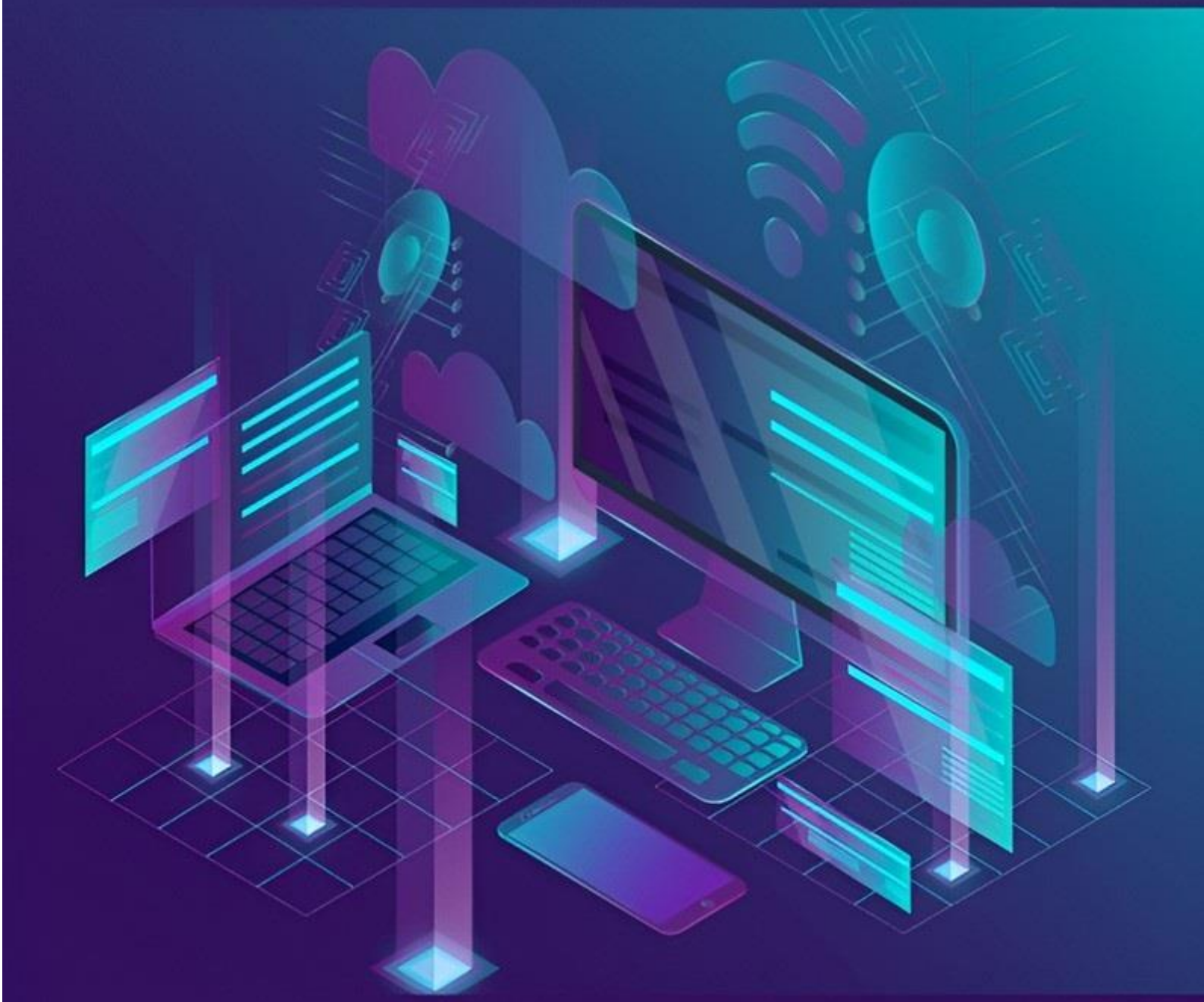
Прикладні інформаційні технології



24 травня 2024 року

м. Вінниця

Міністерство освіти і науки України
Донецький національний університет імені Василя Стуса
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»

Прикладні інформаційні технології

МАТЕРІАЛИ
V ВСЕУКРАЇНСЬКОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ
ЗДОБУВАЧІВ, АСПІРАНТІВ ТА МОЛОДИХ ВЧЕНИХ

24 травня 2024 р.

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

Вінниця
2024

Рекомендовано до друку
вченою радою факультету інформаційних і прикладних технологій
Донецького національного університету імені Василя Стуса
(протокол № 14 від 19.08.2024 р.)

Прикладні інформаційні технології: матеріали V Всеукраїнської науково-практичної конференції здобувачів, аспірантів та молодих вчених (24 травня 2024 р.). Вінниця. ДонНУ імені Василя Стуса, 2024. 205 с.

У збірнику розміщено матеріали V Всеукраїнської науково-практичної конференції здобувачів, аспірантів та молодих вчених, яка відбулася 24 травня 2024 року. У збірнику висвітлено актуальні питання, що стосуються сучасних викликів вітчизняної та світової науки, інноваційних розробок, а також перспективних напрямів досліджень. Конференція спрямована на активізацію наукового діалогу між молодими дослідниками, обмін ідеями та досвідом, а також сприяння розвитку міждисциплінарних досліджень.

Для науковців, викладачів закладів вищої освіти, а також аспірантів і студентів техніних факультетів.

ISSN (print): 2708-5821

ЗМІСТ

СЕКЦІЯ 1
ПРИКЛАДНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ В ОРГАНІЗАЦІЙНИХ, СОЦІАЛЬНО-ЕКОНОМІЧНИХ СИСТЕМАХ ТА СИСТЕМАХ ОБРОБКИ СИГНАЛІВ

СЕКЦІЯ 2
АЛГОРИТМІЗАЦІЯ ТА РОЗРОБКА
ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

СЕКЦІЯ 3
ТЕХНОЛОГІЇ ЗБОРУ, ПРЕДСТАВЛЕННЯ ОБРОБКИ, ЗБЕРІГАННЯ ІНФОРМАЦІЇ В СУЧАСНИХ ІНФОРМАЦІЙНИХ ТА КОМП'ЮТЕРНИХ СИСТЕМАХ

СЕКЦІЯ 4
ТЕХНОЛОГІЇ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ ТА ПРИЙНЯТТЯ РІШЕНЬ

СЕКЦІЯ 5
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНІ ТЕХНОЛОГІЇ

СЕКЦІЯ 1
ПРИКЛАДНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ В ОРГАНІЗАЦІЙНИХ, СОЦІАЛЬНО-ЕКОНОМІЧНИХ СИСТЕМАХ ТА СИСТЕМАХ ОБРОБКИ СИГНАЛІВ

УДК 621.311

Мишківська Я. В., *здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,*
Сеник І. О., *асистент кафедри інформаційних технологій*

АПРОКСИМАЦІЯ ДЛЯ ОПТИМІЗАЦІЇ РОБОТИ СОНЯЧНИХ ПАНЕЛЕЙ: ВИКОРИСТАННЯ ЧИСЕЛЬНИХ МЕТОДІВ ДЛЯ МАКСИМІЗАЦІЇ ВИХІДНОЇ ПОТУЖНОСТІ

Донецький національний університет імені Василя Стуса, м. Вінниця

Вступ. Сучасний світ зазнає значних викликів у сфері енергетики, спричинених різними факторами. Серед них основними є постійний ріст попиту на енергію, обмежені запаси традиційних джерел енергії та посилення екологічних вимог. У таких умовах використання відновлюваних джерел енергії, зокрема сонячної, стає ключовим рішенням для забезпечення енергетичної безпеки, економічного розвитку та зниження негативного впливу на навколишнє середовище. Україна, маючи значний потенціал сонячної енергії, повинна ефективно використовувати цей ресурс для виробництва електроенергії [3].

Актуальність дослідження. Незважаючи на високий інтерес до сонячної енергетики, існують проблеми, які гальмують її широке впровадження. Серед найбільших викликів – воєнні дії на території країни, висока вартість обладнання, що робить його недоступним для багатьох, а також недостатня ефективність виробництва енергії через неоптимальне розташування сонячних панелей та інші фактори, які часто не враховуються під час будівництва сонячних електростанцій через брак досвіду або кваліфікації.

Підвищення ефективності сонячних панелей дасть змогу ефективніше перетворювати сонячну енергію на електричну, що зробить їх більш вигідними та доступними. Для досягнення цього вчені працюють над розробкою нових методів та покращенням наявних пристроїв. Мета полягає в тому, щоб максимально ефективно використовувати потужність сонячних панелей, забезпечуючи оптимальне перетворення сонячної енергії на електричну [1].

Мета дослідження полягає в аналізі можливостей чисельних методів у підвищенні вихідної потужності сонячних панелей.

Виклад основного матеріалу. Основні шляхи підвищення ефективності сонячних панелей включають: 1) використання систем концентрації сонячного випромінювання; 2) застосування систем стеження за Сонцем; 3) використання контролерів MPPT з різними алгоритмами відстеження точки максимальної потужності (ТМП) сонячних панелей; 4) розробка технологій для виготовлення фотоелектричних модулів, спрямованих на підвищення коефіцієнта корисної дії (ККД) та зменшення терміну їх деградації. Ці методи широко використовуються в сучасному світі і мають свої переваги та недоліки, які можна знайти в літературі.

Оскільки вихідна потужність сонячних модулів залежить від факторів сонячного випромінювання і температури сонячних батарей, вимірювання сонячних модулів здійснюється в стандартних умовах (STC), а стандартні умови визнача-

ються як: якість атмосфери AM1.5, інтенсивність світла 1 000 Вт/м², температура 25 °C.

У цьому дослідженні проаналізуємо вплив температури. Відомо, що сонячні батареї (СБ) перетворюють лише частину сонячної енергії на електрику, а інша частина енергії витрачається на нагрів панелі. Це стається через те, що підвищена температура поряд із сонячною панеллю зменшує ширину забороненої зони напівпровідника, і струм насичення зростає через менший розмір енергії, необхідний для перетворення електронно-діркових пар [2].

