (Power BI, Tableau, Qlik Sense тощо) забезпечують збір, інтеграцію та візуалізацію даних про персонал у єдиному середовищі. Це дає змогу HR-менеджерам оперативно аналізувати ключові показники ефективності (KPI): рівень плинності кадрів, час закриття вакансій, витрати на найм, результати оцінки персоналу, рівень задоволеності працівників. Завдяки інтерактивним панелям BI керівництво отримує змогу виявляти тенденції, прогнозувати ризики та ухвалювати рішення на основі фактів, а не припущень.

Згідно з результатами дослідження Anna Karmańska, 88,6 % HR-фахівців відзначили покращення рекрутингових процесів після впровадження BI-аналітики, а 83,6 % – підвищення ефективності планування персоналу. Це свідчить про те, що BI-технології дають змогу створити цілісну систему моніторингу кадрових процесів, забезпечуючи швидке реагування на зміни в організації. BI перетворює

великі обсяги даних на зрозумілу та візуально структуровану інформацію, що сприяє своєчасному прийняттю управлінських рішень [1].

У публікації MDPI (Implementation of Business Intelligence and Analytics in Human Resource Management as a Core Competency for Sustainability) BI розглядають як ключовий елемент сталого розвитку організації. Автори наголошують, що завдяки BI-аналітиці HR-департаменти можуть ефективно прогнозувати майбутні потреби у кадрах, формувати стратегії розвитку талантів, а також оцінювати ефективність навчальних програм. Це особливо важливо для довгострокового планування та підтримання балансу між економічними і соціальними цілями компанії [2].

Khaliq та Saritha у статті «Benefits and Challenges of Adopting HR Analytics» зазначають, що BI-інструменти скорочують час підготовки звітності більш ніж на 50 %, знижують рівень плинності кадрів до 25 % і підвищують точність прогнозів щодо кадрових ризиків. Завдяки цьому HR-аналітика з використанням BI перетворює традиційний відділ кадрів із адміністративного підрозділу на аналітичний центр прийняття стратегічних рішень [3].

У дослідженні «HR Analytics for Data-Driven Employee Attrition Management» (HRMARS) наведено практичний приклад застосування BI-систем для аналізу причин звільнення працівників. Використання Power BI дало змогу створити панелі моніторингу, які виявляють ризикові групи персоналу, прогнозують можливість звільнення та допомагають HR-менеджерам розробляти індивідуальні стратегії утримання. Такий підхід допомагає компаніям не лише реагувати на проблеми, а й запобігати їм, що знижує витрати на рекрутинг і підвищує стабільність колективу [4].

Практичні переваги BI-технологій підкреслюють і в статті RockCrest «What Is Business Intelligence Software and How It Helps HR». Серед основних переваг – покращення видимості кадрових процесів (workforce visibility), оптимізація рекрутингу, оцінка ефективності працівників і підвищення рівня прозорості управлінських рішень. BI дає змогу створити інтегровану систему HR-даних, яка об’єднує інформацію про кадри, продуктивність, компенсації та навчання, що робить управління персоналом більш обґрунтованим і ефективним [5].

Водночас у наукових роботах наголошується на низці викликів, пов’язаних зі впровадженням BI-рішень у HR-середовищі. Серед них – висока вартість ліцензійного програмного забезпечення, складність інтеграції з наявними HR-системами, потреба у спеціалістах із аналітичними компетенціями та питання захисту персональних даних. Однак більшість досліджень сходяться на тому, що переваги BI значно переважають потенційні труднощі. Розвиток аналітичної культури, навчання працівників і поступове впровадження BI-модулів дають змогу мінімізувати ризики та забезпечити ефективність інноваційних рішень.

Отже, використання BI-технологій у HR створює нову парадигму управління людськими ресурсами, де рішення приймаються не на основі інтуїції, а на основі даних, фактів і прогнозів. Це підвищує довіру до HR-функцій, покращує якість комунікації між керівництвом і працівниками та сприяє побудові прозорої, результативної системи управління персоналом**.**

Висновки. Отже, впровадження BI-інструментів у сферу HR-аналітики є одним із ключових напрямів розвитку сучасного менеджменту. Використання

таких технологій дає змогу систематизувати кадрові дані, підвищити точність аналітичних оцінок, скоротити витрати часу та ресурсів, а також сприяє формуванню довгострокових стратегій розвитку персоналу.

BI-аналітика забезпечує перехід від традиційного, інтуїтивного управління до науково обґрунтованого, прогнозного та стратегічного підходу, що підвищує ефективність усіх рівнів організаційного управління. Вона дає змогу HR-департаментам виконувати не лише адміністративні, а й стратегічні функції, перетворюючи їх на аналітичний центр компанії.

Отже, BI-технології у HR-сфері є не просто технічним нововведенням, а фундаментальною зміною у підходах до управління людськими ресурсами, яка відкриває нові можливості для сталого розвитку, підвищення продуктивності та зміцнення конкурентних позицій підприємств у цифрову епоху.

Список використаних джерел

1. The Benefits of HR Analytics. URL: https://journals.ue.wroc.pl/pn/article/view/612 (дата звернення: 23.10.2025).

2. Implementation of Business Intelligence and Analytics in Human Resource Management as a Core Competency for Sustainability. URL: https://www.mdpi.com/2504-3900/101/1/22 (дата звернення: 23.10.2025).

3. Benefits and Challenges of Adopting HR Analytics: A Comprehensive Review. URL: https://journaljemt.com/index.php/JEMT/article/view/1159 (дата звернення: 23.10.2025).

4. What Is Business Intelligence Software and How It Helps HR. URL: https://www.rockcrest.com/post/what-is-business-intelligence-software-and-how-it-helps-hr (дата звернення: 23.10.2025).

СЕКЦІЯ 4
БАЗИ ДАНИХ І ЗНАНЬ. BIG DATA

УДК 004.428.2

Лухверчик С. А., здобувач 2 курсу
ОС «Магістр» спеціальності
F3 Комп'ютерні науки,
Потапова Н. А., канд. екон. наук, доцент,
доцент кафедри інформаційних технологій

АНАЛІТИЧНІ МЕТОДИ ОБРОБКИ ВЕЛИКИХ ДАНИХ У ЛОГІСТИЧНИХ СИСТЕМАХ

Донецький національний університет імені Василя Стуса, м. Вінниця

Цифрова трансформація логістичних процесів призводить до стрімкого зростання обсягів інформації, яку генерують транспортні засоби, складські системи, модулі відстеження замовлень та аналітичні сервіси. Дані надходять у вигляді координат, часових міток, атрибутів замовлень і телеметричних параметрів, формуючи різнорідні інформаційні потоки.

Для логістичної галузі характерні проблеми неповноти, розрізненості та нерівномірного просторового розподілу даних, що ускладнює розробку ефективних моделей підтримки рішень. Саме тому методи Big Data стають ключовим інструментом для виявлення закономірностей, прогнозування попиту та формування структурованих логістичних зон.

На основі аналізу реальних логістичних наборів даних виконано структуризацію просторової інформації та оцінено вплив попередньої підготовки даних на якість подальшого аналізу. Виявлено, що пропуски, дублікати координат і аномальні значення значно спотворюють результати моделювання, тому критичними є етапи очищення, нормалізації та фільтрації.

Ключовим напрямом дослідження стала кластеризація точок доставки – один з найпоширеніших методів просторової аналітики в логістиці. Алгоритм K-Means продемонстрував ефективність у випадках відносно рівномірного розподілу точок, даючи змогу формувати компактні й інтерпретовані кластери. Водночас DBSCAN виявив здатність коректно групувати точки в умовах нерівномірної щільності, що є типовою проблемою для кур'єрських та регіональних служб доставки. Результати показали, що кластеризація допомагає:

1) формувати логістичні мікрорегіони для подальшої маршрутизації;
2) оцінювати динаміку та щільність попиту;
3) візуалізувати «гарячі зони» доставки;
4) визначати території з потенційними логістичними ризиками;
5) знижувати навантаження на транспортні ресурси завдяки правильному групуванню точок.

У межах дослідження було побудовано візуалізації кластерних моделей, що дають змогу наочно представити логістичну структуру території.

На рис. 1 наведено приклади типових результатів кластеризації у логістиці: компактні центроїдні кластери (K-Means) та кластери довільної форми (DBSCAN), що відображають просторову нерівномірність.

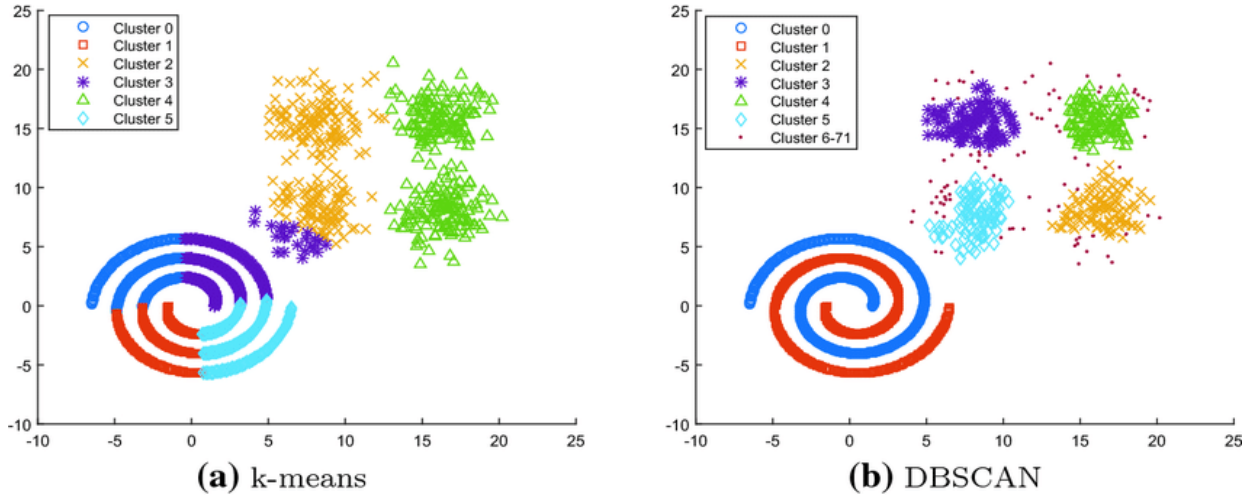


Рисунок 1 – Порівняння результатів кластеризації методом K-Means та DBSCAN

Аналітичний підхід до обробки Big Data у логістиці забезпечує підвищення точності оцінювання попиту, прозорість логістичних операцій та можливість формувати адаптивні регіональні моделі, придатні для практичного застосування в кур'єрських службах, дистрибуційних компаніях і системах міської доставки.

Методи аналітики великих даних відіграють важливу роль у формуванні сучасних логістичних рішень. Проведене дослідження показало, що кластеризація є одним з найефективніших інструментів попереднього структурування даних, яке дає змогу суттєво покращити подальші процеси моделювання та управління. Використання K-Means та DBSCAN забезпечує можливість адаптивного поділу логістичного простору залежно від щільності та характеристик розташування точок доставки.

Отже, Big Data-підходи створюють основу для розробки більш гнучких, стійких і прозорих логістичних систем, що відповідають викликам сучасної цифрової економіки.

Список використаних джерел

1. Han J., Kamber M., Pei J. Data Mining: Concepts and Techniques. Amsterdam: Elsevier, 2012. 740 p.

2. Provost F., Fawcett T. Data Science for Business. Sebastopol: O'Reilly Media, 2013. 414 p.

3. Gandomi A., Haider M. Beyond the hype: Big Data concepts, methods, and analytics. *International Journal of Information Management.* 2015. URL: https://doi.org/10.1016/j.ijinfomgt.2014.10.007 (дата звернення: 22.11.2025).

4. Raghupathi W., Raghupathi V. Big Data Analytics in Healthcare: Promise and Potential. *Health Information Science and Systems.* 2014. № 3. URL: https://doi.org/10.1186/2047-2501-2-3 (дата звернення: 22.11.2025).

5. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise (DBSCAN) / M. Ester, H.-P. Kriegel, J. Sander, X. Xu. *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96).* Portland, USA, 1996. P. 226–231. URL: https://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf (дата звернення: 22.11.2025).

УДК 004.65:004.8

Трохимчук О. М., здобувачка 3 курсу,
Січко Т. В., канд. техн. наук, доцент,
доцент кафедри інформаційних технологій

БАЗИ ДАНИХ ТА ШТУЧНИЙ ІНТЕЛЕКТ

Донецький національний університет імені Василя Стуса, м. Вінниця

Стрімкий розвиток інформаційних технологій призводить до безпрецедентного зростання обсягів структурованих та неструктурованих даних. Це зумовлює потребу у створенні інтелектуальних систем, здатних забезпечувати не лише зберігання, а й глибоку аналітичну обробку інформації. Бази даних відіграють ключову роль у цифровій трансформації сучасних організацій, а штучний інтелект (ШІ) виступає каталізатором підвищення їх ефективності. Сучасні СКБД інтегрують алгоритми машинного навчання, що дає змогу автоматизувати оптимізацію запитів, прогнозувати стан системи та виявляти аномалії у даних [1].

Актуальність теми полягає у зростанні ролі інтелектуальних СКБД у бізнесі, медицині, фінансах, виробництві та державному секторі. Сучасні компанії переходять на моделі data-driven decision making, де якість і швидкість обробки даних стають критичними факторами конкурентоспроможності. У 2023–2025 рр. відзначається тенденція до впровадження автономних баз даних, які використовують ШІ для самоналаштування, самовідновлення та самостійного управління продуктивністю системи. Це дає змогу скоротити людський фактор, знизити ризики та оптимізувати витрати на адміністрування [2].

Метою роботи є дослідження взаємозв’язку між СКБД та алгоритмами штучного інтелекту, аналіз сучасних напрямів їх інтеграції, а також формування прогнозів щодо розвитку інтелектуальних СКБД у найближчі роки.