Проте під час цього струм короткого замикання слабко збільшується, а напруга холостого ходу зменшується, що знижує вихідну потужність панелі. Вплив температури на вихідну потужність СБ можна описати виразом:

$$P_{\text{C}} = \text{P}_0\,(1 + \beta\Delta t),$$

де P_{C} – потужність сонячної батареї, Вт;

P_0 – потужність СБ при 25 °C, Вт;

β – температурний коефіцієнт потужності, °C^{-1};

Δt – зміна температури, °C.

Температурний коефіцієнт змінює вихідну потужність у межах від –0,2 % до –0,5 % під час зміни на 1 °C. Температурний коефіцієнт потужності β для монокристалічного кремнію дорівнює 0,4 %/°C. Це означає, що вихідна потужність зменшується на 0,4 % за кожного градуса підвищення температури поза робочим діапазоном для СБ.

Існує три способи відведення тепла від нагрітого об’єкта: конвекція, теплопровідність і випромінювання. Так, теплопровідність використовується за наявності різниці температур між об’єктом, наприклад, сонячною батареєю, і іншим об’єктом, включно з повітрям навколо модуля. Здатність сонячної батареї передавати тепло іншому об’єкту характеризується тепловим опором матеріалів, з яких складається сонячна батарея загалом.

Відведення тепла від сонячної батареї здійснюється за допомогою теплопровідності, зокрема за допомогою елементів Пельтьє, які розміщуються з холодного боку зворотної сторони сонячної панелі. Математична модель для сонячного модуля може бути побудована на основі рівнянь. Фотострум сонячного елемента розраховується за формулою:

$$I_{\text{ф}} = \left(I_{\text{с.н.}} + k_i \cdot (T - T_{\text{н}})\right)\frac{G}{G_{\text{н}}},$$

де $I_{\text{ф}}$ – фотострум;

$I_{\text{с.н.}}$ – фотострум сонячного елемента за температури 25 °C і освітленості від Сонця G = 1 000 Вт/м²;

k_i – температурний коефіцієнт за струмом, % / °C;

T – температура навколишнього середовища, °C;

$T_{\text{н}}$ – номінальна температура (25 °C);

G – поточна освітленість від Сонця;

$G_{\text{н}}$ – номінальна сонячна освітленість (1 000 Вт/м²).

Формула струму насичення:

$$I_{\text{д}} = I_{\text{д.н.}} \left(\frac{T}{T_{\text{н}}}\right)^2 \cdot \exp\left(\frac{q \cdot \text{E}_{g0} \cdot \left(\frac{1}{T_{\text{н}}} - \frac{1}{T}\right)}{n \cdot K}\right),$$

де $I_{\text{д}}$ – струм насичення;

q = 1,6021 10^{-19} Кл – заряд електрона;

E_{g0} = 1,2 еВ – ширина забороненої зони;

n = 1,3 – параметр сумісності з реальними характеристиками сонячних елементів;

K – стала Больцмана (К = 1,3806503 · 10^{-23} Дж / К);

T – температура навколишнього середовища, °С.

Температурний потенціал сонячної батареї розраховується за формулою ($V_{t\text{н}}$, °С):

$$V_{t\text{н}} = \frac{N_s + K \cdot T}{R_{\text{ш}}},$$

де $V_{t\text{н}}$ – температурний потенціал сонячної батареї, °С;

N_s – кількість сонячних елементів, з'єднаних послідовно.

Шунтуючий струм $I_{\text{ш}}$:

$$I_{\text{ш}} = \frac{U_{mp} + I_{mp} \cdot R_{\text{п}}}{R_{\text{ш}}},$$

де U_{mp} – напруга при максимальній потужності;

I_{mp} – струм при максимальній потужності;

$R_{\text{п}}$ – послідовний опір сонячного елемента (0,221 Ом);

$R_{\text{ш}}$ – паралельний опір сонячного елемента (415,405 Ом).

Вихідний струм I (струм сонячної панелі на моделі):

$$I = I_{\text{ф}} - I_{\text{д}} \cdot \left(\exp\left(\frac{q \cdot \left(U_{mp} + I_{mp} \cdot R_s\right)}{n \cdot N_s \cdot K \cdot T}\right) - 1\right) - I_{\text{ш}}.$$

Висновки. Проведені розрахунки дають змогу припустити, що використовуючи термоелектричні елементи, можна знизити температуру модуля сонячних батарей із сумарною потужністю 1 кВт на 10 °С і збільшити потужність, що видається модулем сонячних батарей, на 110 Вт.

Список використаних джерел

1. Кирисов І. Г., Михайлов Б. К., Лосенко Є. В. Вплив затінення та пошкоджень сонячних батарей на їх параметри. *Вчені записки ТНУ імені В. І. Вернадського. Серія: Технічні науки*. 2023. Том 34(73), № 1. С. 180–185. DOI: 10.32782/2663-5941/2023.1/27.
2. Мельник Л. Г., Маценко О. І., Терещенко С. В. Наукове обґрунтування підвищення техніко-економічної ефективності використання сонячної енергії. *Механізм регулювання економіки*. 2020. № 2. DOI: 10.21272/mer.2020.88.10.
3. Ричка Р. Ю. Оптимізація розташування сонячних панелей для досягнення максимальної виробничої потужності. *Вісник східноукраїнського національного університету імені Володимира Даля*. 2024. № 1(281). С. 76–84. DOI: 10.33216/1998-7927-2024-281-1-76-84.

УДК 330.131.7

Радзіховська А. О., *здобувачка 4 курсу спеціальності 122 Комп'ютерні науки,*
Антонов Ю. С., *канд. фіз.-мат. наук, доцент, доцент кафедри інформаційних технологій*

РИЗИК-МЕНЕДЖМЕНТ ПІД ЧАС РОЗРОБКИ ДОДАТКА ДЛЯ ВИБОРУ НАУКОВОГО КЕРІВНИКА

Донецький національний університет імені Василя Стуса, м. Вінниця

Сьогодні існує велика кількість спроб автоматизації різноманітних процесів, що виникають під час навчання [1–**Помилка! Джерело посилання не знайдено.**]. Під час розробки та впровадження проєкту з автоматизації процесу обрання наукового керівника [4–**Помилка! Джерело посилання не знайдено.**] виникають численні ризики, які можуть вплинути на успішність та ефективність вибору. Ризик-менеджмент – один із напрямів сучасного менеджменту, що вивчає проблеми управління ризиками, які виникають у діяльності самостійної господарської організації [5]. Розуміння цих ризиків є ключовим для розробки стратегій їх управління та мінімізації можливих негативних наслідків. Ефективний процес управління ризиком не може бути сукупністю фрагментарних дій, оскільки він повинен бути сформованим у комплекс дій, який є частиною загального управління бізнесом [**Помилка! Джерело посилання не знайдено.**].

Реєстр ризиків складається з декількох основних пунктів. Ось розподіл кожного критерію з реєстру ризиків:

- номер ризику;
- опис ризику;
- ймовірність настання. Вимірюється за шкалою «дуже ймовірно», «ймовірно», «малоймовірно».
- вплив ризику на результат;
- загальний рівень ризику;
- на який параметр проєкту впливає;
- наслідки;
- стратегія реагування;
- план дій;
- вплив.

У табл. 1 наведено частину реєстру ризиків, який було пропрацьовано. Також в аналізі реєстру ризиків було розглянуто три основні аспекти: наслідки, стратегії реагування та плани дій.

Таблиця 1 – Реєстр ризиків проєкту обрання наукового керівника

№	Опис ризику	Ймовірність настання	Вплив ризику на результат	Загальний рівень ризику	На який параметр проєкту впливає	Вплив
1	Низька зацікавленість з боку здобувачів	3	2	6	Залучення	Середній
2	Нестача ресурсів	2	3	6	Розробка	Середній

№	Опис ризику	Ймовірність настання	Вплив ризику на результат	Загальний рівень ризику	На який параметр проєкту впливає	Вплив
3	Недостатньо кваліфікована команда розробників	2	4	8	Розробка	Високий
4	Незлагоджена робота в команді	2	3	6	Розробка	Середній
5	Недооцінка необхідності документування коду та процесів розробки	2	4	8	Розробка	Середній
6	Зміна пріоритетів у розробників	3	2	6	Терміни	Високий

Усі ці параметри наведено нижче:

1. Низька зацікавленість з боку здобувачів:
   1.1. Наслідки: недосягнення цільової кількості користувачів.
   1.2. Стратегія реагування: проведення інформаційної кампанії, розширення функціоналу.
   1.3. План дій: розробка маркетингової стратегії, додавання нових можливостей.
2. Нестача ресурсів:
   2.1. Наслідки: затримки в розробці.
   2.2. Стратегія реагування: пошук додаткової робочої сили серед здобувачів, оптимізація ресурсів.
   2.3. План дій: пріоритизація задач.
3. Недостатньо кваліфікована команда розробників:
   3.1. Наслідки: недоліки в роботі додатка.
   3.2. Стратегія реагування: залучення менторів, розширення команди.
   3.3. План дій: пошук досвідчених фахівців, онлайн-курси.
4. Незлагоджена робота в команді:
   4.1. Наслідки: затримки в розробці.
   4.2. Стратегія реагування: застосування методологій Agile, комунікація.
   4.3. План дій: скрам, щоденні мітинги.
5. Недооцінка необхідності документування коду та процесів розробки:
   5.1. Наслідки: збільшення складності управління кодом та розробкою, можливість втрати даних та збоїв.
   5.2. Стратегія реагування: систематичне документування, використання кращих практик управління версіями.
   5.3. План дій: встановлення стандартів документування, регулярна перевірка та оновлення документації.
6. Зміна пріоритетів у розробників:
   6.1. Наслідки: затримка випуску проєкту.
   6.2. Стратегія реагування: створення чіткого планування, за потреби замороження проекту.
   6.3. План дій: Розробка календарного плану, за потреби пауза проєкту.

Застосування ефективних стратегій управління ризиками є ключовим фактором успішності проєкту. Наприклад, відповідне планування, розробка альтерна-

тивних сценаріїв, регулярний моніторинг та аналіз ризиків дають змогу забезпечити реагування на негативні події та мінімізувати їх вплив на проєкт.

Отже, управління ризиками вимагає системного підходу та уважного аналізу ситуації. Це процес, що включає ідентифікацію, оцінку, управління та моніторинг потенційних загроз для проєкту. Під час ідентифікації ризиків команда проєкту визначає можливі проблеми, їх ймовірність та вплив на результати проєкту. Далі проводиться оцінка кожного ризику, де враховуються його потенційні наслідки та ймовірність виникнення.

Управління ризиками дає змогу забезпечити стабільність та успішне завершення проєкту, оскільки ретельне планування й ефективне управління потенційними загрозами забезпечують зниження ймовірності негативних наслідків та збільшують можливості досягнення поставлених цілей. В результаті команда проєкту може працювати більш ефективно, забезпечуючи успішне виконання завдань та досягнення успіху.

Список використаних джерел

1. Антонов Ю. С. Оцінка повноти відповідей в автоматизованих системах контролю знань. *Наукові праці Донецького національного технічного університету. Сер.: Інформатика, кібернетика та обчислювальна техніка*. 2012. № 15. С. 113–117.
2. Антонов Ю. С. Комп'ютерні системи тестування на основі технології трирівневих баз даних. *Інформаційні технології і засоби навчання*. 2008. Т. 6, № 2. URL: https://journal.iitta.gov.ua/index.php/itlt/article/view/133 (дата звертання: 21.04.2021).
3. Антонов Ю. С., Мулярчук О. П. Особливості розробки підсистем обліку академічної успішності здобувачів. *Матеріали наукової конференції професорсько-викладацького складу, наукових працівників і здобувачів наукового ступеня за підсумками науково-дослідної роботи за період 2017–2018* (16–17 травня 2019 р.): у 2-х томах. Вінниця: Донецький національний університет імені Василя Стуса, 2019. Том 2. С. 106–107.
4. Антонов Ю. С., Мазурук О. В. Особливості розробки підсистеми обрання наукового керівника. *Наукові праці ДонНТУ Серія «Інформатика, кібернетика та обчислювальна техніка»*. 2022, № 2(35); 2023, № 1(36). С. 38–46.
5. Боровик М. В. Ризик-менеджмент. Харків: ХНУМГ ім. О. М. Бекетова, 2018. 67 с.

УДК 004.05

Гуменюк К. В., *здобувачка 4 курсу спеціальності 122 Комп'ютерні науки,*
Січко Т. В., *канд. техн. наук, доцент кафедри інформаційних технологій*

ПЕРЕДОВІ МЕТОДИ АНІМАЦІЇ ВЕБІНТЕРФЕЙСІВ: ВПЛИВ НА КОРИСТУВАЦЬКИЙ ДОСВІД

Донецький національний університет імені Василя Стуса

З поширенням вебтехнологій та зростанням конкуренції в онлайн-середовищі отримання ефективного користувацького досвіду стає ключовим завданням для вебдизайнерів. Анімація вебінтерфейсів є неодмінним складником сучасного дизайну. Новітні вебсайти та додатки активно використовують передові методи

анімації, які значно впливають на сприйняття та взаємодію користувачів з інтерфейсом.

Анімація вебінтерфейсів виконує багатофункціональну роль у покращенні користувацького досвіду. Вона візуально посилює інформацію, полегшує розуміння складних концепцій, покращує взаємодію користувача з сайтом та навіть збільшує конверсію. Анімація додає ще один «вимір» до дизайнерської розробки й «спілкується» на іншому рівні, ніж інструменти дизайну, як-от шрифт або колір. Людині властиво надавати значення тому, що рухається, спираючись на знання фізики реального світу або уявляючи анімовані об'єкти антропоморфними [1].

Передові методи анімації вебінтерфейсів містять широкий спектр технік, прийомів та підходів, які використовуються для створення анімації на вебсайтах та у вебдодатках з метою покращення користувацького досвіду. Розглянемо деякі з них. Одною з технік є мікроанімація [2] – це дрібні, невеликі анімаційні ефекти, які відбуваються під час взаємодії користувача з елементами вебінтерфейсу. Вони допомагають створити відчуття взаємодії та залученості, підвищуючи увагу користувача до деталей. Мікроанімація може використовуватися для підкреслення важливості певного елементу, наприклад, анімована зміна кольору кнопки під час наведення курсора на неї.

Паралакс-ефект [3] – це ефект, під час якого об'єкти на передньому плані рухаються швидше, ніж об'єкти на задньому плані, коли користувач прокручує вебсторінку або рухає мишкою. Це створює враження глибини та тривимірності і може покращити візуальний досвід користувача. Він може бути використаний на різних елементах вебсторінки, як-от фонові зображення, банери, відео та інші графічні елементи. Паралакс-ефекти особливо ефективні на сторінках із довгими скролами, вони створюють враження глибини та руху, що привертає увагу користувача та збільшує залученість до контенту.

Деякі вебсайти використовують анімацію для зменшення відчуття очікування під час завантаження контенту. Наприклад, анімований індикатор завантаження може зробити очікування менш тягучим та сприяти позитивному сприйняттю користувачем процесу. Ця стратегія особливо ефективна в умовах повільного інтернет-з'єднання або великої кількості контенту на сторінці. Замість безкінечного чекання користувачі можуть бути зайняті спостереженням за анімацією, що дає їм відчуття активності та продовжує взаємодію з сайтом. Тривіальна анімація також може використовуватися для підсвічування важливих елементів на сторінці або для підказок користувачам щодо дій, які вони можуть виконати. Наприклад, анімована стрілка, яка показує напрямок прокрутки сторінки або рухається до кнопки «Купити», може привернути увагу користувача та спонукати його до дії.

Один із способів використання анімації для покращення користувацького досвіду – це використання масштабованих векторних зображень (SVG). SVG дає змогу створювати складні графічні об'єкти, які можна анімувати без втрати якості та роздільної здатності. Анімація SVG [4] може бути використана для створення цікавих, інтерактивних та естетично привабливих інтерфейсів, які залучають та утримують увагу користувачів.

WebGL [5] – це технологія, що дає змогу використовувати графічний адаптер для створення 3D-графіки без використання плагінів. За допомогою WebGL можна створювати вражаючі 3D-сцени та об’єкти, які забезпечують реалістичний вигляд та поведінку. Анімація WebGL може значно покращити користувацький досвід, додаючи реалізму та інтерактивності до вебінтерфейсів. За допомогою цієї технології дизайнери мають можливість створювати інноваційні вебсайти та додатки.

Отже, передові методи анімації вебінтерфейсів мають великий потенціал у покращенні користувацького досвіду, забезпечуючи привабливіші, зручніші та ефективніші вебдодатки та сайти. Однак важливо пам’ятати, що анімація повинна бути використана обережно і не заважати користувачам. Її надмірне використання може викликати відчуття роздратування та втрати продуктивності. Тому передові методи використання анімації вимагають збалансованого підходу та уваги до потреб і відгуків користувачів.

Список використаних джерел

1. Head V. Designing Interface Animation: Improving the User Experience Through Animation, Rosenfeld Media. 2016. С. 24–25.
2. Dordevic B. Micro Animations: Why Are They Crucial And How To Use Them Properly? *Alpha Efficiency.* URL: https://alphaefficiency.com/micro-animations
3. Ryzha K. Parallax Effects: Best Practices & Examples, Crocoblock. URL: https://crocoblock.com/blog/parallax-effects-best-practices-and-examples/
4. Drasner S. SVG Animations: From Common UX Implementations to Complex Responsive Animation. *O’Reilly Media.* 2017. С. 17–20.
5. Cozzi P. WebGL Insights, CRC Press, 2015. С. 13–15.

УДК 519.6

Кохан Д. Ю., *здобувач 2 курсу спеціальності 122 Комп’ютерні науки,*
науковий керівник:
Римар П. В., *старший викладач кафедри інформаційних технологій*

ПОНЯТТЯ ТА ВИДИ ПОХИБОК У МЕТОДАХ ОБЧИСЛЕНЬ

Донецький національний університет імені Василя Стуса, м. Вінниця

Вступ. У сучасному світі точність вимірювань відіграє критично важливу роль у різних сферах діяльності. Від інженерії до медицини, від економіки до наукових досліджень – точні вимірювання є основою для прийняття рішень та розвитку технологій. Похибки вимірювань, які виникають під час обчислювальних процесів, можуть мати значний вплив на результати роботи та їх подальше застосування.

Похибки вимірювань – це відхилення отриманих значень від справжніх або еталонних. Вони виникають через різні причини, включно з недосконалістю вимірювальних приладів, впливом зовнішніх факторів та людським фактором. Існу-

ють різні види похибок, як-от похибки розв'язку, похибки обчислень та похибки наближень.

Розглянемо більш детально кожен з цих видів похибок, щоб зрозуміти їх природу та вплив на результати обчислень.

Похибки розв'язку. Похибки розв'язку виникають під час визначення точного розв'язку математичної задачі. Вони пов'язані з різницею між точним аналітичним розв'язком і наближеним чисельним розв'язком. Ці похибки можна виразити за допомогою такої формули:

$$\Delta = x_{\text{точне}} - x_{\text{наближене}},$$

де $x_{\text{точне}}$ – точне значення, а $x_{\text{наближене}}$ – наближене значення. Похибки розв'язку важливі, наприклад, під час моделювання фізичних процесів, де невелика похибка може призвести до значних відхилень у результатах.