Для досягнення поставленої мети необхідно виконати такі завдання:

– дослідити типи сучасних СКБД та їх можливості;

– визначити роль ШІ у підвищенні ефективності роботи баз даних;

– проаналізувати приклади застосування інтелектуальних СКБД у різних сферах;

– розглянути тенденції розвитку технологій зберігання та обробки даних;

– сформувати висновки щодо перспектив інтеграції ШІ та СКБД у світовій практиці [3].

Бази даних сучасного покоління можна поділити на реляційні, нереляційні, графові та гібридні. Реляційні СКБД залишаються найбільш поширеними завдяки своїй стабільності та структурованому підходу до зберігання інформації. Нереляційні СКБД, або NoSQL-системи, набули популярності через здатність опрацьовувати великі масиви неструктурованих даних. Графові СКБД активно використовуються у задачах аналізу зв’язків та соціальних мереж [4].

Штучний інтелект у сучасних базах даних застосовується для:

– автоматичної оптимізації запитів;

– прогнозування продуктивності;

– виявлення аномалій у даних;
– інтелектуального кешування;
– семантичного пошуку та класифікації.

Окрему увагу привертають автономні системи керування базами даних, як-от Oracle Autonomous Database, Google AlloyDB та Azure SQL з Copilot. Ці системи використовують вбудовані алгоритми ШІ, забезпечуючи роботу за принципом self-driving databases. Вони здатні автоматично обирати оптимальні індекси, оновлювати конфігурацію, прогнозувати навантаження та рекомендувати оптимальні плани запитів [5].

Отже, сучасний розвиток штучного інтелекту радикально трансформує можливості систем керування базами даних. Інтеграція інтелектуальних алгоритмів дає змогу створити СКБД нового покоління – автономні, високопродуктивні та здатні до самостійного аналізу власних процесів. Найближчими роками очікується повна автоматизація більшості адміністративних завдань.

Список використаних джерел

1. SQL Server and Azure SQL: AI-Powered Optimization. *Microsoft.* 2024. URL: https://learn.microsoft.com/sql/ (дата звернення: 24.11.2025).
2. Autonomous Database Technical Overview. *Oracle.* 2023. URL: https://www.oracle.com/autonomous-database/ (дата звернення: 24.11.2025).
3. AI-Driven Data Management Trends. *Google Cloud.* 2025. URL: https://cloud.google.com/blog/ (дата звернення: 24.11.2025).
4. Intelligent Database Systems Report. *IBM Research.* 2024. URL: https://research.ibm.com (дата звернення: 25.11.2025).
5. NoSQL and Machine Learning Integration Guide. *MongoDB Inc.* 2023. URL: https://www.mongodb.com/resources (дата звернення: 25.11.2025).
6. Степанюк О. С., Січко Т. В. Особливості використання реляційних та нереляційних баз даних в Big Data. *Комп'ютерні технології обробки даних: матеріали всеукр. наук.-практ. конф.* Вінниця, 2020. С. 103–106.

УДК 004.6:[004.056.53:004.9](043.2)

Байраківська В. В., здобувачка 3 курсу,
Січко Т. В., канд. техн. наук, доцент,
доцент кафедри інформаційних технологій

ВПРОВАДЖЕННЯ ПРОЦЕДУР ОЧИЩЕННЯ ДАНИХ (DATA CLEANSING) ДЛЯ ПІДВИЩЕННЯ ДОСТОВІРНОСТІ ІНФОРМАЦІЇ В ІНФОРМАЦІЙНОМУ ПРОСТОРІ

Донецький національний університет імені Василя Стуса, м. Вінниця

Вступ. Електронні реєстри є важливим елементом сучасних інформаційних систем, оскільки забезпечують зберігання, обробку та використання структурованих даних у державному, медичному, фінансовому чи корпоративному секторах.

Достовірність даних у таких системах визначається їх точністю, повнотою, узгодженістю та актуальністю. Саме ці характеристики дають змогу ухвалювати правильні управлінські рішення та формувати надійну аналітичну інформацію. Проте на практиці низька якість даних є однією з головних проблем цифрових реєстрів. Вона проявляється у вигляді дублювання записів, наявності неповних або застарілих відомостей, суперечливих полів і помилкових форматів. Такі дефекти часто призводять до спотворення статистики, хибних аналітичних висновків, а іноді – до значних фінансових втрат або неправомірних управлінських рішень.

Мета роботи. Проаналізувати підходи і технології очищення даних у електронних реєстрах для підвищення їх достовірності та ефективності функціонування. Робота спрямована на обґрунтування важливості системного підходу до забезпечення якості даних і визначення практичних методів виявлення та усунення помилок.

Впровадження системного процесу очищення даних (Data Cleansing або Data Cleaning) є невід'ємним елементом забезпечення якості відомостей в інформаційному просторі. Під очищенням даних розуміють сукупність процедур, спрямованих на виявлення, виправлення або видалення некоректних, неповних, неточних чи дубльованих записів у базах даних [3]. На практиці виокремлюють кілька основних типів дефектів: неповнота (наявність пропусків або значень «null»), неточність (помилки введення, застарілі дані, неправильні формати), несуперечливість (розбіжності між записами про одну сутність) та дублювання (наявність кількох записів для одного об'єкта) [1; 6]. Кожен із цих дефектів знижує рівень довіри до джерела, ускладнює звітність і може спричиняти додаткові витрати на перевірку чи корекцію інформації.

Процес очищення даних зазвичай реалізується поетапно. Першим етапом є профілювання даних, тобто дослідження їх поточного стану з метою виявлення джерел і масштабів проблем. На цьому етапі аналізуються пропуски, аномалії, помилки форматів, частота дублювання записів та статистичні закономірності. Такий аналіз дає змогу визначити пріоритети подальших дій та оцінити, які саме ділянки потребують найглибшого втручання [5].

Другим етапом є розробка правил очищення, що визначає стандарти й бізнес-логіку корекції даних. До прикладу, можуть бути встановлені обов'язкові поля, стандартизовані формати дат або кодів, правила для виявлення та усунення дублікатів, а також алгоритми обробки пропущених значень (видалення, заповнення або позначення як невідомих). На цьому етапі важливо залучити експертів предметної області, аналітиків та IT-фахівців, щоб забезпечити узгодженість технічних і змістовних вимог [7].

Третім етапом є реалізація процедур очищення. Для цього застосовуються автоматизовані скрипти, ETL-інструменти (Extract → Transform → Load) та спеціалізовані програмні утиліти. Вони виконують перевірку відповідності правил, перетворення форматів, злиття або видалення дублікатів, виправлення значень за визначеними алгоритмами. Коли автоматизація не може забезпечити 100-відсоткову коректність результату, застосовується ручна верифікація даних. Особливо це важливо для контекстів, де потрібна логічна оцінка фахівця, наприклад, під час об'єднання близьких, але не ідентичних записів. Усі зміни повинні фіксуватися в журналах аудиту із зазначенням часу, виконавця та причини корекції [4].

Завершальний етап – моніторинг і аудит якості даних. Після первинного очищення необхідно впровадити систему постійного контролю, щоб запобігти появі нових дефектів. Для цього визначають ключові показники якості (наприклад, частка заповнених полів, рівень дублювання, кількість некоректних форматів), а також встановлюють регулярні перевірки та звітування. Як показують дослідження, впровадження безперервного моніторингу дає змогу скоротити кількість помилок до 70 % уже протягом перших шести місяців [2]. Окрім технічного контролю, важливим є й організаційний аспект – формування культури якості даних серед користувачів реєстру, навчання персоналу та призначення відповідальних осіб, які контролюють дотримання стандартів [5].

Технічний складник очищення даних включає використання різних методів і засобів. Найпоширенішими інструментами є ETL-платформи, утиліти для профілювання даних і системи моніторингу якості. Для перевірки правильності форматів застосовуються регулярні вирази (regex), для пошуку дублювань – нечітке зіставлення (fuzzy matching), що дає змогу знаходити схожі записи, навіть якщо вони містять орфографічні розбіжності [3; 4]. Важливо також реалізовувати стандартизацію даних: приведення назв, адрес, кодів або одиниць вимірювання до єдиного формату. Перед виконанням процедур очищення обов’язковим є резервне копіювання бази, щоб уникнути втрати інформації у разі помилкового видалення або зміни записів.

Висновки. Очищення даних є не одноразовою технічною операцією, а системним і безперервним процесом, який має бути інтегрований у загальну стратегію управління якістю даних. Його реалізація забезпечує підвищення достовірності інформації, покращення точності аналітичних звітів, скорочення ризиків помилкових рішень та економію ресурсів, адже усунення помилок на ранніх етапах завжди обходиться дешевше, ніж їх наслідки [1; 2]. Ефективне впровадження Data Cleansing створює підґрунтя для формування надійного інформаційного середовища, що відповідає сучасним вимогам цифрового урядування, прозорості й довіри. У перспективі розвиток методів очищення даних передбачає застосування технологій машинного навчання для автоматичного виявлення дефектів і прогнозування ризиків появи помилкових записів. Отже, системне очищення є стратегічним інструментом підвищення достовірності даних, який безпосередньо впливає на ефективність функціонування електронних реєстрів та прийняття управлінських рішень.

Список використаних джерел

1. Azeroual O., Saake G., Abuosba M. Data Quality Measures and Data Cleansing for Research Information Systems. *Journal of Digital Information Management.* 2018. Vol. 16(1). URL: https://dspacecris.eurocris.org/handle/11366/632 (дата звернення: 12.11.2025).

2. Using a Data Quality Framework to Clean Data Extracted from Electronic Health Records / O. Dziadkowiec et al. *Applied Clinical Informatics.* 2016. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC4933574/ (дата звернення: 12.11.2025).

3. Data Cleansing: What Is It and How It Works. Experian Data Quality (EDQ). URL: https://www.edq.com/glossary/data-cleansing/ (дата звернення: 12.11.2025).

4. Data Cleansing – Data Cleaning. DataSilk. https://datasilk.com/data-cleansing/
5. 10 Essential Data Cleansing Best Practices. Procogia (blog). URL: https://procogia.com/data-cleansing-best-practices/ (дата звернення: 12.11.2025).
6. Алексюк В. В., Січко Т. В. Дослідження особливостей програмного забезпечення для аналізу даних. *Комп'ютерні технології обробки даних:* матеріали всеукр. наук.-практ. конф., Вінниця, 2022. С. 249–251.

УДК 004.001

Алексюк В. В., здобувач 2 курсу
ОС «Магістр» спеціальності
F3 Комп'ютерні науки,
Потапова Н. А., канд. екон. наук, доцент,
доцент кафедри інформаційних технологій

ЗАСТОСУВАННЯ БАЗ ДАНИХ NOSQL У ВЕБЗАСТОСУНКАХ ДЛЯ АНАЛІТИКИ ПРОДАЖІВ

Донецький національний університет імені Василя Стуса, м. Вінниця

У сучасних умовах цифровізації бізнес-процесів електронна комерція генерує великі обсяги даних, пов'язаних із транзакціями, клієнтською активністю, товарними позиціями та маркетинговими кампаніями. Ефективна обробка та аналіз цієї інформації є критично важливими для прийняття управлінських рішень, прогнозування попиту та оптимізації асортименту [1]. Вебзастосунки, що виконують функції аналітики, повинні забезпечувати швидкий доступ до даних, гнучке структурування та інтерактивну візуалізацію результатів.

Одним із ключових компонентів таких систем є база даних. У випадку роботи з напівструктурованими даними, які можуть змінюватися з часом, традиційні реляційні СУБД часто виявляються недостатньо гнучкими. Саме тому все більшого поширення набувають NoSQL-рішення, зокрема MongoDB, які дають змогу адаптувати структуру даних до потреб аналітичної системи.

NoSQL (Not Only SQL) – це клас систем управління базами даних, які не використовують традиційну реляційну модель. MongoDB, як представник документно-орієнтованих СУБД, зберігає дані у форматі BSON (розширений JSON), що дає змогу:

– зберігати документи з довільною структурою;
– легко додавати нові поля без зміни схеми;
– використовувати вкладені об'єкти та масиви;
– виконувати складні агрегаційні запити.

Ці властивості особливо корисні для аналітичних вебзастосунків, де структура даних може змінюватися залежно від бізнес-завдань [2].

Аналітичні вебзастосунки, орієнтовані на обробку комерційної інформації, зазвичай працюють з різними типами даних: транзакціями, інформацією про

клієнтів, товарними позиціями, часовими показниками тощо. У контексті використання NoSQL баз даних структура зберігання таких даних має бути гнучкою, масштабованою та адаптивною до змін бізнес-логіки [3].

Документно-орієнтована модель дає змогу зберігати кожну одиницю даних у вигляді окремого документа, який може містити вкладені об’єкти, масиви, а також поля з довільною структурою. Це особливо корисно для систем, де дані надходять із різних джерел або мають неоднорідну природу. Наприклад, транзакції можуть містити додаткові метадані, а інформація про клієнтів – змінний набір атрибутів залежно від сегмента.

Зв’язки між сутностями реалізуються через референсні поля або вбудовані об’єкти, що дає змогу балансувати між нормалізацією та денормалізацією залежно від вимог до продуктивності. Такий підхід забезпечує ефективну обробку запитів, зменшує кількість операцій з’єднання (join) і допомагає швидко агрегувати дані для побудови аналітичних звітів.

MongoDB забезпечує низку переваг для побудови аналітичних систем:

– Горизонтальне масштабування – можливість обробки великих обсягів даних без втрати продуктивності.

– Гнучкість схеми – адаптація структури даних до змін бізнес-логіки [4].

– Інтеграція з Node.js та Vue.js – зручна взаємодія між серверною та клієнтською частинами через JSON-формат.

– Агрегаційний фреймворк – підтримка складних запитів для обчислення метрик (середній чек, виручка за регіонами, топ-продукти).