Похибки обчислень. Похибки обчислень виникають через обмежену точність арифметичних операцій та використовуваних чисельних методів. Вони можуть бути спричинені обмеженою кількістю цифр у поданні чисел (наприклад, під час роботи з плаваючою комою). Формула для визначення похибки обчислень така:

$$\varepsilon = \frac{|x_{\text{точне}} - x_{\text{обсислене}}|}{|x_{\text{точне}}|},$$

де $x_{\text{точне}}$ – точне значення, а $x_{\text{обчислене}}$ – обчислене значення. Похибки обчислень критичні в комп'ютерних науках та інженерних розрахунках, де висока точність є обов'язковою.

Похибки наближень. Похибки наближень виникають під час використання наближених методів для розв'язання складних задач, де точний аналітичний розв'язок важко або неможливо отримати. Нехай x – точне значення деякої величини, x^*– її відоме наближене значення. Абсолютною похибкою числа x^* називається величина $\Delta(x^*)$, що задовольняє умову [1]:

$$\Delta x^* = |x - x^*|.$$

У цьому випадку число x можна подати у вигляді:

$$x = x^* \pm \Delta(x^*).$$

Відносною похибкою числа x^* називається величина δ, що задовольняє умову:

$$\delta = \frac{\Delta x^*}{|x|}.$$

Для оцінки похибок використовуються різні методи, зокрема статистичні та аналітичні.

Статистичні методи включають у себе використання статистичних інструментів, як-от стандартне відхилення, довірчі інтервали та регресійний аналіз для оцінки випадкових похибок [2].

Аналітичні методи полягають у використанні математичних моделей для визначення систематичних похибок та їх впливу на результати вимірювань [3].

Правильне розуміння та врахування похибок у професійній діяльності дає змогу підвищити точність результатів, знизити ризики та оптимізувати процеси.

Професіонали різних галузей повинні враховувати різні види похибок у своїй роботі для забезпечення найвищої якості та надійності виконуваних завдань.

Висновки. Точність вимірювань є основою для наукових досліджень та практичної діяльності у багатьох сферах. Розуміння та мінімізація похибок дає змогу отримувати достовірні дані, що сприяє прийняттю обґрунтованих рішень та забезпеченню безпеки. Визначення та аналіз похибок розв'язання, похибок обчислень та похибок наближень допомагають підвищити точність вимірювань та мінімізувати їх вплив на отримані результати.

Список використаних джерел

1. Чисельні методи: навчальний посібник / Л. О. Волонтир, О. В. Зелінська, Н. А. Потапова, І. А. Чіков. Вінниця: ВНАУ. 2020 322 с.
2. Баховський П. Ф. 2. Похибки вимірювання. Лекція 1. Основні визначення. *Робоча програма з дисципліни «Монтаж та вимірювання пристроїв зв'язку» для студентів спеціальності 172 – «Телекомунікації та радіотехніка». Elib.lntu.* 2019. URL: http://surl.li/twpae (дата звернення: 16.05.2024).
3. Москвіна С. М. Лекція 1. Елементи теорії похибок. *Комп'ютерні методи дослідження та аналіз даних: навчальний посібник. Київ: ВНТУ, 2010. Web.posibnyky.* URL: http://surl.li/twpak (дата звернення: 18.05.2024).

УДК 004.09

Швець Х. І., *здобувачка 4 курсу спеціальності 122 Комп'ютерні науки,*
Римар П. В., *старший викладач кафедри інформаційних технологій*

ВИКОРИСТАННЯ FLUX-АРХІТЕКТУРИ В СУЧАСНИХ ВЕБЗАСТОСУНКАХ

Донецький національний університет імені Василя Стуса, м. Вінниця

У сучасному світі розробка вебзастосунків вимагає інноваційних підходів для забезпечення високої продуктивності, масштабованості та зручності в управлінні станом додатків. Однією з таких інновацій є Flux-архітектура, яка надає ефективні інструменти для управління потоком даних та станом додатка.

Flux-архітектура була розроблена для подолання обмежень традиційних підходів до розробки, пропонуючи односторонній потік даних, який значно полегшує відстеження змін стану та робить систему передбачуваною і керованою. У цій роботі ми розглянемо основні принципи Flux-архітектури, її переваги та значення для сучасних вебзастосунків.

Результати дослідження показують, що використання Flux-архітектури в сучасних вебзастосунках має кілька ключових переваг. По-перше, односторонній потік даних у Flux значно спрощує управління станом додатка, що приводить до зменшення складності коду і полегшує процес відлагодження та підтримки. По-друге, розробники зазначають покращену передбачуваність поведінки системи, що дає змогу легше виявляти і виправляти помилки.

Дослідження також виявило, що Flux-архітектура сприяє підвищенню продуктивності розробки завдяки чіткій структурі та модульності, що полегшує розподіл завдань у команді розробників і покращує масштабованість проєктів.

До того ж результати показують, що застосування Flux дає змогу забезпечити стабільнішу і надійнішу роботу вебзастосунків у довгостроковій перспективі, оскільки уніфікований підхід до управління станом зменшує ймовірність виникнення непередбачуваних поведінкових проблем.

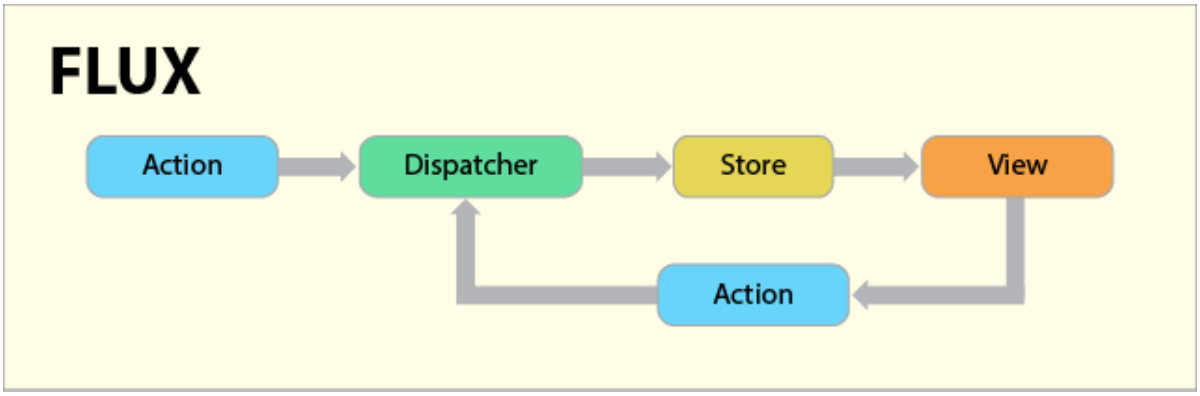


Рис. 1. Схема Flux-архітектури

У висновках до цього дослідження встановлено, що Flux-архітектура має значні переваги, порівняно з традиційними підходами до розробки вебзастосунків. Односторонній потік даних, який забезпечує Flux, спрощує управління станом додатка, зменшує складність коду та підвищує передбачуваність системи. Використання Flux-архітектури сприяє підвищенню продуктивності розробки завдяки її чіткій структурі та модульності, що дає змогу розробникам більш ефективно розподіляти завдання, особливо у командній роботі та великих проєктах.

До того ж Flux забезпечує стабільнішу та надійнішу роботу вебзастосунків у довгостроковій перспективі, оскільки уніфікований підхід до управління станом зменшує ймовірність виникнення непередбачуваних проблем. Дослідження підтвердило важливість використання Flux-архітектури для розробки сучасних вебзастосунків, які відповідають вимогам цифрового середовища.

Список використаних джерел

1. In-Depth Overview. Flux / by Yangshun. Mar. 21, 2023. URL: http://surl.li/udpgr (дата звернення: 01.05.2024).

2. Weerakoon P. An introduction to Flux architecture: Explain the high-level overview, benefits, and how it works. Feb. 3, 2023. URL: http://surl.li/udphn (дата звернення: 05.05.2024).

УДК: 004.9

Дорофєєв Є. О., *здобувач 2 курсу спеціальності 122 Комп'ютерні науки,*
науковий керівник:
Потапова Н. А., *канд. екон. наук, доцент, доцент кафедри інформаційних технологій*

ПОРІВННЯ МЕТОДІВ ДИХТОМІЇ ТА ІТЕРАЦІЇ

Донецький національний університет імені Василя Стуса, м. Вінниця

У світі числових методів для знаходження розв'язків рівнянь і оптимізаційних задач існує цілий арсенал інструментів, серед яких особливе місце займають метод дихотомії та метод ітерації. Вони допомагають нам вирішувати складні математичні проблеми, розглядаючи їх з різних поглядів та застосовуючи різні стратегії. Метод дихотомії і метод ітерації – це числові методи для знаходження наближеного розв'язку рівнянь або оптимізаційних задач. Перший базується на послідовному діленні інтервалу, другий – на ітеративному застосуванні перетворення до початкового наближення. Метод дихотомії гарантує збіжність, але може бути повільним, тоді як метод ітерації може збігатися швидше, але потребує добре обраного початкового наближення. Метод дихотомії ще називають методом половинного поділу. Під час розв'язання нелінійного рівняння методом половинного поділу задаються відрізок [a,b], на якому існує лише один розв'язок, і бажана точність $e > 0$ розв'язання задачі. Розглянемо алгоритм цього методу:

Нехай дано рівняння $f(x) = x\text{^}3 - 3x\text{^}2 + 24x + 10$. Необхідно знайти його корінь з точністю e на відрізку [0,1], на якому функція безперервна і у кінцях має значення різних знаків, тобто $f(a) \times f(b) < 0$. Отже, згідно з теоремою 1, на цьому відрізку існує хоча б один розв'язок рівняння.

Знаходиться середина відрізку [a,b] – точка c (рис. 1). Корінь може опинитись на відрізку [a,c] або на [c,b], чи співпасти з c. В останньому випадку метод припиняє роботу, інакше за допомогою перевірки виконання умов $f(a) \times f(c) < 0$і $f(c) \times f(b) < 0$ з'ясовується, на якій частині відрізку залишився корінь. Далі процедура повторюється для тієї половини відрізку, на якій є корінь, доки відрізок не зменшиться настільки, що його довжина буде менше від заданої похибки [1].

Алгоритм методу:

Крок 1. Знаходиться середина відрізка c: = (b+a)/2.

Крок 2. Перевіряються такі умови:

1) якщо $f(c) = 0$ – корінь знайдено;

2) якщо $f(a) \times f(c) < 0$ – корінь на [a,c], тому b:=c;

3) якщо $f(c) \times f(b) < 0$ – корінь на [c,b], тому a:=c.

Крок 3. Перевіряється умова b–a. Якщо вона виконується, то корінь знайдено. В цьому випадку він дорівнює (a+b)/2. Інакше повертаються до кроку 1.

	y=x^3-3x^2+24x+10				e=0,5*10^-6				[-1,0]	
№	f(a)	a	f(b)	b	c	f©	e		f(a)*f©	f©*f(b)
0	-18	-1	10	0	-0,5	-2,875	-1		51,75	-28,75
1	-2,875	-0,5	10	0	-0,25	3,796875	-0,5		-10,91601563	37,96875
2	-2,875	-0,5	3,796875	-0,25	-0,375	0,525391	-0,25		-1,510498047	1,994843
3	-2,875	-0,5	0,5253906	-0,375	-0,4375	-1,15796	-0,125		3,32913208	-0,60838
4	-1,157958984	-0,4375	0,5253906	-0,375	-0,40625	-0,31216	-0,0625		0,361473463	-0,16401
5	-0,312164307	-0,40625	0,5253906	-0,375	-0,390625	0,107632	-0,03125		-0,03359877	0,056549
6	-0,312164307	-0,40625	0,1076317	-0,390625	-0,398438	-0,10201	-0,01563		0,031843959	-0,01098
7	-0,10201025	-0,398438	0,1076317	-0,390625	-0,394531	0,002875	-0,00781		-0,000293234	0,000309
8	-0,10201025	-0,398438	0,0028746	-0,394531	-0,396484	-0,04955	-0,00391		0,005054798	-0,00014
9	-0,049551867	-0,396484	0,0028746	-0,394531	-0,395508	-0,02333	-0,00195		0,001156276	-6,7E-05
10	-0,023334664	-0,395508	0,0028746	-0,394531	-0,39502	-0,01023	-0,00098		0,000238692	-2,9E-05
11	-0,010229058	-0,39502	0,0028746	-0,394531	-0,394775	-0,00368	-0,00049		3,76123E-05	-1,1E-05
12	-0,003677003	-0,394775	0,0028746	-0,394531	-0,394653	-0,0004	-0,00024		1,47508E-06	-1,2E-06
13	-0,000401162	-0,394653	0,0028746	-0,394531	-0,394592	0,001237	-0,00012		-4,96122E-07	3,55E-06
14	-0,000401162	-0,394653	0,0012367	-0,394592	-0,394623	0,000418	-6,1E-05		-1,67597E-07	5,17E-07
15	-0,000401162	-0,394653	0,0004178	-0,394623	-0,394638	8,31E-06	-3,1E-05		-3,33318E-09	3,47E-09
16	-0,000401162	-0,394653	8,309E-06	-0,394638	-0,394646	-0,0002	-1,5E-05		7,8799E-08	-1,6E-09
17	-0,000196427	-0,394646	8,309E-06	-0,394638	-0,394642	-9,4E-05	-7,6E-06		1,84757E-08	-7,8E-10
18	-9,40588E-05	-0,394642	8,309E-06	-0,394638	-0,39464	-4,3E-05	-3,8E-06		4,03277E-09	-3,6E-10
19	-4,2875E-05	-0,39464	8,309E-06	-0,394638	-0,394639	-1,7E-05	-1,9E-06		7,41013E-10	-1,4E-10
20	-1,72831E-05	-0,394639	8,309E-06	-0,394638	-0,394639	-4,5E-06	-9,5E-07		7,75519E-11	-3,7E-11
21	-4,48715E-06	-0,394639	8,309E-06	-0,394638	-0,394638	1,91E-06	-4,8E-07		-8,57414E-12	1,59E-11
22	-4,48715E-06	-0,394639	1,911E-06	-0,394638	-0,394638	-1,3E-06	-2,4E-07		5,7802E-12	-2,5E-12
23	-1,28817E-06	-0,394638	1,911E-06	-0,394638	-0,394638	3,11E-07	-1,2E-07		-4,01041E-13	5,95E-13
24	-1,28817E-06	-0,394638	3,113E-07	-0,394638	-0,394638	-4,9E-07	-6E-08		6,29166E-13	-1,5E-13
25	-4,8842E-07	-0,394638	3,113E-07	-0,394638	-0,394638	-8,9E-08	-3E-08		4,32479E-14	-2,8E-14

Рис. 1. Метод дихотомії

Метод дихотомії має малу швидкість збіжності, оскільки інтервал, де знаходиться корінь, з кожним кроком зменшується не більше, ніж удвічі.

Метод ітерації, також відомий як метод простої ітерації або метод змінних точок, є числовим методом для розв'язання рівнянь або систем рівнянь. Він використовується для знаходження наближеного розв'язку рівняння шляхом послідовного покращення початкового наближення.

Основна ідея методу полягає в тому, щоб взяти початкове наближення до розв'язку і використовувати ітераційний процес для покращення цього наближення, поки не буде досягнута необхідна точність або максимальна кількість ітерацій [2].

Припустимо, що корінь $x*$ рівняння $f(x) = x^3 - 3x^2 + 24x + 10$ знаходиться на відрізку [a,b] (рис. 2). Тоді алгоритм розв'язку рівняння методом простої ітерації такий:

Згідно з методом простої ітерації, задане рівняння $f(x) = f(x) = x^3 - 3x^2 + 24x + 10$ замінюють функцією $\varphi(x)$, де $\varphi(x) = x - \lambda f(x)$, а – дійсна стала $(\lambda \neq 0)$, і обчислюють ітерації $xn + 1 = \varphi(xn)$.

Алгоритм методу:

1. Виокремлюють корні рівняння на відрізку ізоляції (будь-яким із методів).

2. Перевіряють умови застосування методу (знаки похідних).

3. Проводять пошук заміни $\varphi i(x) = x - \lambda i\, f(x)$ для кожного з відрізків i = 1, 2… і визначають λ.

4. Проводять розрахунки для відрізка (відрізків) ізоляції з урахуванням λ [3].

Отже, зробимо висновок, що методи дихотомії та ітерації є широко використовуваними алгоритмами для знаходження нуля (або кореня) функції. Ось деякі плюси та мінуси кожного з цих методів.

Метод дихотомії:

1. Плюси:

- простий у реалізації;
- гарантовано збігається до кореня функції, якщо вона є неперервною та має змінний знак на кінцях вибраного інтервалу;
- має стабільну збіжність, навіть за невеликої точності обчислень.

2. Мінуси:

- потребує великої кількості ітерацій для досягнення високої точності, особливо для функцій з малою швидкістю збіжності;
- неефективний для функцій з великими коливаннями або нерівномірним розподілом коренів на заданому інтервалі.

y=x^3-3x^2+24x+10
y'=3x^2-6x+24
y"=6x-6

x	y(x)	y'=f(x)	y"=f(x)
-10	-1530	384	-66
-9	-1178	321	-60
-8	-886	264	-54
-7	-648	213	-48
-6	-458	168	-42
-5	-310	129	-36
-4	-198	96	-30
-3	-116	69	-24
-2	-58	48	-18
-1	-18	33	-12
0	10	24	-6
1	32	21	0
2	54	24	6
3	82	33	12
4	122	48	18
5	180	69	24
6	262	96	30
7	374	129	36
8	522	168	42
9	712	213	48
10	950	264	60

f'(a)	33
f'(b)	24
Max(abs)	33

№	x	f(x)	L
0	-1	-18	0,030303
1	-0,454545	-1,62284	
2	-0,405368	-0,288426	
3	-0,396628	-0,053416	
4	-0,39501	-0,009963	
5	-0,394708	-0,001861	
6	-0,394651	-0,000348	
7	-0,394641	-6,49E-05	
8	-0,394639	-1,21E-05	
9	-0,394638	-2,27E-06	
10	-0,394638	-4,23E-07	
11	-0,394638	-7,91E-08	
12	-0,394638	-1,48E-08	

Рис. 2. Метод простої ітерації

Метод ітерації:

1. Плюси:

- швидше збігається до кореня, особливо порівняно з методом дихотомії;
- ефективний для функцій зі складними формами або з низькою швидкістю збіжності;
- дає змогу обчислення похідних функції, що може прискорити збіжність.

2. Мінуси:

- вимагає обчислення похідних функції, що може бути витратним процесом;
- не завжди гарантує збіжність, особливо якщо похідна змінює знак близько до кореня або функція має особливості;
- метод може розбігатися, якщо похідна змінює знак недалеко від кореня.

Вибір між цими методами залежить від конкретного завдання, типу функції та потрібної точності.

Список використаних джерел

1. Чисельні методи: навчальний посібник / Л. О. Волонтир, О. В. Зелінська, Н. А. Потапова, І. А. Чіков. Вінниця: ВНАУ, 2020. 322 с.