Отже, застосування NoSQL баз даних, зокрема MongoDB, у вебзастосунках для аналітики продажів є обґрунтованим рішенням, що відповідає сучасним вимогам до гнучкості, масштабованості та інтерактивності. Документно-орієнтована модель дає змогу ефективно працювати з напівструктурованими даними, забезпечуючи швидкий доступ до інформації та підтримку складних аналітичних запитів. Отримані результати можуть бути використані як теоретична основа для подальшої реалізації прототипу системи аналітики.

Список використаних джерел

1. MongoDB Analytics for Big-Data. URL: https://www.geeksforgeeks.org/mongodb/mongodb-analytics-for-big-data (дата звернення: 13.10.2025).

2. SQL чи NoSQL? Хто є хто і з чим їх їдять. URL: https://habr.com/ru/articles/955358 (дата звернення: 13.10.2025).

3. Difference between Relational database and NoSQL. URL: https://www.geeksforgeeks.org/dbms/difference-between-relational-database-and-nosql (дата звернення: 13.10.2025).

4. Choosing the Right Database for Your eCommerce Platform: SQL *vs.* NoSQL *vs.* NewSQL. URL: https://www.snow.dog/blog/choosing-the-right-database-for-your-ecommerce-platform-sql-vs.-nosql-vs.-newsql (дата звернення: 13.10.2025).

УДК 004.75

Федорович І. А., аспірант,
Осухівська Г. М., канд. техн. наук, доцент, доцент
кафедри комп'ютерних систем та мереж

ЗАСТОСУВАННЯ SPARK-RAPIDS ДЛЯ ЗДІЙСНЕННЯ GPU-ОБЧИСЛЕНЬ НА РУШІЇ APACHE SPARK

Тернопільський національний технічний університет імені Івана Пулюя, м. Тернопіль

Apache Spark – це рушій для розподіленого обчислення великих даних, який має підтримку декількох мов програмування (Python, Java, Scala) [1] та може бути запущений як на одному вузлі обробки (з паралелізацією на рівні потоків), так і в кластерному режимі (з розподілом задачі на різні вузли в режимі «драйвер – виконавці») [2]. Завдяки цим властивостям Apache Spark широко застосовується для побудови високопродуктивних систем аналізу великих даних, зокрема в задачах машинного навчання, обробки природної мови, потокової аналітики та бізнес-інтелекту.

Попри значну масштабованість Apache Spark, більшість обчислень у стандартних конфігураціях виконується на центральних процесорах (CPU). Це створює суттєві обмеження щодо продуктивності під час оброблення великих обсягів даних, особливо у випадках, де необхідно виконувати складні обчислення над великими масивами інформації. Зі зростанням обсягів даних та вимог до швидкодії класичні CPU-орієнтовані підходи недостатні для використання у застосунках, орієнтованих на інтенсивну обробку даних.

Графічні процесори (GPU) відзначаються високим рівнем паралелізму та здатністю виконувати тисячі однотипних операцій одночасно, що робить їх особливо ефективними для задач обробки великих масивів даних. Проте безпосередня інтеграція GPU в екосистему Apache Spark є нетривіальною задачею через відмінності в архітектурі обчислень, моделі пам'яті та необхідність оптимізації обміну даними між CPU та GPU. Саме ці фактори тривалий час стримували масове впровадження GPU у застосунках Spark.

Для розв'язання зазначених проблем компанією NVIDIA було розроблено бібліотеку NVIDIA RAPIDS Accelerator for Apache Spark (Spark-RAPIDS), яка надає можливість виконувати частину обчислювальних операторів Apache Spark безпосередньо на GPU за допомогою використання бібліотек RAPIDS [3]. Spark-RAPIDS реалізований у вигляді плагіна до Apache Spark, який автоматично підміняє деякі CPU-оператори на їх GPU-аналоги без необхідності суттєвої модифікації прикладного коду користувача. Це дає змогу розробникам використовувати переваги GPU-обчислень, зберігаючи знайомий інтерфейс роботи зі Spark.

Основною ідеєю Spark-RAPIDS є переведення обчислень у колонковий (columnar) формат із використанням GPU-пам'яті замість оперативної пам'яті процесора. Бібліотека забезпечує апаратно-прискорене виконання операцій фільтрації, агрегування, сортування, об'єднання таблиць (join), а також перетворення даних. Саме ці операції становлять основу більшості ETL-процедур та підготовки даних для подальшого аналізу й машинного навчання.

Застосування Spark-RAPIDS є особливо доцільним у задачах аналізу великих текстових масивів, потокової обробки інформації, обробки лог-файлів, новинних стрічок, даних із соціальних мереж та систем моніторингу. У таких застосуваннях значна частина часу витрачається саме на попередню обробку та агрегацію даних, тому перенесення цих етапів на GPU дає змогу досягти суттєвого скорочення загального часу виконання обчислювальних задач.

Перевагою Spark-RAPIDS є також те, що він допомагає масштабувати GPU-обчислення в межах кластерної інфраструктури, використовуючи кілька графічних прискорювачів на різних вузлах. Це відкриває нові можливості для побудови високопродуктивних розподілених систем аналізу даних, у яких GPU використовується не лише для машинного навчання, але й для класичних операцій обробки даних.

Водночас використання Spark-RAPIDS має і певні обмеження. Зокрема, не всі оператори Apache Spark наразі підтримуються у GPU-варіанті, що призводить до автоматичного повернення виконання окремих етапів на CPU [4]. До того ж ефективність GPU-прискорення значною мірою залежить від апаратної конфігурації, обсягу відеопам’яті та накладних витрат на передавання даних між центральною та графічною пам’яттю. У деяких випадках, особливо для невеликих обсягів даних, використання GPU може не дати суттєвого виграшу у продуктивності.

Отже, бібліотека Spark-RAPIDS є перспективним інструментом для інтеграції GPU-обчислень у середовище Apache Spark та може суттєво підвищити ефективність розподіленої обробки великих даних. Її використання є доцільним у задачах, де переважають масові однорідні обчислення та операції агрегації, зокрема в задачах потокової аналітики та обробки текстових даних. Подальші дослідження у цьому напрямі можуть бути спрямовані на експериментальну оцінку продуктивності Spark-RAPIDS у прикладних задачах аналізу природної мови та розроблення гібридних підходів, що поєднують можливості Spark-RAPIDS і користувацьких GPU-обчислювальних модулів.

Список використаних джерел

1. Comparative analysis of large data processing in Apache Spark using Java, Python and Scala / I. Borodii, I. Fedorovych, H. Osukhivska, D. Velychko, R. Butsii. *Proceedings of the 3rd International Workshop on Computer Information Technologies in Industry 4.0 (CITI 2025)* (11–12 June 2025, Ternopil, Ukraine). P. 189–198. URL: https://ceur-ws.org/Vol-4057/paper13.pdf

2. Apache Spark™ – Unified Engine for large-scale data analytics. URL: https://spark.apache.org/ (дата звернення: 27.11.2025).

3. Overview – Spark RAPIDS User Guide. URL: https://docs.nvidia.com/spark-rapids/user-guide/latest/overview.html (дата звернення: 27.11.2025).

4. Supported Operators spark-rapids. URL: https://nvidia.github.io/spark-rapids/docs/supported_ops.html (дата звернення: 27.11.2025).

УДК 001.102:004.738.5

Костенко Р. О., здобувач 2 курсу ОС «Магістр»
спеціальності F3 Комп'ютерні науки,
Веселовська Н. Р., д-р техн. наук, професор,
професор кафедри інформаційних технологій

НАУКОВА КОМУНІКАЦІЯ В УМОВАХ ЦИФРОВІЗАЦІЇ СУСПІЛЬСТВА

Донецький національний університет імені Василя Стуса, м. Вінниця

Цифровізація суспільства радикально змінила способи створення, поширення та обговорення наукового знання. Традиційна модель наукової комунікації, що базувалася на друкованих журналах, очних конференціях і бібліотечних фондах, поступається місцем цифровим платформам, які забезпечують миттєвий доступ до інформації та глобальну взаємодію дослідників [1]. Ця трансформація охоплює всі етапи наукового процесу: від пошуку літератури до публікації результатів та їх обговорення в науковому співтоваристві.

Соціальні мережі стали важливим інструментом для поширення наукових результатів та побудови професійних зв'язків. Спеціалізовані платформи, як-от ResearchGate, Academia.edu та Mendeley, об'єднують мільйони дослідників по всьому світу, даючи змогу обмінюватися публікаціями, даними та ідеями [2]. Загальні соціальні мережі, зокрема Twitter (X) та LinkedIn, також активно використовуються для науково-популяризаторської діяльності та швидкого інформування про нові дослідження. Такі платформи формують нові метрики наукового впливу: альт-метрики фіксують згадки публікацій у соціальних медіа, що доповнює традиційні бібліометричні показники.

Перевагами використання соціальних мереж у науковій комунікації є швидкість поширення інформації, можливість отримати зворотний зв'язок від широкої аудиторії та підвищення видимості досліджень. Водночас виникають ризики поверхового сприйняття складних наукових концепцій, поширення недостовірної інформації та фрагментації наукового дискурсу. Важливим аспектом є також цифрова нерівність: не всі дослідники мають однаковий доступ до цифрових платформ через технічні, мовні або інституційні обмеження.

Онлайн-конференції та вебінари стали альтернативою традиційним очним заходам, особливо актуальною після пандемії COVID-19. Цифровий формат знижує фінансові та часові витрати на участь, розширює географічне охоплення та дає змогу зберігати записи доповідей для подальшого перегляду. Платформи типу Zoom, Microsoft Teams та спеціалізовані рішення для наукових заходів забезпечують інтерактивність через чати, опитування та віртуальні сесії. Гібридні конференції, що поєднують очну та онлайн-участь, стають новим стандартом, забезпечуючи гнучкість та інклюзивність.

Електронні наукові журнали та репозиторії відкритого доступу трансформували модель наукового публікування. Рух Open Access забезпечує вільний доступ до результатів досліджень, що прискорює науковий прогрес та підвищує соціальний вплив науки. Платформи попереднього друку, як arXiv та bioRxiv, дають

змогу оприлюднювати результати до формального рецензування, що особливо важливо для швидкоплинних галузей. Електронні журнали впроваджують нові можливості: мультимедійний контент, інтерактивні візуалізації, прикріплення даних та коду до публікацій.

Цифровізація змінила формат наукових дискусій. Постпублікаційне рецензування на платформах типу PubPeer допомагає обговорювати опубліковані статті, виявляти помилки та ініціювати ретракції. Коментарі до онлайн-публікацій, наукові блоги та подкасти створюють простір для неформального обміну ідеями. Цей діалог є менш ієрархічним, порівняно з традиційним рецензуванням: молоді дослідники можуть публічно коментувати роботи визнаних експертів [3]. Водночас відкритість дискусій вимагає дотримання етичних норм та протидії токсичності в онлайн-середовищі.

Цифрові інструменти змінюють також процес колаборації: хмарні сервіси для спільної роботи над текстами, системи управління дослідницькими даними та цифрові лабораторні журнали підвищують ефективність наукових команд. Штучний інтелект впроваджується для пошуку релевантної літератури, виявлення плагіату та автоматичного реферування статей. Ці технології формують екосистему цифрової науки, де традиційні бар'єри комунікації поступово зникають.

Виклики цифровізації наукової комунікації включають питання достовірності інформації, захисту інтелектуальної власності та збереження якості наукового діалогу. Надмірна кількість публікацій ускладнює фільтрацію якісних досліджень, а тиск на швидку публікацію може знижувати ретельність перевірки результатів. Необхідність цифрової грамотності та критичного мислення стає важливою компетенцією для сучасних дослідників.

Висновки. Цифровізація наукової комунікації є незворотним процесом, що відкриває нові можливості для поширення знань та міжнародної співпраці. Соціальні мережі, онлайн-конференції та електронні журнали формують нову екосистему науки, де інформація стає доступнішою, а дискусії – відкритішими. Водночас цифрова трансформація вимагає переосмислення критеріїв якості наукової комунікації, етичних норм та механізмів забезпечення достовірності. Подальший розвиток цифрових інструментів має спрямовуватися на підвищення інклюзивності, збереження академічної доброчесності та створення балансу між швидкістю поширення інформації і ретельністю наукової верифікації.

Список використаних джерел

1. Ярошенко Т., Ярошенко О., Чуканова С. Наукова комунікація в цифрову епоху. Київ: Видавничий дім «Києво-Могилянська академія», 2024. 320 с.

2. Луговий В. І., Оржель О. Ю. Відкритість, цифровізація й оцінювання в науці. Київ: Інститут вищої освіти НАПН України, 2023. 173 с.

3. Сокур О. Л. Інформаційно-комунікаційна діяльність бібліотечної мережі НАН України в системі наукових комунікацій. Київ: НБУВ, 2020. 268 с.

СЕКЦІЯ 5
ПРИКЛАДНІ АСПЕКТИ ОБРОБКИ ДАНИХ В ІНФОРМАЦІЙНИХ СИСТЕМАХ

УДК 004.5:004.8

Оврамець І. В., здобувач 2 курсу
ОС «Магістр» спеціальності
F3 Комп'ютерні науки,
Антонов Ю. С., канд. фіз.-мат. наук, доцент,
доцент кафедри інформаційних технологій

АВТОМАТИЗОВАНА СИСТЕМА ГЕНЕРАЦІЇ ВІДПОВІДЕЙ НА ПОВІДОМЛЕННЯ З ВИКОРИСТАННЯМ DOCKER КОНТЕЙНЕРІВ

Донецький національний університет імені Василя Стуса, м. Вінниця

Вступ. У сучасних умовах цифровізації обсяг електронних комунікацій зростає експоненційно. Користувачі щодня обмінюються мільярдами повідомлень через месенджери, корпоративні чати та системи підтримки клієнтів [1]. Це створює потребу в автоматизації процесів обробки та генерації відповідей, які б враховували контекст діалогу та зберігали природність спілкування.

Сучасні технології штучного інтелекту, зокрема великі мовні моделі (LLM) та методи обробки природної мови (NLP), дають змогу створювати інтелектуальні системи, здатні аналізувати історію переписки, визначати наміри користувачів і формувати релевантні відповіді. Однак для ефективного впровадження таких систем необхідно забезпечити їх незалежність від конкретної інфраструктури, можливість швидкого розгортання та масштабування.

Контейнеризація на основі Docker стала стандартом де-факто для побудови сучасних розподілених систем. Вона забезпечує ізоляцію середовища виконання, спрощує процеси розгортання та дає змогу реалізувати мікросервісну архітектуру, в якій кожен компонент функціонує автономно.

Основний текст. Архітектура автоматизованої системи генерації відповідей побудована на принципах модульності, ізоляції компонентів та незалежності від конкретного середовища виконання. Ключовою особливістю реалізації є виділення модуля генерації відповідей в окремий Docker-контейнер, що забезпечує його портативність і можливість використання як самодостатнього мікросервісу.

Система складається з кількох функціональних блоків, кожен з яких виконує чітко визначену роль. Центральним елементом є REST API, що забезпечує взаємодію між клієнтськими застосунками та серверною частиною. API структуровано за доменними модулями: авторизація (api/login), управління навчальними даними (Learning), робота з повідомленнями (Messages) та генерація рекомендацій (Recommendations). Усі ендпоінти документовані за допомогою Swagger/OpenAPI, що значно спрощує інтеграцію та тестування системи.

Особливу увагу приділено механізмам безпеки та валідації даних. Система використовує токенову автентифікацію (JWT), яка забезпечує захищений доступ до всіх ресурсів, окрім ендпоінту входу. Це унеможливлює несанкціонований доступ до внутрішніх даних та модифікацію інформації. Додатково реалізовано багаторівневу валідацію запитів, що блокує некоректні, небезпечні або спотворені дані на етапі їх надходження до системи.

Обробка вхідних повідомлень здійснюється через два послідовні підмодулі. Перший, MessageSanitizer, відповідає за видалення чутливої інформації – номерів банківських карток, email-адрес, телефонів, IP-адрес, IBAN та інших персональних даних. Це забезпечує відповідність вимогам GDPR та захищає конфіденційність користувачів. Другий підмодуль, AnalysingModule, виконує лінгвістичну обробку тексту: нормалізацію, токенізацію, виділення ключових слів, визначення намірів користувача та побудову структурованого представлення діалогу.

Для генерації відповідей використовується готова велика мовна модель (LLM) з бібліотеки Hugging Face Transformers. Реалізовано сервіс LLMService, який інкапсулює логіку завантаження моделі та генерації тексту. Модель та токенайзер завантажуються один раз під час першого виклику, що дає змогу уникнути повторного створення ресурсоємних об'єктів. Генерація відбувається з параметрами do_sample = True, top_k = 30 та temperature = 0.7, що забезпечує баланс між креативністю та релевантністю відповідей.

Ключовою перевагою запропонованого підходу є використання Docker-контейнеризації. Система упакована у Docker-контейнер разом з усіма залежностями, конфігураційними файлами та бібліотеками, що гарантує однакову роботу у будь-якому середовищі – від локального комп'ютера до хмарних платформ. Використання Docker Compose допомагає одночасно запускати декілька контейнерів (вебсервер, базу даних PostgreSQL, модуль рекомендацій) з налаштованими зв'язками між ними.

Така архітектура забезпечує низку переваг: масштабованість через можливість запуску кількох екземплярів контейнерів, ізоляцію середовища виконання, що мінімізує конфлікти залежностей, простоту розгортання на різних платформах, включно з AWS, Azure та Google Cloud, а також можливість швидкого оновлення окремих компонентів без зупинки всієї системи.

Експериментальна перевірка системи показала здатність генерувати контекстно релевантні відповіді з дружньою тональністю. Система успішно аналізує історію переписки, враховує попередні повідомлення та формує логічні продовження діалогу. Використання відносно невеликої моделі дає змогу забезпечити прийнятну швидкість генерації за обмежених обчислювальних ресурсів, що робить рішення придатним для демонстраційних та тестових сценаріїв.

Висновки. За результатами проведеного дослідження розроблено автоматизовану систему генерації відповідей на повідомлення, реалізовану у вигляді окремого Docker-контейнера. Система поєднує сучасні технології обробки природної мови, великі мовні моделі та принципи мікросервісної архітектури.

Запропонована архітектура забезпечує модульність, масштабованість та незалежність від конкретного середовища виконання. Реалізовано REST API з повною документацією через Swagger/OpenAPI, багаторівневі механізми безпеки та валідації даних, систему видалення чутливої інформації та аналітичну обробку повідомлень.

Використання Docker-контейнеризації дає змогу швидко розгортати систему на різних платформах, легко масштабувати її у разі зростання навантаження та інтегрувати з наявними комунікаційними системами – корпоративними месенджерами, службами підтримки клієнтів або вебчатами.

Експериментальна перевірка підтвердила працездатність системи та її здатність генерувати релевантні відповіді на основі контексту діалогу. Подальші дослідження можуть бути спрямовані на інтеграцію більш потужних мовних моделей, впровадження механізмів донавчання на спеціалізованих датасетах та розробку системи автоматичного оцінювання якості згенерованих відповідей.

Список використаних джерел

1. The Effect of WhatsApp Usage on Employee Innovative Performance at the Workplace: Perspective from the Stressor – Strain – Outcome Model. URL: https://ojs.srce.hr/index.php/entrenova/article/view/30184 (дата звернення: 07.09.2025).

2. Prieur B. Traitement automatique du langage naturel avec Python: le NLP avec spaCy et NLTK. Paris: ENI, 2024. 277 p.

3. Kane S. P., Matthias K. Docker: Up & Running. Sebastopol: O'Reilly Media, 2023. 327 p.

4. Corinne C. Hugging Face Transformers: A Step-by-Step Guide to Building NLP Applications with Python. Sebastopol: O'Reilly Media, 2025. 317 p.

УДК 004.422.5

Кудирка М. Ю., здобувач 2 курсу спеціальності 122 Комп'ютерні науки,
Павлюк О. А., канд. техн. наук, старший викладач кафедри доцент, доцент кафедри інформаційних технологій

АЛГОРИТМИ ВИЗНАЧЕННЯ МАКСИМАЛЬНОЇ ПРОПУСКНОЇ ЗДАТНОСТІ В ПОТОКОВИХ МЕРЕЖАХ

Донецький національний університет імені Василя Стуса, м. Вінниця

Задача визначення максимальної пропускної здатності в потокових мережах є однією з фундаментальних оптимізаційних задач у теорії графів. Потокова мережа – це абстрактна модель, що являє собою орієнтований граф, де кожне ребро (канал) має певну пропускну здатність, а два вузли виділені як джерело та сток. Мета полягає у знаходженні максимального обсягу «потоку», який можна передати від джерела до стоку. Цей підхід широко застосовується для моделювання систем у логістиці, телекомунікаціях, енергетиці та економіці. Теоретичною основою для розв'язання цієї задачі є теорема про максимальний потік та мінімальний розріз, яка стверджує, що максимальна величина потоку дорівнює мінімальній пропускній здатності серед усіх можливих розрізів мережі [1; 2].

Основоположним методом розв'язання задачі є підхід Форда–Фалкерсона, який полягає в ітеративному нарощуванні потоку. Його ідея базується на пошуку так званих доповнюючих шляхів – маршрутів від джерела до стоку в залишковій мережі, якими ще можна направити додатковий потік [1]. Залишкова мережа динамічно відображає вільну пропускну здатність кожного каналу. Процес повторюється

доти, доки знайти такий шлях стає неможливо, що свідчить про досягнення максимального потоку.

Практичною реалізацією цього методу є алгоритм Едмондса–Карпа, який для пошуку доповнюючого шляху на кожній ітерації використовує пошук в ширину (BFS). Це гарантує, що завжди буде знайдено найкоротший за кількістю ребер шлях, що дає змогу уникнути потенційних проблем з продуктивністю базового методу. Обчислювальна складність алгоритму Едмондса–Карпа є поліноміальною і становить $O(VE^2)$, де V – кількість вершин, E – кількість ребер [1; 3].

Для підвищення ефективності був розроблений алгоритм Дініца, який за одну фазу роботи обробляє одразу групу найкоротших шляхів. На кожному етапі він будує так звану шарувату мережу, яка містить лише ребра, що є частиною найкоротших шляхів від джерела. Після цього в цій допоміжній мережі знаходиться блокуючий потік, який насичує принаймні один канал на кожному шляху від джерела до стоку. Такий підхід значно прискорює обчислення і має кращу асимптотичну складність $O(V^2E)$ [1].

Принципово іншу стратегію використовують алгоритми проштовхування-перепозначення. Вони працюють з локальними операціями, підтримуючи «передпотік», де потік, що входить у вершину, може тимчасово перевищувати вихідний. Вершини з надлишковим потоком намагаються «проштовхнути» його до сусідніх вершин, що знаходяться «нижче» за спеціальною функцією висоти. Якщо це неможливо, висота вершини збільшується (операція перепозначення). Цей підхід є одним з найефективніших на практиці [1; 4].

Практичне застосування цих алгоритмів є надзвичайно широким. У логістичних системах вони дають змогу розрахувати максимальний обсяг вантажу для транспортних коридорів. У телекомунікаціях – визначити максимальну швидкість передачі даних між двома вузлами мережі. Варіації цих методів також використовуються в комп’ютерному зорі для сегментації зображень, де мінімальний розріз графу допомагає відокремити об’єкт від фону.

Однією з основних проблем під час роботи з потоковими мережами є висока обчислювальна складність, особливо для великих та щільних графів. Продуктивність може суттєво падати, якщо обрати невідповідний алгоритм для конкретної структури мережі. Наприклад, для графів з тисячами вершин та ребер використання базового алгоритму Едмондса–Карпа може бути надто повільним, тоді як алгоритм Дініца чи методи проштовхування-перепозначення впораються із задачею значно швидше.

Отже, алгоритми пошуку максимального потоку є потужним інструментом для моделювання та оптимізації складних систем. Їх правильний вибір та застосування дають змогу вирішувати критично важливі задачі розподілу ресурсів, що є ключовим аспектом у сучасних технологіях та інженерії.

Список використаних джерел

1. Вступ до алгоритмів / Т. Г. Кормен, Ч. Е. Лейзерсон, Р. Л. Рівест, К. Стайн. 3-тє вид. Київ: К.І.С., 2019. 1288 с.

2. Нікольський Ю. В., Пасічник В. В., Щербина Ю. М. Дискретна математика: підручник. 3-тє вид. Львів: «Магнолія 2006», 2008. 432 с.

3. Креневич А. П. Алгоритми і структури даних: підручник. Київ: ВПЦ «Київський університет», 2021. 200 с.

4. Goldberg A. V., Tarjan R. E. A new approach to the maximum-flow problem. *Journal of the ACM*. 1988. Vol. 35, № 4. P. 921–940.

УДК 004.422.5

Клименко Ю. Ю., здобувач 2 курсу спеціальності 122 Комп'ютерні науки,
Потапова Н. А., канд. екон. наук, доцент, доцент кафедри інформаційних технологій

АЛГОРИТМІЧНІ АСПЕКТИ РЕАЛІЗАЦІЇ СИСТЕМ АНТИПЛАГІАТУ МЕТОДАМИ SHINGLING ТА RABIN-KARP

Донецький національний університет імені Василя Стуса, м. Вінниця

Сучасний інформаційний простір характеризується експоненційним зростанням обсягів текстового контенту. Це породжує проблему забезпечення академічної доброчесності та захисту авторських прав. Актуальність розробки ефективних алгоритмів антиплагіату зумовлена необхідністю автоматизованої перевірки робіт студентів, наукових статей, а також SEO-аналізу вебресурсів, де унікальність тексту є критичним фактором ранжування.

Класичні методи порівняння рядків, як-от відстань Левенштейна, мають квадратичну складність $O(n * m)$, що робить їх непридатними для обробки великих баз даних у реальному часі. Сучасні підходи базуються на хешуванні та теорії множин, що дає змогу досягти лінійної складності та значно підвищити швидкодію систем [1].

Мета роботи полягає у розробці та програмній реалізації гібридного алгоритму мовою C#, що поєднує високу швидкість пошуку точних підрядків та стійкість до нечітких дублікатів.

В основі запропонованого підходу лежить послідовне застосування алгоритмів Рабіна–Карпа та методу шинглів (Shingling). Алгоритм Рабіна–Карпа використовується для миттєвого виявлення прямих запозичень (цитат). Його ключовою особливістю є використання поліноміального плинного хешу (rolling hash), що дає змогу перераховувати хеш-суму вікна тексту під час зсуву на один символ за константний час $O(1)$ за формулою [1]:

$$H_{i+1} = (d * (H_i - T[i] * h) + T[i + m])(mod q), \quad (1)$$

де H_i – поточний хеш;
d – розмір алфавіту (множник зсуву);
$T[i]$ – символ, що виходить;
$h = d^{m-1}(mod\ q)$ – вага старшого розряду;
q – просте число.

Для виявлення модифікованого тексту (рерайту), де прямий пошук не дає результатів, застосовується метод шинглів. Текст піддається канонізації та розбивається на n-грами (шингли). Ступінь схожості документів A та B визначається через коефіцієнт Жаккара:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}, \tag{2}$$

де $|A \cap B|$ – кількість спільних шинглів (перетин множин),

$|A \cup B|$ – загальна кількість унікальних шинглів (об'єднання множин).

Результат $J(A,B)$ належить проміжку [0,1], де 1 означає повний збіг [2].