2. Швець Х. І., Потапова Н. А. Уточнення наближених розв'язків методом простої ітерації. *Прикладні інформаційні технології:* матеріали ІІІ Всеукраїнської науково-практичної конференції здобувачів, аспірантів та молодих вчених (м. Вінниця, 22 квітня 2022 р.). Вінниця: ДонНУ імені Василя Стуса, 2022. С. 44–46. URL: https://jait.donnu.edu.ua/article/view/12252

3. Павлов Д. Л., Потапова Н. А. Використання методу дихотомії для розв'язку прикладних задач. *Прикладні інформаційні технології:* матеріали ІІІ Всеукраїнської науково-практичної конференції здобувачів, аспірантів та молодих вчених (м. Вінниця, 22 квітня 2022 р.). Вінниця: ДонНУ імені Василя Стуса, 2022. С. 26–28. URL: https://jait.donnu.edu.ua/article/view/12244

УДК 530.145(075.8)

Діброва І. С., *здобувач 2 курсу спеціальності 122 Комп'ютерні науки,*
науковий керівник:
Потапова Н. А., *канд. екон. наук, доцент, доцент кафедри інформаційних технологій*

КВАНТОВІ ОБЧИСЛЕННЯ

Донецький національний університет імені Василя Стуса, м. Вінниця

Не відстаючи від швидкого розвитку технологій і науки, зростає попит на більш досконалі обчислювальні системи. Традиційні комп'ютери досягли своїх обмежень, коли справа доходить до вирішення складних завдань, що вимагає дослідження нових підходів до обчислень. Квантові обчислення пропонують багатообіцяльне рішення, яке використовує принципи квантової механіки для вирішення цільових проблем з більшою швидкістю та ефективністю [1].

Одним з основних завдань квантових обчислень є розробка та реалізація алгоритмів і технологій, які використовують квантові властивості матеріалів для вирішення обчислювальних завдань, складних для вирішення традиційними комп'ютерами. Це включає пошук нових квантових алгоритмів, розробку квантового апаратного забезпечення та дослідження можливості застосування квантових обчислень у різних галузях науки й техніки, від криптографії до матеріалознавства [2].

Квантові комп'ютери відрізняються від традиційних транзисторних комп'ютерів тим, що тоді, коли класичні комп'ютери працюють з використанням даних, закодованих у двійкових числах (бітах), коли квантовий комп'ютер використовує кванти, кожне число завжди знаходиться в одному з двох станів (0 або 1). Біти (кубіти) можуть перебувати в суперпозиції станів.

Теоретично квантові комп'ютери здатні вирішувати певні проблеми швидше, ніж класичні комп'ютери, як-от факторизація цілих чисел або проблема ефективного моделювання квантових систем багатьох тіл. Існує багато квантових алгоритмів, як-от алгоритм Шора, алгоритм Саймона та інші, які виконуються за набагато менший час, ніж будь-який класичний імовірнісний алгоритм [3].

У класичних комп’ютерах для обробки інформації використовуються основні двійкові цифри (біти). Фізична реалізація цього біта базується на принципі, що електричний потенціал може бути вищим за певний рівень (який відповідав би значенню біта 1) або нижчим (0).

У квантових комп’ютерах інформація також зазвичай представлена за допомогою двійкових елементів. У якості такого елемента використовується фізична система з двома можливими станами, яка описується у квантовій механіці двовимірним комплексним простором.

Для геометричного представлення квантового простору прийнято використовувати сферу Блоха [4] (рис. 1).

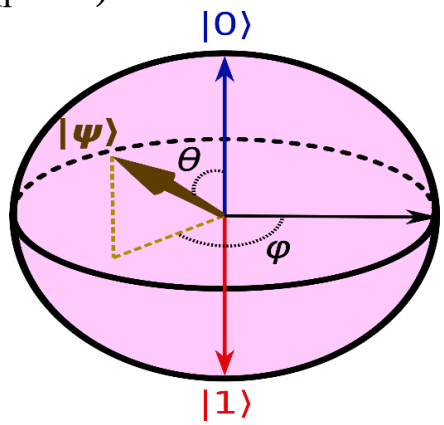


Рис. 1. Сфера Блоха для представлення квантового простору

Сфера Блоха є одиничною двовимірною сферою, кожна пара діаметрально протилежних точок якої відповідають взаємно ортогональним векторам стану. Зокрема, північний і південний полюси сфери Блоха вважаються такими, що відповідають базисним векторам [0] та [1], які можуть відповідати, наприклад, двом спіновим станам електрона («спін вгору» та «спін вниз»).

До того ж з квантової механіки відомо, що повна ймовірність має дорівнювати одиниці, тому на компоненти накладається умова нормування.

Актуальним напрямом для відкриття нових можливостей розв’язання складних завдань є застосування квантових обчислень у програмуванні. Одним із ключових аспектів програмування квантових обчислень є використання квантових алгоритмів. Ці алгоритми відрізняються від класичних тим, що вони використовують квантові біти (або кюбіти) замість класичних бітів.

Квантові обчислення в програмуванні можуть бути використані для розв’язання різних завдань, від криптографічних до оптимізаційних. Наприклад, алгоритм Шора використовується для факторизації великих чисел, що є важливим завданням у криптографії. Алгоритм Гровера може бути використаний для швидкого пошуку в базах даних.

Щоб використовувати квантові обчислення, програмісти зазвичай використовують спеціалізовані мови програмування, як-от Q# (розроблена Microsoft для роботи з квантовими обчисленнями) або Quipper. Ці мови мають спеціальні конструкції для роботи з квантовими об’єктами та алгоритмами.

Ще однією важливою характеристикою квантових обчислень є принцип суперпозиції. У класичних обчисленнях біт може знаходитися у стані 0 або 1, але у

квантових обчисленнях кюбіт може перебувати у суперпозиції обох станів одночасно, з деякою ймовірністю кожного стану. Це дає можливість використовувати квантові обчислення для паралельної обробки даних.

Одним із ключових принципів, на яких базуються квантові обчислення, є принцип квантової переплетеності. Квантова переплетеність означає, що стани двох або більше кюбітів можуть бути сполучені так, що стан кожного кюбіта залежить від станів інших кюбітів. Це дає змогу кюбітам утворювати взаємозалежні стани, що використовується для створення складних обчислювальних структур. Принцип квантової переплетеності можна виразити математично за допомогою тензорного добутку. Якщо ми маємо два кюбіти $|\psi 1\rangle$ та $|\psi 2\rangle$, які перебувають у станах $|\psi 1\rangle = \alpha|0\rangle + \beta|1\rangle$ та $|\psi 2\rangle = \gamma|0\rangle + \delta|1\rangle$ $|\psi 2\rangle = \gamma|0\rangle + \delta|1\rangle$ відповідно, то їх квантовий стан, що переплетений, можна записати так:

$$|\Psi\rangle = \alpha\gamma|00\rangle + \alpha\delta|01\rangle + \beta\gamma|10\rangle + \beta\delta|11\rangle \quad (1)$$

Фундаментальним математичним інструментом для аналізу квантових обчислень є квантова механіка. Квантова механіка надає математичні засоби для опису станів квантових систем, їх еволюції у часі та вимірювання. Основною структурою для опису квантових систем є квантовий стан, який може бути представлений у вигляді вектора у гільбертовому просторі.

Для опису еволюції квантових систем використовуються унітарні оператори, які забезпечують лінійне та обернене перетворення квантових станів. Унітарні оператори відіграють ключову роль у реалізації квантових вентилів, які є основними будівельними блоками квантових алгоритмів.

За допомогою цих математичних концепцій та квантових вентилів можна будувати складні квантові обчислювальні схеми для розв'язання різноманітних завдань. У подальших дослідженнях буде важливо дослідити ефективні методи синтезу та оптимізації квантових алгоритмів для різних застосувань у програмуванні.

Список використаних джерел

1. Вакарчук І. О. Квантова механіка. Львів: ЛНУ ім. Івана Франка, 2022. 872 с.
2. Ткачук В. М. Фундаментальні проблеми квантової механіки. Львів: ЛНУ ім. Івана Франка, 2021. 144 с.
3. Simon D. R. On the power of quantum computation. Foundations of Computer Science, 2004 Proceedings., 35th Annual Symposium. P. 116–123.

УДК 517.957]:(005.336.1-042.2:004.424.44:514.112)

Афанасьєва Д. С., *здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,*
науковий керівник:
Потапова Н. А., *канд. екон. наук, доцент, доцент кафедри інформаційних технологій*

ПОРІВНЯННЯ ЕФЕКТИВНОСТІ МЕТОДІВ ДИХОТОМІЇ ТА ХОРД У РОЗВ'ЯЗАННІ НЕЛІНІЙНИХ РІВНЯНЬ

Донецький національний університет імені Василя Стуса, м. Вінниця

В сучасній математичній та інженерній практиці розв'язання нелінійних рівнянь відіграє важливу роль у вирішенні різноманітних завдань, їх ефективне розв'язання має велике значення для розвитку науки та технологій. Одними з провідних методів розв'язання таких рівнянь є методи дихотомії та хорд. Проте для успішного застосування цих методів у практичних задачах необхідно детально оцінити їх ефективність та області застосування.

Для знаходження кореня рівняння $f(x) = 0$ з деякою точністю ε на відрізку $[a, b]$, де функція є неперервною та має значення різних знаків на кінцях a та b, можна використати метод дихотомії. Процес виконання методу полягає у такому: спочатку визначається середина відрізка $[a,b]$ (точка c). Якщо корінь рівняння співпадає зі значенням c, то обрахунки зупиняються, і результат обчислення отриманий. В іншому випадку корінь знаходиться на відрізку $[a, c]$ чи $[b, c]$, тоді значення для наступних ітерацій визначаються за допомогою перевірки умов $f(a) * f(c) < 0$ та $f(b) * f(c) < 0$. Далі алгоритм повторюється для відрізка, що містить розв'язок, доти, доки довжина відрізка не стане меншою від заданої похибки [1]. Приклад програмної реалізації методу дихотомії з використанням мови програмування Python показано на рис. 1.

```
def bisection_method(a, b, tol=0.5e-6, max_iter=1000):
    if func(a) * func(b) >= 0:
        print("Метод дихотомії не може бути використаний, оскільки "
              "на кінцях відрізка функція має однаковий знак.")
        return None
    iter_count = 0
    while (b - a) / 2.0 > tol:
        c = (a + b) / 2.0
        if func(c) == 0:
            return c
        elif func(c) < 0:
            a = c
        else:
            b = c
        iter_count += 1
        if iter_count > max_iter:
            print("Досягнуто максимальну кількість ітерацій.")
            return None
    print(f"Кількість ітерацій методу дихотомії: {iter_count}")
    print("Розв'язок методом дихотомії:", round(((a + b) / 2.0), 6))
```

Рис. 1. Програмна реалізація методу дихотомії

У конструкції while програмного коду показано принцип визначення a та b для кожної ітерації. Якщо $f(c) < 0$, то відрізок набуває значень $[c, b]$, якщо $f(c) > 0$ – $[a, c]$.