Програмна реалізація (рис. 1) виконана мовою C# з використанням оптимізованих структур даних. Результати тестування показали, що за значної модифікації тексту (рерайтинг) алгоритм Рабіна–Карпа фіксує наявність ключового патерну, а метод шинглів визначає залишкову схожість на рівні 20,00 %.

```
// Частина алгоритму Рабіна-Карпа (оновлення хешу)
if (i < text.Length - M)
{
    t = (d * (t - text[i] * h) + text[i + M]) % q;

    if (t < 0)
        t = t + q;
}

// Частина методу Шинглів (пошук спільних елементів)
double intersection = 0;
foreach (int hash in set1)
{
    if (set2.Contains(hash))
    {
        intersection++;
    }
}

// Розрахунок коефіцієнта схожості
double union = set1.Count + set2.Count - intersection;
double result = intersection / union;
```

Рисунок 1 – Фрагмент програмної реалізації гібридного методу

```
Текст 1 (Оригінал): ...Алгоритми пошуку плагіату є важливою час...
Текст 2 (Рерайт):   ...Алгоритми пошуку плагіату складають важл...
Патерн пошуку: систем освіти

Шинглів у T1: 12
Шинглів у T2: 12
Перетин: 4
Об'єднання: 20

Точний збіг (Rabin-Karp): True
Схожість (Shingling):     20,00%
```

Рисунок 2 – Результат роботи коду

Висновки. Розроблений гібридний алгоритм дає змогу нівелювати обмеження класичних методів рядкового зіставлення. Алгоритм Рабіна–Карпа забезпечує лінійну складність пошуку цитувань, тоді як метод шинглів гарантує стійкість системи до лексичних модифікацій.

Список використаних джерел

1. Introduction to algorithms / T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein. 4th ed. Cambridge, MA: The MIT Press, 2022. 1312 p. URL: https://books.google. com.ua/books?id=NLngYyWFl_YC&printsec=copyright&hl=uk#v=onepage&q&f=false (дата звернення: 27.11.2025).

2. Leskovec J., Rajaraman A., Ullman J. D. Mining of Massive Datasets. 3rd ed. Cambridge: Cambridge University Press, 2020. P. 75–88. URL: http://www.mmds.org/ (дата звернення: 27.11.2025).

3. Meuschke N., Gipp B. State-of-the-art in detection of academic plagiarism. *Frontiers in Computer Science*. 2024. URL: https://www.frontiersin.org (дата звернення: 27.11.2025).

УДК 004.422.5

Вершигора Р. О., здобувач
2 курсу спеціальності 122 Комп'ютерні науки,
Потапова Н. А., канд. екон. наук, доцент,
доцент кафедри інформаційних технологій

ОПТИМІЗАЦІЯ ОБРОБКИ ДАНИХ ЗА ДОПОМОГОЮ ХЕШ-СТРУКТУР У C#

Донецький національний університет імені Василя Стуса, м. Вінниця

Оптимізація обробки даних за допомогою хеш-структур у C# набуває все більшої значущості у сучасному програмуванні, враховуючи зростаючі обсяги інформації та потребу у швидкому доступі до неї. Використання хеш-структур дає змогу зменшити час пошуку, вставки та видалення елементів, що суттєво підвищує продуктивність програмних систем [1]. Ця тема є актуальною для дослідників та розробників, які прагнуть створити масштабовані та ефективні додатки.

Хеш-структури – це типи даних, які реалізують швидкий доступ до об'єктів за допомогою обчислення індексів через хеш-функції. Основний елемент – пара ключ-значення, де ключ перетворюється у числовий індекс. У C# широко використовуються три основні хеш-структури: Hashtable, Dictionary і HashSet. Hashtable підтримує зберігання необмеженого обсягу пар без суворої типізації, Dictionary є більш сучасною типізованою колекцією, що підвищує швидкість і безпеку, а HashSet [2] зберігає унікальні елементи, що зручно для фільтрації та перевірки належності даних.

Основний принцип роботи хеш-структур полягає у конвертації ключа в хеш-код за допомогою хеш-функції, який потім відображається у позицію зберігання

в масиві. Завдяки цьому доступ до елементу має амортизований час O(1), що значно швидше за традиційний послідовний пошук [3]. Хеш-функція має забезпечувати мінімальні колізії – ситуації, коли різні ключі мають однаковий індекс, для підвищення ефективності.

Практичне застосування хеш-структур у C# охоплює кешування даних, швидке визначення унікальності елементів, індексацію у базах даних і реалізацію пошукових алгоритмів [1], [3]. У базах даних вони знижують часові витрати на вибірку даних, у кешах дають змогу швидко перевіряти наявність інформації без додаткових запитів. В алгоритмах пошуку й сортування хеш-структури допомагають уникнути дублювань і оптимізують логіку доступу до елементів.

Однак хеш-структури не позбавлені проблем. Колізії, що виникають через співпадіння хеш-кодів, негативно впливають на продуктивність. Для їх усунення застосовують методи відкритого та закритого хешування, як-от лінійне зонування, квадратичне зонування та використання ланцюжків [2]. Управління пам’яттю є ще однією важливою проблемою, особливо у великих проєктах та багатопотокових середовищах. Для таких випадків в C# рекомендовано використовувати потокобезпечні структури, зокрема ConcurrentDictionary, що оптимізує доступ і синхронізацію у багатопотоковому режимі [4].

Підсумовуючи, хеш-структури є потужним інструментом для оптимізації обробки даних у C#. Вони дають змогу час доступу до інформації, ефективно працюють з великими обсягами даних і забезпечують безпечну роботу у багатопотокових додатках. Вивчення їх особливостей і методів підвищення продуктивності є важливим для розробників, що прагнуть створювати швидкісні та надійні програмні продукти.

Список використаних джерел

1. Курс «Алгоритми та структури даних на C#». URL: https://www.youtube.com/watch?v=DvIjpscPIoI (дата звернення 25.11.2025).

2. Грудзинський Ю. Є. Алгоритми та структури даних: навч. посіб. Київ: КПІ ім. І. Сікорського, 2023. URL: https://ela.kpi.ua/server/api/core/bitstreams/0db974f9-16fa-459c-9f19-fab0021222ed/content (дата звернення 25.11.2025).

3. Wengrow J. A Common-Sense Guide to Data Structures and Algorithms. 2nd edition: Level Up Your Core Programming Skills. Pragmatic Bookshelf, 2020. 508 p.

4. Microsoft Docs. Compute hash values in C#. URL: https://learn.microsoft.com/ru-ru/troubleshoot/developer/visualstudio/csharp/language-compilers/compute-hash-values (дата звернення 25.11.2025).

УДК 004.01

Хуторна Д. В., здобувачка 2 курсу
спеціальності 122 Комп'ютерні науки,
Фриз І. В., канд. фіз.-мат. наук, старший
викладач кафедри інформаційних технологій

ОСОБЛИВОСТІ ВИКОРИСТАННЯ ХЕШ-ТАБЛИЦЬ ТА ЗБАЛАНСОВАНИХ ДЕРЕВ ПОШУКУ В АЛГОРИТМІЗАЦІЇ

Донецький національний університет імені Василя Стуса, м. Вінниця

Асоціативні масиви, або словники, є базовим компонентом у сучасному програмуванні. Вони дають змогу зберігати пари «ключ-значення» та виконувати ефективний доступ до значення за заданим ключем. Найпоширенішими структурами даних для реалізації словників є хеш-таблиці та збалансовані дерева пошуку. Ці структури мають спільне функціональне призначення, однак різняться як за внутрішніми механізмами, так і за характеристиками ефективності в різних умовах. Метою цього дослідження є порівняльний аналіз переваг та недоліків хеш-таблиць і збалансованих дерев пошуку з урахуванням практичних аспектів їх застосування.

Хеш-таблиця виконує відображення ключа у певну позицію масиву за допомогою хеш-функції. За умови рівномірного розподілу хешів доступ до елементу в середньому здійснюється за сталий час $O(1)$. Саме очікувана (середня) швидкодія є головною перевагою цієї структури [1]. Однак у разі великої кількості колізій використання неякісної хеш-функції або надмірного коефіцієнта заповнення час доступу може зрости до $O(n)$. Попри високу практичну ефективність, хеш-таблиці не зберігають ключі впорядковано та не дають змоги виконувати діапазонні запити або здійснювати послідовну ітерацію у відсортованому вигляді без додаткових структур [2].

Збалансовані дерева пошуку (зокрема AVL-дерева та червоно-чорні дерева) працюють за іншим принципом: кожна операція вставки або видалення супроводжується ребалансуванням, щоб висота дерева залишалася логарифмічною відносно кількості вузлів. Завдяки цьому час виконання пошуку, вставки та видалення гарантується як $O(\log n)$ у найгіршому випадку. Дерева підтримують впорядкованість ключів, що забезпечує можливість виконання діапазонних запитів, пошуку мінімального та максимального значення, послідовної ітерації за зростанням ключів. Це надає перевагу вибору збалансованих дерев у задачах, де важлива не тільки швидкість доступу, а й структура даних.

Вибір між хеш-таблицею та збалансованим деревом тісно пов'язаний з компромісом між середньою швидкодією та гарантованою у найгіршому випадку ефективністю. Хеш-таблиці значно швидші в типовому сценарії, проте не забезпечують стабільності за несприятливого розподілу ключів. Натомість дерева стабільні за продуктивністю, але потребують більше часу на виконання базових операцій, порівняно з ідеальною поведінкою хеш-таблиці [3].

Ще однією важливою відмінністю є відношення цих структур до пам'яті. Хеш-таблиці зазвичай резервують більший масив, ніж поточна кількість еле-

ментів, що дає змогу зменшити колізії, але збільшує споживання пам'яті. Дерева працюють ефективніше з погляду використання пам'яті, оскільки щоразу збільшуються лише на один вузол без необхідності розширення масиву.

У практичних застосуваннях хеш-таблиці найчастіше використовуються у мовах програмування як реалізація словників і множин (Python dict, Java HashMap, C++ unordered_map). Вони ефективні для кешування, пошуку у таблицях символів та обробки запитів з унікальними ключами, де впорядкованість не потрібна [1]. Збалансовані дерева, навпаки, широко застосовуються у базах даних, файлових системах та індексах, де важливо підтримувати впорядковані ключі. Прикладом є B-дерева і B+-дерева у СУБД, як-от MySQL і PostgreSQL [4]. Також дерева використовують у системах, що обробляють діапазонні запити, нечіткі пошуки, лексикографічну навігацію або завдання зі збереженням порядку.

Підсумовуючи, можна зазначити: вибір між хеш-таблицею та збалансованим деревом пошуку залежить від характеру задачі. Якщо необхідна висока швидкість доступу за ключем і немає вимог до впорядкування, оптимальним вибором є хеш-таблиця. Якщо ж важливо зберігати порядок елементів або гарантувати стабільну продуктивність незалежно від вхідних даних, треба обирати збалансоване дерево. Успішне застосування тієї чи іншої структури залежить від врахування особливостей предметної області, природи вхідних даних та пріоритетів ефективності.

Список використаних джерел

1. Sedgewick R., Wayne K. Algorithms. 4th ed. Boston: Addison-Wesley, 2011. 955 p.
2. Introduction to Algorithms / T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein. 3rd ed. Cambridge: MIT Press, 2009. 1312 p.
3. Goodrich M., Tamassia R., Goldwasser M. Data Structures and Algorithms in Java. 6th ed. Wiley, 2014.
4. Silberschatz A., Korth H. F., Sudarshan S. Database System Concepts. 7th ed. New York: McGraw-Hill, 2020.

УДК 004.925.8

Родюк А. І., здобувачка 3 курсу спеціальності 111 Математика,
Комаров В. Ф., канд. техн. наук, старший викладач кафедри прикладної математики та кібербезпеки

ПОРІВНЯЛЬНИЙ АНАЛІЗ МЕТОДІВ ПОБУДОВИ ФРАКТАЛЬНИХ СТРУКТУР

Донецький національний університет імені Василя Стуса, м. Вінниця

Вступ. Фрактальна геометрія є ключовим інструментом моделювання нелінійних процесів. Актуальність теми підтверджується дослідженнями, де розглядаються фрактальні генеративні моделі [1] та оптимізація ітераційних процесів [2]. Проте більшість праць зосереджена на обчисленні розмірностей [3], залишаючи поза увагою комплексне порівняння алгоритмів побудови візуальних образів.

Мета роботи полягає у здійсненні порівняльного аналізу методів MRCM, «Хаотична гра», L-систем та клітинних автоматів для визначення специфіки їх застосування.

Постановка задачі – розглянути математичні основи методів, здійснити програмне моделювання фрактальних структур та порівняти результати за критеріями детермінованості, реалістичності й емерджентності.

Основний текст. Копіювальна машина з багаторазовим зменшенням (MRCM – Multiple Reduction Copy Machine). Метод базується на детермінованих ітераційних функціональних системах (IFS), що працюють як «копіювальна машина» зі зворотним зв'язком. Набір стискаючих відображень («лінз») визначає унікальний атрактор, до якого прагне система.

Для практичної ілюстрації методу MRCM було використано онлайн-інструмент IFS Fractal Generator [4]. На рис. 1 представлено фрактальну структуру, згенеровану шляхом ітерації системи з трьох афінних перетворень. Кожна «лінза» задає коефіцієнт стиснення та кут повороту, а отримане зображення демонструє, як складна самоподібна структура виникає з простих геометричних правил.