Використання методу хорд для знаходження кореня рівняння $f(x) = 0$ на відрізку $[a, b]$, де функція є неперервною, та $f(a) * f(b) < 0$, з деякою точністю ε базується на побудові певного числа прямих, що перетинають вісь x та проходять через дві початкові точки $f(x)$. Алгоритм реалізації цього методу включає пошук похідних першого та другого порядків для функції $f(x)$ на кінцях відрізка, та перевірку, чи знак похідних на них не змінюється. Після цього обирається значення початкового наближення, як будь-яка точка на відрізку, де виконується умова $f(x) * f''(x) < 0$. Далі визначаємо нерухомий кінець c, як будь-яку точку на відрізку, де виконується умова $f(\mathrm{c}) * f''(\mathrm{c}) > 0$. Значення x для першої ітерації буде рівне початковому наближенню, а для наступних ітерацій буде обчислене за формулою (1) [2]:

$$x_{k+1} = x_k - \frac{f(x_k)}{f(x_k) - f(c)}(x_k - c). \quad (1)$$

Продовжувати ітерації треба, поки абсолютне значення різниці між поточним та попереднім наближеннями менше за задану точність. Приклад програмної реалізації методу хорд з використанням мови програмування Python, де похідні першого та другого порядків визначаються попередньо заданими функціями first_derivative() та second_derivative(), показано на (рис. 2):

```
def chord_method(a, b, epsilon=0.5e-6, max_iter=1000):
        if first_derivative(a) * second_derivative(a) < 0 or \
            first_derivative(b) * second_derivative(b) < 0:
        print("На кінцях відрізка змінюється знак похідної")
        return None

    x0 = a
    while True:
        if func(x0) * second_derivative(x0) < 0: break
        else: x0 += 0.1

    c = b
    while True:
        if func(c) * second_derivative(c) > 0: break
        else: c -= 0.1

    iter_count = 0
    fc = func(c)
    while True:
        fx0 = func(x0)
        x = x0 - ((fx0 / (fx0 - fc))*(x0 - c))
        if abs(x - x0) < epsilon or iter_count >= max_iter: break

        x0 = x
        iter_count += 1

    print(f"Кількість ітерацій: {iter_count}")
    print("Розв'язок методом хорд:", round(x, 6))
```

Рис. 2. Програмна реалізація методу хорд

Після проведення обчислень за допомогою методів дихотомії та хорд можна здійснити порівняльний аналіз їх ефективності згідно з отриманими результатами:

1. Для функції, що була обрана для проведення дослідження, метод хорд є ефективнішим з погляду швидкості збіжності. Адже з отриманих результатів виконання програм можна побачити, що метод хорд потребував лише 8 ітерацій, порівняно з 20 ітераціями методу дихотомії (рис. 3).

```
Кількість ітерацій методу дихотомії: 20
Розв'язок методом дихотомії: 0.297786

Кількість ітерацій: 8
Розв'язок методом хорд: 0.297786
```

Рис. 3. Результати виконання програм

2. Зважаючи на те, що результати методу хорд були здебільшого залежні від зміни знака похідної, метод хорд може виявитися менш стійким до нульових похідних, ніж метод дихотомії.

3. Метод дихотомії не потребує визначення знаків похідних та зазвичай обмежується вибором кінців інтервалу. Натомість метод хорд може вимагати більше обчислень для вибору початкових значень, що може збільшити обчислювальну складність.

Внаслідок дослідження було встановлено, що для цього випадку метод хорд виявився більш ефективним, порівняно з методом дихотомії. Адже у практичних обчисленнях метод хорд продемонстрував меншу кількість ітерацій, порівняно з методом хорд, що свідчить про його більшу ефективність. Проте метод дихотомії у деяких випадках може бути більш простим у застосуванні, оскільки не вимагає знання знаків похідних та обмежується вибором кінців інтервалу. Отже, вибір методу залежить від конкретної задачі та вимог до швидкості і простоти обчислень.

Список використаних джерел

1. Комп’ютерне моделювання систем та процесів. Методи обчислень. Частина 1: навчальний посібник / Р. Н. Квєтний, І. В. Богач, О. Р. Бойко, О. Ю. Софина, О. М. Шушура. Вінниця: ВНТУ, 2012. 193 с. URL: http://kist.ntu.edu.ua/textPhD/kmsp.pdf

2. Чисельні методи: навчальний посібник / Л. О. Волонтир, О. В. Зелінська, Н. А. Потапова, І. А. Чіков. Вінницький національний аграрний університет. Вінниця: ВНАУ, 2020. 322 с.

3. Верещак Р. Знаходження наближеного розв’язку нелінійного алгебраїчного рівняння методом хорд. *www.mathros.net.ua: сайт для студентів спеціальності інформатика.* 2012. URL: https://www.mathros.net.ua/znahodzhennja-nablyzhenogo-rozvjazku-nelinijnogo-algebraichnogo-rivnjannja-metodom-hord.html (дата звернення: 02.05.2024).

4. Smit A. Program for Bisection Method. *geeksforgeeks.org.* 2022. URL: https://www.geeksforgeeks.org/program-for-bisection-method/ (дата звернення: 02.05.2024).

УДК 004.6

Тимчук О. Г., *здобувач*
1 курсу ОС «Магістр»
спеціальності 122 Комп'ютерні науки,
Потапова Н. А., *канд. екон. наук,*
доцент кафедри
інформаційних технологій

МОДЕЛІ ТЕСТУВАННЯ ПРОГРАМНИХ ПРОДУКТІВ

Донецький національний університет імені Василя Стуса, м. Вінниця

Програмне забезпечення є невід'ємною частиною сучасного суспільства і пронизує кожен аспект нашого життя, від охорони здоров'я до комерції. Однак зі зростанням його важливості зростають і вимоги до якості та надійності. У цьому контексті тестування програмного забезпечення є важливим етапом розробки для виявлення та виправлення дефектів, задоволення потреб користувачів та забезпечення надійності продукту. Тестування програмного забезпечення – це складний, багатогранний процес, що включає в себе різні методи і моделі. Вибравши правильний підхід до тестування, розробники можуть забезпечити якість і продуктивність продукту. За останні кілька десятиліть як програмне забезпечення, так і методології тестування значно еволюціонували. Ми живемо в епоху постійних змін та інновацій, і тестування програмного забезпечення йде в ногу з цими змінами. Постійно вдосконалюючи методи тестування, ми можемо забезпечити високу якість і надійність програмних продуктів, які відповідають сучасним вимогам користувачів. Тому обґрунтованість питань щодо тестування програмних продуктів залишається завжди актуальною.

З урахуванням швидкого темпу змін у сфері IT та постійного вдосконалення програмних продуктів методи та моделі тестування також постійно еволюціонують. Оновлені підходи до тестування дають змогу ефективно впроваджувати нові функціональності та забезпечувати стабільну роботу програм. Існує чимало моделей тестування, але серед усіх виділяються такі:

- Модель водоспаду (Waterfall model).
- Модель V-форми (V-model).
- Інкрементна модель (Incremental model).
- Модель спіралі (Spiral model).

Модель водоспаду (Waterfall model). Її також називають лінійною послідовною моделлю життєвого циклу. У моделі водоспаду кожна фаза повністю завершується лише перед початком наступної фази. Така модель розробки програмного забезпечення використовується для невеликих проєктів без чітких визначених вимог. У кінці кожної фази проводяться тести, щоб визначити, чи відповідає проєкт поставленим вимогам. Тестування починається тільки тоді, коли завершується розробка [1].

Переваги водоспаду: Waterfall є доволі простим для розуміння і використання. У цій моделі фази виконуються послідовно і завершуються також послідов-

но. Фази не можуть повторюватися. Модель водоспаду найкраще підходить для невеликих проєктів.

Недоліки водоспаду: коли додаток знаходиться на етапі тестування, дуже важко повернутися назад і змінити речі, які не були повністю продумані на етапі розробки концепції; ризики через невизначеність високі; не підходить для складних або об'єктно-орієнтованих проєктів; не підходить для проєктів, де вимоги змінюються від середнього ризику до високого; не підходить для довгострокових, безперервних проєктів.

V-модель (V model) розшифровується як модель верифікації та валідації. Подібно до моделі водоспаду, життєвий цикл V-моделі – це послідовний шлях процесів. Кожна фаза повинна бути завершена до того, як почнеться наступна фаза. У цій моделі план тестування системи готується до початку розробки. План тестування спрямований на виконання функцій, визначених під час збору вимог [2].

Переваги V-моделі: вона легка і проста у використанні; економиться багато часу, оскільки планування, розробка тестів та інші дії з тестування відбуваються задовго до написання коду.

Недоліки V-моделі: програмне забезпечення розробляється на етапі реалізації, тому перший прототип програмного забезпечення не створюється.