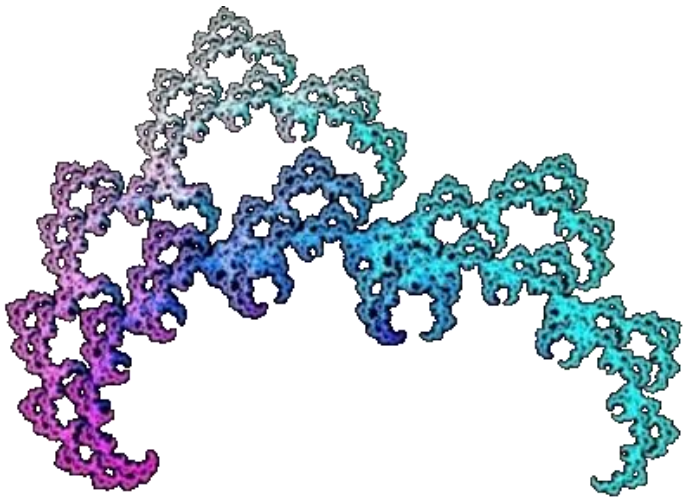


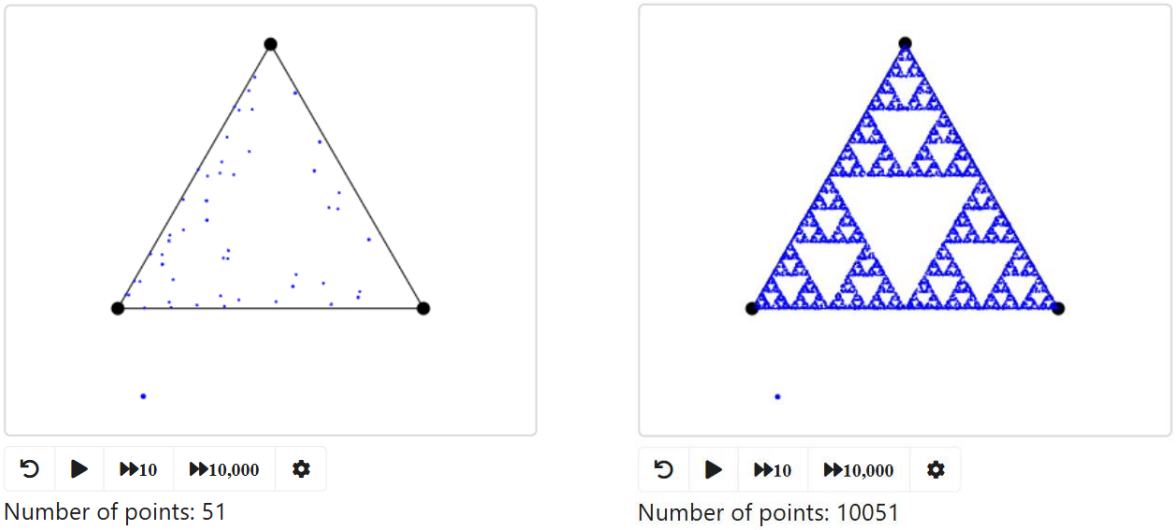
Рисунок 1 – Фрактальне дерево MRCM

Хаотична гра. Цей метод реалізує ймовірнісний підхід до побудови фракталів. Алгоритм генерує послідовність точок, де кожна наступна точка z_{k+1} є серединою відрізка між попередньою точкою z_k та випадково обраним маркером із фіксованого набору вершин. На відміну від детермінованого MRCM, тут зображення формується стохастично.

Моделювання процесу здійснено на платформі Chaos Game Simulation. На рис. 2 наведено порівняння двох етапів роботи алгоритму. Перша частина зображення демонструє початковий стан (51 ітерація), де окремі точки ілюструють випадковий характер вибору маркерів (див. рис. 2а). Друга частина показує результат після 10 000 ітерацій, де множина точок щільно заповнює простір, утворюючи чітку структуру трикутника Серпінського (див. рис. 2б). Це доводить, що хаотичний процес на мікрорівні переходить у впорядковану форму на макрорівні.

L-системи (Системи Ліденмайєра). L-системи – це машини переписування рядків, де правила продукції застосовуються паралельно до всіх символів.

Ця властивість відображає їх біологічне походження та дає змогу моделювати процеси росту. Геометрична інтерпретація здійснюється через «черепашачу графіку», що перетворює текстовий рядок у візуальний образ.



а) б)

Рисунок 2 – Візуалізація методу «Хаотична гра»: а) початковий хаотичний розподіл точок (51 ітерація); б) сформований атрактор (трикутник Серпінського) після 10 000 ітерацій

Для демонстрації біологічної природи L-систем в середовищі L-Systems Turtle Graphics було згенеровано фрактальну рослину (див. рис. 3). Використано аксіому X та правило перезапису $F \to FF$, яке імітує ріст гілок під кутом $25°$. Отримане зображення показує, як послідовна інтерпретація символьного рядка дає змогу моделювати природні об'єкти з високим рівнем самоподібності.

Клітинні автомати (Cellular Automata). Клітинні автомати – це дискретні динамічні системи, що еволюціонують у часі. Стан кожної клітинки на кроці $t + 1$ визначається станом її сусідів на кроці t за фіксованим алгоритмом. Це дає змогу спостерігати, як локальні взаємодії призводять до глобальних патернів.

Візуалізацію еволюції одновимірного автомата проведено за допомогою алгоритму Rule 30 в WolframAlpha. На рис. 4 представлено 100 кроків еволюції, де стан клітинок визначається логічною операцією над сусідами. Зображення ілюструє феномен емерджентності: з простих початкових умов виникає складна, частково хаотична структура, непередбачувана без покрокової симуляції.

Порівняння розглянутих алгоритмів демонструє, що попри візуальну схожість результатів (властивість самоподібності), механізми їх генерації кардинально відрізняються природою обчислень. Якщо метод MRCM оперує глобальними геометричними трансформаціями площини, то L-системи та клітинні автомати моделюють локальний динамічний ріст (через граматичні правила або еволюцію станів сусідів). Окреме місце посідає «Хаотична гра», яка, на відміну від детермінованих підходів MRCM та L-систем, доводить можливість виникнення суворого геометричного порядку зі стохастичного хаосу. Причому клітинні автомати (зокрема Rule 30) демонструють найвищий рівень емерджентності, де складність структури не закладена в початкових умовах, а виникає спонтанно, що робить їх унікальним інструментом для моделювання непередбачуваних природних процесів.

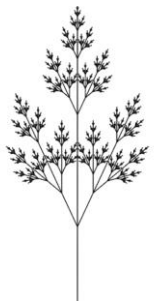


Рисунок 3 – Рослина / Дерево L-системи

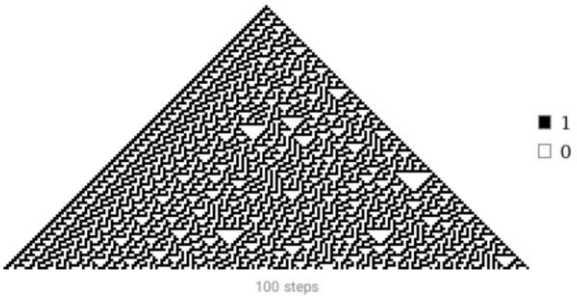


Рисунок 4 – Піраміда Rule 30

Висновки. Проведене порівняння ілюструє, що фрактальні структури можуть формуватися на основі різних математичних принципів: геометричних, ймовірнісних чи алгоритмічних. Результати моделювання свідчать про доцільність застосування методу MRCM у задачах, що вимагають точного відтворення форм, тоді як L-системи демонструють значний потенціал для імітації процесів біологічного росту. Перспективним напрямом подальших досліджень може бути розгляд гібридних підходів, зокрема для можливої процедурної генерації тривимірних (3D) об’єктів.

Список використаних джерел

1. Fractal Generative Models / T. Li et al. New York: arXiv, 2025. 13 p. (Препринт. Cornell University; 2502.17437 (cs)). URL: https://arxiv.org/abs/2502.17437 (дата звернення: 25.11.2025).

2. Fractal generation and analysis using modified fixed-point iteration / A. Tassaddiq et al. *AIMS Mathematics.* 2025. Vol. 10, № 4. P. 9462–9492.

3. A Comparative Study of Fractal Models Applied to Artificial and Natural Data / G. Silva et al. *Fractal and Fractional.* 2025. Vol. 9, № 2. P. 87. DOI: https://doi.org/10.3390/fractalfract9020087 (дата звернення: 25.11.2025).

4. SirXemic. IFS Fractal Generator: інтерактивний генератор фракталів. 2025. URL: https://github.com/SirXemic/jquery-fractals (дата звернення: 03.12.2025).

УДК: 004.738.5

Костюшин М. В., здобувач 2 курсу
ОС «Магістр» спеціальності
F3 Комп’ютерні науки,
Ніколюк П. К., д-р фіз.-мат. наук, професор
кафедри інформаційних технологій

ПОРІВНЯННЯ ПІДХОДІВ ДО ЗЛИТТЯ ДАНИХ У СИСТЕМАХ МУЛЬТИМОДАЛЬНОГО РОЗПІЗНАВАННЯ ОБ’ЄКТІВ

Донецький національний університет імені Василя Стуса, м. Вінниця

Вступ. Сучасні системи штучного інтелекту дедалі частіше працюють не з одним видом даних, а з кількома. Найпоширенішими є зображення, аудіо, текс-

тові описи, сенсорні сигнали. Об'єднання цих джерел інформації дає змогу глибше інтерпретувати складні сцени, підвищувати точність прийняття рішень і забезпечувати стійкість у варіативних умовах. В основі таких систем лежать методи ф'южну, тобто злиття ознак або проміжних представлень, що надходять із різних каналів. Попри значний прогрес у розвитку мультимодальних моделей, оптимальний спосіб поєднання даних залишається відкритим питанням, оскільки різні підходи по-різному впливають на продуктивність, узагальнювальну здатність і обчислювальну складність.

Метою цієї роботи є виконання оглядово-аналітичного порівняння підходів до ф'южну в мультимодальному розпізнаванні об'єктів та виявлення їх переваг, недоліків і сфер доцільного застосування.

Основний текст. Потреба у мультимодальних системах зростає у сферах автономного транспорту, медичної діагностики, робототехніки, відеоаналітики та людино-комп'ютерної взаємодії. Одномодальні рішення вже не забезпечують достатньої надійності, особливо в умовах шуму, часткової втрати інформації чи неоднозначності вхідних сигналів. Ф'южн дає змогу компенсувати недоліки кожної модальності та отримувати синергетичний ефект. Проте наявність кількох принципово різних підходів – раннього, пізнього, гібридного та attention-орієнтованого ф'южну створює потребу в аналітичному розумінні їх переваг і недоліків. Систематизація цих методів є важливою для коректного вибору архітектури мультимодальних систем у наукових і практичних застосуваннях.

У науковій літературі останніх років простежується активний розвиток трансформерних мультимодальних архітектур, де ф'южн реалізується переважно завдяки механізмам cross-attention та contextual alignment. Моделі на кшталт ViLT, CLIP, Flamingo та нові великі мультимодальні моделі (GPT-4o, Gemini, Claude 3.5) демонструють, що багаторівневий ф'южн дає змогу збільшити узгодженість між модальностями та покращувати семантичне розуміння.

Класичні підходи – ранній та пізній ф'южн широко описані в роботах саме з комп'ютерного зору та обробки аудіо, де зазначається, що ранній ф'южн добре працює з низькорівневими ознаками, тоді як пізній ф'южн забезпечує більшу гнучкість, але може втрачати міжмодальні залежності. У сучасних роботах з'являються також гібридні моделі, що комбінують кілька рівнів інтеграції. Однак єдиного універсального підходу не існує, і вибір методу ф'южну залежить від типу задачі, структури даних та вимог до системи [1].

Методи ф'южну зазвичай поділяють на три ключові групи: ранній (early fusion), пізній (late fusion) та гібридний (hybrid/mid-level fusion). Окремо виділяють сучасні моделі ф'южну на основі механізмів уваги (attention), що стали визначальними у трансформерних архітектурах [2].

Ранній ф'южн передбачає інтеграцію модальностей на рівні вхідних даних або низькорівневих ознак, наприклад, шляхом конкатенації каналів (RGB+D) чи сумісного подання сирих сенсорних сигналів. Такий підхід сприяє формуванню спільного латентного простору, у якому зберігаються міжмодальні кореляції й доступні комплексні залежності між модальностями. Основними перевагами є максимальне збереження інформації та раннє виявлення комплементарних ознак; основними недоліками – вимога точної синхронізації й нормалізації модальностей, чутливість до шуму та потенційне домінування нестабільної модальності.

Ранній ф’южн зазвичай ефективний у задачах, де модальності структурно узгоджені та мають близьку просторову розгортку.

Пізній ф’южн виконує інтеграцію на рівні рішень окремих одномодальних моделей через агрегування ймовірностей, голосування або метакласифікатор. Це забезпечує модульність архітектури, можливість незалежного тренування кожної модальності й спрощену інтеграцію pre-trained компонентів. Недоліком є втрата детальних міжмодальних залежностей, оскільки кореляції між ознаками не враховуються на спільному рівні представлень. Метод доцільний у сценаріях із різнорідними даними або в системах, де важливо зберегти автономність і замінність окремих модулів.

Гібридні підходи комбінують раннє злиття ознак із пізнім узгодженням рішень або проміжних представлень. Інтеграція може виконуватися на кількох абстрактних рівнях, наприклад, у просторі фіч-мапів, а фінальне рішення формується шляхом додаткового агрегування модальних прогнозів. Перевагами є кращий баланс між структурною гнучкістю та здатністю моделі виявляти складні кореляції; недоліками – підвищені обчислювальні витрати й складність оптимізації. Такі підходи демонструють високу ефективність у задачах, де потрібна точна взаємодія модальностей за збереження модульності.

Attention-орієнтований ф’южн та трансформерні підходи

Attention-механізми, зокрема cross-attention, дають змогу моделі динамічно зважувати внесок кожної модальності й вибірково підсилювати релевантні ознаки, що істотно підвищує гнучкість і точність мультимодальної інтеграції. Такі підходи забезпечують багаторівневе узгодження модальностей, підтримують масштабування на великі набори даних та формують основу сучасних мультимодальних трансформерів. Обмеженнями є висока обчислювальна вартість, необхідність великих датасетів і складність інтерпретації міжмодальних взаємодій. Attention-ф’южн є стандартом у візуально-мовних та мультимодальних LLM [3].

Підсумовуючи дані: ранній ф’южн оптимальний для структурно узгоджених модальностей із синхронізованою просторовою організацією; пізній – для систем, де критична модульність і незалежність потоків; гібридний – для задач, що вимагають балансу між точністю та гнучкістю; attention-ф’южн – для сценаріїв, де необхідно моделювати складні, контекстно-залежні кореляції на різних рівнях представлення.

Висновки. Отже, оптимальний вибір методу залежить від структури даних, вимог до точності, ресурсних обмежень і характеру взаємодії між модальностями.

Список використаних джерел

1. Baltrušaitis T., Ahuja C., Morency L.-P. Multimodal Machine Learning: A Survey and Taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2018. URL: https://www.researchgate.net/publication/317185818_Multimodal_Machine_LearningA_Survey_and_Taxonomy (дата звернення 22.11.2025).

2. Multimodal fusion for multimedia analysis: A survey / P. Atrey, M. Hossain, A. El Saddik, M. Kankanhalli. *Multimedia Systems.* 2010. URL: https://www.researchgate.net/publication/220460981_Multimodal_fusion_for_multimedia_analysis_A_survey (дата звернення: 22.11.2025).

3. Ahmad J., Del Bue A. mmFUSION: Multimodal Fusion for 3D Object Detection. 2020. URL: https://arxiv.org/abs/2311.04058 (дата звернення 22.11.2025).

УДК: 519.674:517.537.3]:004

Замотайло В. В., здобувач 2 курсу спеціальності 122 Комп'ютерні науки,
Фриз І. В., канд. фіз.-мат. наук, старший викладач кафедри інформаційних технологій

ПРОГРАМНА РЕАЛІЗАЦІЯ АПРОКСИМАЦІЇ ПЕРІОДИЧНИХ СИГНАЛІВ РЯДАМИ ФУР'Є В СИСТЕМАХ ОБРОБКИ ДАНИХ

Донецький національний університет імені Василя Стуса, м. Вінниця

Вступ. У сучасних інформаційних системах важливим завданням є обробка, стиснення та передача даних, які часто мають періодичну природу, наприклад, аудіосигнали, зображення, телеметрія. Для програмної реалізації під час обчислення значень та аналізу таких функцій використовуються методи, що спираються на розклад складного сигналу на простіші складники. Одним з інструментів, який дає змогу це робити, є гармонічний аналіз на основі рядів Фур'є.

Актуальність дослідження зумовлена необхідністю пошуку балансу між точністю відтворення сигналу і обчислювальними витратами процесора. Очевидно, що програмна генерація складних форм хвилі (наприклад, прямокутної або пилкоподібної) в цифрових синтезаторах та системах зв'язку виконується шляхом сумування лише обмеженої кількості гармонік. Отже, метою роботи є програмне моделювання та візуальний аналіз ефективності апроксимації розривного періодичного сигналу (меандру) за допомогою часткових сум ряду Фур'є.

Основний текст. Відомо, що періодична функція $f(x)$ з періодом $2l$, що задовольняє умови Діріхле, може бути розкладена у тригонометричний ряд Фур'є:

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} (a_n \cos nx + b_n \sin nx),$$

де a_n, b_n – коефіцієнти Фур'є, обчислення яких залежить від інтервалу, періоду функції, парності і непарності [1].

Для дослідження обрано один із класичних прикладів із цифрової електроніки – прямокутний сигнал, який описує ідеальні тактові імпульси в комп'ютерних системах, визначений функцією $f(x) = sign(\sin x)$ з періодом 2π, що визначена на $[0,2\pi]$, [2]. Його розклад у ряд Фур'є містить лише непарні синусні гармоніки:

$$f(x) = \frac{4}{\pi} \sum_{k=1}^{\infty} \frac{\sin\big((2k-1)x\big)}{2k-1} = \frac{4}{\pi}\left(\sin x + \frac{\sin 3x}{3} + \frac{\sin 5x}{5} + \cdots\right).$$

Для ілюстрації наближення функції рядом Фур’є використовується часткова сума функціонального ряду $S_n(x)$, яка враховує перші n доданків. Продемонстровано графічне моделювання трьох рівнів апроксимації, які наведені на рис. 1:

1) $S_1(x)$ – врахована лише основна гармоніка (фундаментальна частота);
2) $S_3(x)$ – сума перших трьох гармонік (1-ша, 3-тя, 5-та);
3) $S_{10}(x)$ – сума перших десяти гармонік (до 19-ї частоти включно).

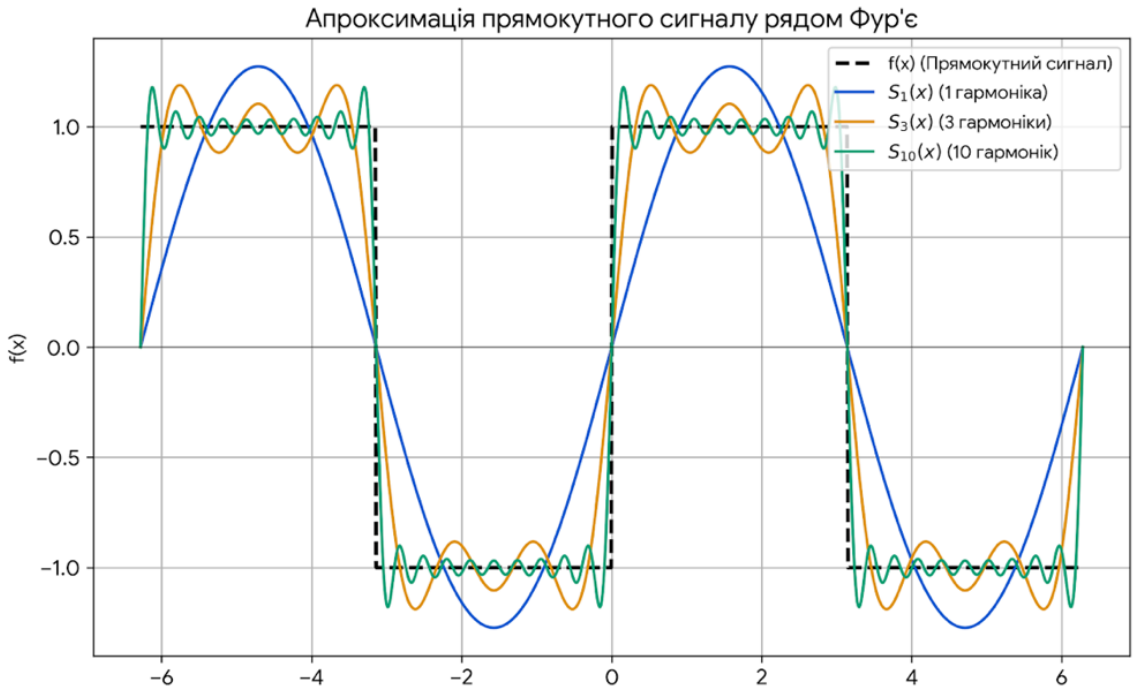


Рисунок 1 – Графічний аналіз апроксимації прямокутного сигналу рядом Фур’є

Отримані результати дають змогу зробити такі висновки:

– для $n = 1$ наближення є грубим і передає лише загальну періодичність сигналу, але не його форму;

– для $n = 3$ форма кривої починає нагадувати прямокутну, проте фронти сигналу залишаються занадто пологими для цифрових схем;

– для $n = 10$ сигнал значно наближається до ідеального меандру, спостерігається чітке формування крутих фронтів.

Водночас на графіку $S_{10}(x)$ в точках розриву (стрибків) функції чітко помітне явище Гіббса – характерні викиди амплітуди, які не зникають зі збільшенням кількості членів ряду, але локалізуються ближче до точки розриву [3].

Висновки. Здійснений аналіз демонструє, що ряди Фур’є є ефективним методом програмного синтезу складних сигналів, проте ряд Фур’є є завжди неперервною функцією. Якість апроксимації та крутизна фронтів імпульсу залежить від кількості гармонік, що враховуються в алгоритмі. Для досягнення високої точності в задачах DSP (Digital Signal Processing) необхідно використовувати достатньо високий порядок часткової суми відповідного ряду, враховуючи обчислювальну складність алгоритму, яка зростає лінійно зі збільшенням n.

Список використаних джерел

1. Чернєй М. І., Новикова Г. К., Денисенко Н. Л. Числові та функціональні ряди. Ряди Фур'є: метод. вказівки до вивчення теми з дисципліни «Вища математика». Київ: НТУУ «КПІ», 2016. 62 с.

2. Авдєєва Т. В., Качаєнко О. Б. Ряди Фур'є: практикум. Київ: НТУУ «КПІ», 2016. 88 с.

3. Бондарев В. М. Основи цифрової обробки сигналів: навч. посіб. Харків: ТОВ «Компанія СМІТ», 2007. 162 с.

УДК 004.738.1

Підруцький Д. А., здобувач 1 курсу
ОС Магістр спеціальності
F3 Комп'ютерні науки,
Потапова Н. А., канд. екон. наук, доцент,
доцент кафедри інформаційних технологій

ПРОЄКТ E-COMMERCE ВЕБЗАСТОСУНКУ З ВИКОРИСТАННЯМ REACT ТА INTEGRATION API

Донецький національний університет імені Василя Стуса, м. Вінниця

Сьогодні більшість купівель переходять в онлайн-середовище, тому створення власних e-commerce рішень є актуальним напрямом для розробників. Торговельні платформи дають можливість швидко знаходити товари, оформлювати замовлення та здійснювати оплату без необхідності фізичного відвідування магазину [1]. Для побудови подібних систем необхідні сучасні інструменти, які забезпечують швидку роботу інтерфейсу і зручність користувача. Одним із таких рішень є React, який у поєднанні з Integration API дає змогу реалізувати повноцінний динамічний вебзастосунок [2].

Обсяги онлайн-продажів постійно зростають, і це створює попит на зручні, швидкі та надійні вебсервіси. Користувачу важливо бачити товари без затримок, легко переміщатися між сторінками, швидко оформлювати замовлення. SPA-підхід на React сприяє цьому, адже сторінки не перезавантажуються повністю, а дані підтягуються через API частинами, що робить роботу сайта плавною і швидкою [3]. Саме тому тема розробки e-commerce застосунку на таких технологіях є актуальною та практично орієнтованою.

У публікаціях із веброзробки значну увагу приділено SPA-архітектурі, обробці станів у React та особливостям взаємодії з REST API. Досліджено питання оптимізації рендерингу компонентів, організації маршрутизації, а також роботи з асинхронними запитами [4]. У контексті e-commerce все частіше розглядають інтеграцію з платіжними сервісами та сторонніми API, адже це один з основних елементів онлайн-торгівлі. Це свідчить, що напрям є доволі розвиненим, але водночас відкритим для вдосконалення [5].

Основною метою розробки e-commerce вебзастосунку є створення робочого прототипу e-commerce сайта на React з підключенням Integration API, який дає змогу відображати каталог товарів, додавати їх у кошик та оформлювати замовлення без перезавантаження сторінки. Основними етапами цього процесу можна визначити:

1. Побудова структури SPA-застосунку.
2. Розробка інтерфейсу на основі компонентів React.
3. Налаштування процесів отримання та відправлення даних через API.
4. Розробка функціоналу каталогу товарів, кошика і оформлення замовлень.
5. Перевірка зручності використання та швидкості роботи застосунку.

Реалізований вебзастосунок містить основні сторінки мають динамічну візуалізацію. Дані про товари надходять через Integration API, а для переходів між розділами використовується маршрутизація без перезавантаження. Реалізовано функціонал фільтрації, перегляду детальної інформації товару, додавання в кошик, зміну кількості позицій та оформлення замовлення.

Комунікація з сервером виконується через REST-запити. Під час тестування виявлено, що застосунок працює швидко, завантаження товарів відбувається без затримок, а навігація між сторінками плавна. Це показує, що обраний підхід є ефективним для побудови онлайн-магазинів [4].

Розроблений застосунок демонструє, що React у поєднанні з Integration API є хорошим рішенням для створення e-commerce сервісів. Такий підхід забезпечує швидку роботу сайта, просту масштабованість і можливість подальшого розширення функціоналу. У майбутньому систему можна доповнити оплатою онлайн, створити адміністративну панель, додати рекомендації та історію замовлень, що зробить проєкт повноцінною торговою платформою.

Список використаних джерел

1. Дражниця С. А., Забурмеха Є. М. Електронна комерція: світові тренди та прогноз розвитку в Україні. *Вісник Хмельницького національного університету. Серія: Економічні науки.* 2018. № 6. С. 69–73.

2. Безверхій О., Куценко О. Ефективність застосування бібліотеки React. *Інформаційні технології та суспільство.* 2023. Вип. 2(4). С. 13–19.

3. Дослідження та реалізація адаптивного дизайну веб-сайту для оптимального відображення на різних пристроях з використанням бібліотеки React.js та її екосистеми. URL: http://elartu.tntu.edu.ua/handle/lib/45509 (дата звернення: 28.11.2025).

4. Розробка веб-додатку SPA з використанням фреймворку React. URL: https://openarchive.nure.ua/entities/publication/91e0cfc4-f711-441e-a638-bbdcfff965e3 (дата звернення: 28.11.2025).

5. Розробка веб-додатків та сервісів на платформі Node.js / О. П. Кошова, О. В. Ольховська, Д. С. Тацій, Ю. Ф. Олексійчук, О. О. Черненко. *Таврійський науковий вісник. Серія: Технічні науки.* 2023. Вип. 2. С. 78–89.

УДК 004.422.5

Пацула А. О., здобувач 2 курсу
спеціальності 122 Комп'ютерні науки,
Потапова Н. А., канд. екон. наук, доцент,
доцент кафедри інформаційних технологій

РЕКУРСІЯ В ПРОГРАМУВАННІ: ПОНЯТТЯ, ПРИНЦИП РОБОТИ ТА ЗАСТОСУВАННЯ

Донецький національний університет імені Василя Стуса, м. Вінниця

У програмуванні рекурсія – це метод розв'язання задачі, коли функція викликає сама себе, поступово спрощуючи задачу до базового випадку. Інакше кажучи, розв'язання задачі будується на розв'язанні її менших випадків [1; 2]. Рекурсія забезпечує природний опис багатьох алгоритмів: під час виклику функції кожен рекурсивний крок обчислює частину задачі, доки не буде досягнуто умову завершення (базовий випадок), після чого за всіма рівнями рекурсії формується результат. Рекурсивні процедури часто застосовують для обчислень, заданих рекурентними співвідношеннями [3]. Наприклад, для обчислення факторіалу або чисел Фібоначчі функція викликає сама себе з меншими параметрами, доки не досягне відомих початкових значень [3]. Проаналізуємо приклади використання рекурсивних алгоритмів:

1. Генерація всіх підмножин / комбінацій – «subsets»

```
using System;
using System.Collections.Generic;
static List<List<int>> GetSubsets(int[] nums, int index = 0) {
List<List<int>> all = new List<List<int>>();
if (index >= nums.Length) {
all.Add(new List<int>()); // порожня підмножина
return all;
}
int current = nums[index];
List<List<int>> subsetsWithoutCurrent = GetSubsets(nums, index + 1);

// для кожної підмножини без current – повернути копію + поточний елемент
foreach (var subset in subsetsWithoutCurrent) {
all.Add(new List<int>(subset)); // без current
var subsetWith = new List<int>(subset);
subsetWith.Add(current);
all.Add(subsetWith); // з current
}
return all;
}
```

Ця функція рекурсивно формує всі підмножини масиву, що корисно для задач на комбінації, перебори станів, підмножини множин тощо.

Рекурсивний обхід структури даних – дерево чи вкладені колекції. У багатьох задачах є вкладені структури (дерево, списки списків, папки файлів), і рекурсія – природне рішення для їх обходу.

```
using System;
using System.Collections.Generic;
class Node {
public int Value;
public List<Node> Children = new List<Node>();
}
static void Traverse(Node node) {
if (node == null) return;
Console.WriteLine(node.Value);
foreach (var child in node.Children) {
Traverse(child); // рекурсивний виклик для кожного нащадка
}
}
```

Цей підхід легко масштабується на будь-яку глибину вкладення, що ідеально для дерев, графів (якщо контролювати відвідані вершини), JSON / XML / DOM / вкладених колекцій тощо.

Рекурсія має свої переваги та недоліки. Вона дає змогу записувати компактний і зрозумілий код: алгоритм часто описується простіше, ніж еквівалентний ітеративний варіант [4; 5]. Завдяки рекурсії алгоритми стають більш компактними, і «рекурсивна процедура показує, що потрібно робити» [4]. За допомогою рекурсії можливо природно відобразити математичні формули та ієрархічні структури даних, полегшуючи розробку і доведення коректності. Вона широко застосовується у функціональному програмуванні і під час роботи з деревами, графами, фракталами та іншими рекурсивними структурами.

Проте кожен рекурсивний виклик потребує витрат пам’яті стеку для збереження параметрів і локальних змінних, що може призвести до швидкого заповнення стеку для глибоких викликів [6]. Наприклад, неефективне витрачання оперативної пам’яті є однією з головних проблем рекурсивних алгоритмів: кожен виклик створює новий контекст функції [7]. У багатьох випадках рекурсивні алгоритми повільніші за ітеративні. Під час роботи з рекурсивним алгоритмом може виникнути проблема переповнення стеку. Отже, під час використання рекурсії потрібно впевнено встановлювати базовий випадок і обмежувати глибину викликів.

Отже, рекурсія є методом алгоритмізації, який дає змогу розбивати задачу на підзадачі однаковим алгоритмом за дотримання вимог стислості коду та простоти моделювання складних ітеративних процесів.

Список використаних джерел

1. Recursion Algorithms. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/dsa/recursion-algorithms/ (дата звернення: 28.11.2025).

2. C#. Рекурсія. Приклади рекурсивних методів (функцій) в C#. *BestProg.net*. URL: https://www.bestprog.net/uk/2019/01/21/recursion-examples-of-recursive-methods-functions-in-c-ua/ (дата звернення: 28.11.2025).

3. Recursion in Algorithm Design: A Deep Dive. URL: https://www.numberanalytics.com/blog/recursion-in-algorithm-design-deep-dive (дата звернення: 28.11.2025).

4. Рекурсивні функції та алгоритми. URL: https://sancho.iasa.kpi.ua/ASD_HTM/Recurs01.html (дата звернення: 28.11.2025).

5. Programm.top (C#/.NET). URL: https://programm.top/uk/c-sharp/tutorial/recursion/ (дата звернення: 28.11.2025).

УДК 004.65

Бородій І. І., аспірант,
Осухівська Г. М., канд. техн. наук,
доцент, завідувач кафедри
комп'ютерних систем та мереж

ФОРМУВАННЯ ВХІДНИХ ДАНИХ ДЛЯ АНАЛІТИЧНИХ СИСТЕМ НА ОСНОВІ АРХІТЕКТУР DATA LAKEHOUSE

Тернопільський національний технічний університет імені Івана Пулюя, м. Тернопіль

Для побудови аналітичних систем даних провідною технологією є концепція Data Lakehouse, яка поєднує переваги Data Lake та Data Warehouse. Вона забезпечує гнучке зберігання структурованих і напівструктурованих типів даних, ефективне управління метаданими та оптимізоване виконання аналітичних запитів. У контексті реалізації найпоширенішими систем Lakehouse є табличні формати Apache Hudi, Apache Iceberg та Delta Lake.

Apache Iceberg – це продукт компанії Netflix, що є табличним форматом з відкритим кодом, призначений для роботи з великими обсягами аналітичних даних у сховищах типу Data Lakehouse. Apache Hudi є системою збереження даних, орієнтованого на роботу з великими обсягами інформації, що постійно змінюється. Delta Lake є технологією для забезпечення транзакційної цілісності та контролю версій у хмарних системах зберігання даних.

У сучасних роботах розглядають можливості Data Lakehouse архітектур, зокрема у межах порівняння їх продуктивності та особливостей обробки даних [1]. Ефективність використання Lakehouse-підходів у масштабованих аналітичних системах підтверджена в роботі [2]. Авторами дослідження [3] було показано, що вибір способу реалізації ETL-процесів суттєво впливає на продуктивність обробки даних у середовищі Apache Spark.

Метою дослідження є формування вхідних даних для аналітичних систем на основі архітектур Data Lakehouse, щоб здійснити вибір найоптимальнішої її реалізації за критеріями швидкості завантаження даних та ефективності збереження у дисковому просторі.

Формування вхідних наборів даних для аналітичних систем на основі архітектур Data Lakehouse здійснювалося з декількох джерел. Зокрема, вебресурс open-meteo надає погодинні показники температури атмосферного повітря за 2022–2024 рр. з прив'язкою до географічних координат. Цей структурований масив даних було завантажено у файл формату CSV обсягом 7 ГБ. Додатково у дослідженні використано CSV-файл із вебресурсу simplemaps.com, який містить перелік міст та їх координати, розміром приблизно 5 МБ. Напівструктуровані дані у форматі JSON отримано з вебресурсу iqair.com, що містить інформацію про показники

стану якості атмосферного повітря залежно від координат. Завантаження даних здійснювалося в окремі таблиці Data Lakehouse. Зокрема, таблиця weather_events отримує дані з open-meteo у вигляді файла формату CSV та поділена на партиції за полем дати (табл. 1). Таблиця world_cities включає інформацію з simplemaps.com у CSV-форматі та поділена на партиції за полем країни у форматі ISO 2 (табл. 2).

Таблиця 1 – Сутність weather_events

Назва поля	**Тип даних**	**Поле є партицією?**
lat	double	Ні
lng	double	Ні
temperature_ts	timestamp	Ні
temperature	double	Ні
temperature_dt	date	Так

Таблиця 2 – Сутність world_cities

Назва поля	**Тип даних**	**Поле є партицією?**
country	string	Ні
province	string	Ні
city	string	Ні
capital	string	Ні
population	bigint	Ні
lat	double	Ні
lng	double	Ні
country_iso2	string	Так

Таблиця air_quality_level зберігає дані, отримані з AirVisual API, у форматі JSON (табл. 3). Через складну структуру джерела таблиця містить поля типу struct, які відображають вкладені об’єкти, зокрема показники погоди та забруднення повітря. Таблиця партиціонується за полем dt, що забезпечує ефективність часових запитів.

Усі дані були завантажені в таблиці кожної системи Data Lakehouse з застосуванням процесу ETL для аналізу обсягу сформованих таблиць та оцінки ефективності використання дискового простору кожною системою.

У межах цього дослідження вилучення даних полягало у зчитуванні даних із вхідних файлів. Під час виконання етапу трансформації дані та їх структура були модифіковані відповідно до вимог дослідження за допомогою засобу обробки даних Apache Spark. Завершальний етап завантаження передбачав перенесення підготовлених даних до таблиць Data Lakehouse. Для кожної досліджуваної системи Data Lakehouse було сформовано окрему схему бази даних, у межах якої створено таблиці для завантаження даних. Кожна таблиця відповідає різним аспектам аналітики даних.

Таблиця 3 – Сутність air_quality_level

Назва поля	**Тип даних**	**Поле є партицією?**
city	string	Ні
state	string	Ні
country	string	Ні
location	struct	Ні
current	struct	Ні
lng	double	Ні
lat	double	Ні
ts	timestamp	Ні
dt	date	Так

Результатами для кожної з систем Data Lakehouse було створено окрему директорію для збереження даних із таблиць. Всередині цієї директорії розміщуються інші директорії – по одній для кожної з таблиць. Наприклад, оскільки в дослідженні використовується три таблиці, то всередині директорії будуть створені три інші піддиректорії. Організація структури таблиць для кожної з систем Lakehouse є ідентичною. Наприклад, структура для Apache Iceberg відображена на рис. 1.

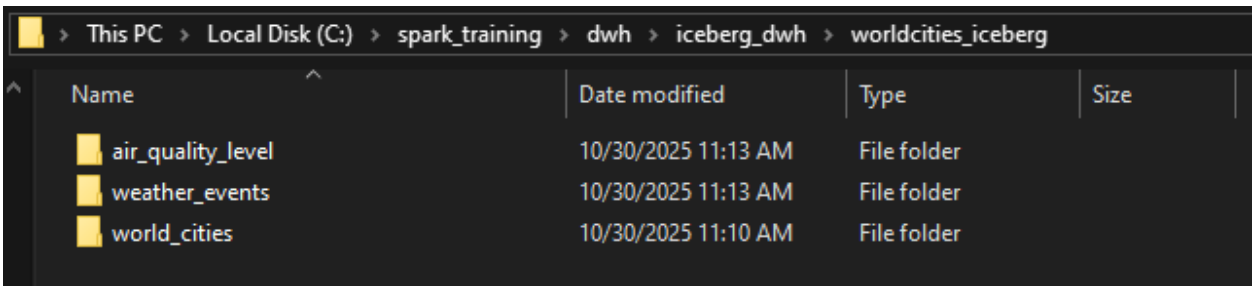


Рисунок 1 – Організація таблиць схеми Apache Iceberg

Партиції в системах Lakehouse теж організовані у вигляді файлової системи. Кожна з систем Data Lakehouse має свої особливості організації та збереження даних, які необхідно враховувати під час побудови аналітичних систем.

У цьому дослідженні описано формування вхідних даних для аналітичних систем на основі архітектур Data Lakehouse з метою порівняння продуктивності табличних форматів Apache Hudi, Apache Iceberg та Delta Lake для вибору найоптимальнішої архітектури.

Список використаних джерел

1. Analyzing and Comparing Lakehouse Storage Systems / P. Jain, P. Kraft, C. Power, T. Das, I. Stoica, M. Zaharia. *Conference on Innovative Data Systems Research (CIDR).* 2023. P. 1–12. URL: https://mail.vldb.org/cidrdb/papers/2023/p92-jain.pdf (дата звернення: 25.11.2025).

2. Begoli E., Goethert I., Knight K. A Lakehouse Architecture for the Management and Analysis of Heterogeneous Data for Biomedical Research and Mega-biobanks. *IEEE International Conference on Big Data.* 2021. P. 967–975. DOI: 10.1109/BigData 52589.2021.9671534.

3. Comparative analysis of large data processing in Apache Spark using Java, Python and Scala / I. Borodii, I. Fedorovych, H. Osukhivska, D. Velychko, R. Butsii. *Proceedings of the 3rd International Workshop on Computer Information Technologies in Industry 4.0 (CITI 2025)* (м. Тернопіль, Україна, 11–12 June 2025). P. 189–198. URL: https://ceur-ws.org/Vol-4057/paper13.pdf (дата звернення: 25.11.2025).

Наукове видання

Колектив авторів

Матеріали

VI ВСЕУКРАЇНСЬКОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ «КОМП'ЮТЕРНІ ТЕХНОЛОГІЇ ОБРОБКИ ДАНИХ»

Редактор О. А. Солдатова
Технічний редактор Т. О. Важеніна-Гопрак

Підписано до друку 13.05.2026.
Формат 60×84/16. Папір офсетний.
Друк – цифровий. Умовн. друк. арк. 7,20.
Тираж 30. Зам. 59.

Донецький національний університет імені Василя Стуса.
21021, м. Вінниця, 600-річчя, 21.
Свідоцтво про внесення суб'єкта видавничої справи
до Державного реєстру
серія ДК № 5945 від 15.01.2018