Інкрементна модель (Incremental model). Ця модель використовує розбиття вимог на різні збірки, через що відбувається багато циклів розробки. Цикли поділяються на менші керовані модулі. У цій моделі кожен модуль послідовно проходить через фази вимог, проєктування, реалізації та тестування. Робоча версія програмного забезпечення створюється в першому модулі. Отже, з програмним забезпеченням можна працювати на ранній стадії життєвого циклу. У кожній наступній версії модуля додається функціональність до попередньої версії. Цей процес триває до тих пір, поки не буде завершена вся система [3].

Переваги інкрементної моделі: модель є дуже гнучкою, а вартість зміни обсягу та вимог є низькою.

Недоліки інкрементної моделі: вона вимагає відповідного планування проєктування, і вся система повинна бути чітко і повністю визначена до того, як система буде декомпонована і побудована поетапно.

Спіральна модель (Spiral model) подібна до інкрементної моделі, але з великим акцентом на аналіз ризиків. Модель поділяється на чотири фази: планування, аналіз ризиків, розробка та оцінка. Програмні проєкти проходять через ці фази ітеративно (в цій моделі вони називаються спіралями). Базова спіраль починається з фази планування, де збираються вимоги та оцінюються ризики. Кожна наступна спіраль ґрунтується на базовій спіралі [4].

Переваги спіральної моделі: ризиків можна уникнути завдяки колективному аналізу ризиків; найкраще підходить для великих і важливих проєктів; функціональність може бути додана не одразу; програмне забезпечення може бути створене на ранній стадії життєвого циклу програмного забезпечення.

Недоліки спіральної моделі: може бути дорогою у використанні; успіх проєкту значною мірою залежить від етапу аналізу ризиків; не підходить для малих проєктів.

Отже, методи та моделі тестування, як-от модель водоспаду, V-модель, інкрементна модель та спіральна модель, надають розробникам різні стратегії для забезпечення якості та надійності продукту.

Список використаних джерел

1. Devaraj K. Waterfall Model in Software Testing. 2024. URL: https://testsigma.com/blog/waterfall-model-in-software-testing/ (дата звернення: 02.05.2024).

2. Swain D. V Model in Software Testing | What, Why, Advantages and Disadvantages. 2024. URL: https://testsigma.com/blog/v-model-in-software-development-life-cycle/ (дата звернення: 02.05.2024).

3. Kumari R. Incremental Testing in Software Testing. 2024. URL: https://testsigma.com/blog/incremental-testing/ (дата звернення: 02.05.2024).

4. Saxena A. Spiral Model in Software Testing. 2023. URL: https://testsigma.com/blog/spiral-model/ (дата звернення: 02.05.2024).

УДК 004.774.6:004.92]: 7.012(494)

Бурківський О. С., *здобувач 2 курсу спеціальності 122 Комп'ютерні науки,*
науковий керівник:
Зелінська О. В., *канд. техн. наук, доцент, доцент кафедри інформаційних технологій*

ШВЕЙЦАРСЬКИЙ СТИЛЬ У ВЕБДИЗАЙНІ

Донецький національний університет імені Василя Стуса, м. Вінниця

Більшість людей обізнані з чудовим швейцарським стандартом. Ця традиція дизайну, заснована майже 100 років тому, залишається незмінною та актуальною і сьогодні. Швейцарський дизайн відмінно справляється з викликами сучасного бізнес-світу, втілюючи в собі якість, простоту й інновації.

Швейцарський дизайн відомий своєю найвищою точністю та простотою серед усіх стилів. Він підкреслює функціональність і практичність, віддаючи перевагу призначенню перед декором. У цьому стилі філософія полягає в тому, що дизайнери повинні бути справжніми майстрами своєї справи, підкреслюючи мистецтво ремесла над мистецтвом [1].

Цей стиль дизайну відрізняється своєю стриманістю і чіткістю в передачі ідеї. Він прагне до простоти, але водночас зберігає високу якість і естетичний стиль. Швейцарський дизайн вчить нас тому, як використовувати мінімалізм та чіткість для створення сильних і лаконічних зображень.

Цей стиль має свої переваги у сучасному світі дизайну:

– дає змогу зосередитися на суттєвих аспектах без зайвих деталей;

– забезпечує консистентність і виразність іміджу бренду;

– підвищує ефективність комунікації, створюючи чітке й зрозуміле візуальне враження.

Швейцарський дизайн – це не лише стиль, але й філософія, яка навчає нас привносити ясність та елегантність у кожен аспект створення.

Швейцарський метод дизайну виникає з простих геометричних форм і узагальнень. Він відрізняється регулярністю і чіткістю, що робить його ідеальним для створення графічних концепцій для брендів, які підкреслюють структурну послідовність об’єктів.

Одним із видатних досягнень швейцарського стилю є розробка модульної (математичної) сітки, яка використовується для створення різноманітних композицій. Цей підхід спрощує ієрархізацію контенту та організацію простору.

Основні риси швейцарського дизайну включають:

– простоту і зрозумілість;
– функціональність і точність;
– чіткість у сприйнятті;
– використання порожнього простору;
– модульна структура;
– використання беззасічкової типографіки.

Швейцарський дизайн володіє потужним інструментарієм для створення привабливих і організованих візуальних концепцій, які забезпечують чіткість і стабільність у сприйнятті. Його особливості включають складні, але виразні композиції і сміливу графіку, а також використання відмінних заголовків [2].

Представники швейцарського стилю розробили прості, універсальні та легкі для читання шрифти без засічок, як-от Transit, Saskia, Zeus, Sabon, Uhertype Grotesk та інші. У 1970-х роках ці шрифти часто використовувалися на знаках метро Нью-Йорка. Ці нейтральні шрифти підкреслюють значення літер, не відволікаючи увагу на їх форму. Один із найвідоміших шрифтів у швейцарському стилі – Helvetica [1].

Графічний дизайн у швейцарському стилі здобуває популярність серед українських споживачів і є частим вибором для створення логотипів та ідентифікаційних елементів брендів. Вирівнювання кожного рядка і літер допомагає створити організовану структуру, яка відображає цінності бренду [3].

Багато дизайнерів вивчають швейцарський стиль і застосовують його принципи для створення незвичайних вебсайтів та оригінальних друкованих матеріалів, які привертають увагу потенційних клієнтів і замовників. Один із прикладів застосування стилю зображено на рис. 1.

Рис. 1. Приклад швейцарського стилю

В Україні швейцарський стиль відомий завдяки багатьом графічним дизайнерам з усього світу. Ральф Шлейфогель, відомий своїми перемогами у престижному Міжнародному конкурсі плакатів імені Шаумана (1996, 2010, 2017), відзначається своїми вражаючими роботами у цій галузі. Крім плакатів, він також займається створенням обкладинок книг і розробкою стилів.

Майк Джойс, дизайнер із Нью-Йорка, є великим шанувальником швейцарського модернізму. Його роботи, зокрема плакати для рок-концертів, представлені на авторському вебсайті swissed.com.

Ніклаус Трокслер є ще одним видатним сучасним швейцарським дизайнером, який спеціалізується на розробці дизайн-концепцій для архітектурних проєктів та подій. Його роботи показані на рис. 2.

Рис. 2. Приклад швейцарського стилю у світі

Отже, швейцарський стиль у вебдизайні представляє сучасний, мінімалістичний підхід до створення вебсайтів, який виник внаслідок розвитку принципів, визначених швейцарськими дизайнерами у 1950-х і 1960-х роках. Його основними характеристиками є чіткість, простота і функціональність. Поняття швейцарського стилю включає в себе використання сітчастої структури, великих шрифтів, раціональне розташування елементів, білий простір та мінімальне використання декоративних деталей. Цей дизайнерський підхід спрямований на те, щоб дати змогу користувачам зосередитися на основному контенті та забезпечити простий і легкий для сприйняття вебдизайн, що має бути зручним для користувачів. Завдяки своїй універсальності і ефективності швейцарський стиль підходить для будь-якого типу вебсайту, від корпоративних до креативних проєктів. Він створений з метою підкреслити контент і спростити навігацію, що робить його популярним серед сучасних дизайнерів.

Список використаних джерел

1. Швейцарский стиль: точность и функциональность до мельчайших деталей: вебсайт. URL: https://welovebrands.com.ua/blog/swiss-style/ (дата звернення: 02.05.2024).

2. Зелінська О. В., Колосова К. К. Огляд методів UX-досліджень під час створення IT-продуктів. *Вісник студентського наукового товариства Донецького національного університету імені Василя Стуса*. 2022. Вип. 14, т. 1. С. 267–270. URL: https://jvestnik-sss.donnu.edu.ua/article/view/12827

УДК 004.451.84/.514

Сімон К. А., *здобувачка 4 курсу спеціальності 122 Комп'ютерні науки,*
Зелінська О. В., *канд. техн. наук, доцент, доцент кафедри інформаційних технологій*

ВИКОРИСТАННЯ ПРИНЦИПІВ UX/UI-ДИЗАЙНУ У РОЗРОБЦІ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ОРГАНІЗАЦІЇ ЗАХОДІВ

Донецький національний університет імені Василя Стуса, м. Вінниця

Ефективний UX- і UI-дизайни є основними складниками успішної розробки будь-якої інформаційної системи. UX, або взаємодія з користувачем, визначає враження користувача від використання інтерфейсу. Він спрямований на створення задоволення та позитивних емоцій під час користування продуктом [1]. UI (User Interface) – це дизайн користувацького інтерфейсу, який фокусується на візуальному складнику та естетиці. Він включає в себе розробку шрифтів, кольорів, графіки, іконок, розташування елементів на екрані. UI-дизайнер робить інтерфейс не лише красивим, але й зручним та інтуїтивно зрозумілим.

UX- та UI-дизайни тісно пов'язані аспекти розробки успішного інтерфейсу. UX є основою, на якій базується UI. Натомість UI представляє візуальне втілення концепції UX [2].

Врахування та ефективне використання цих принципів дає змогу створювати інформаційні системи, що задовольняють потреби користувачів та забезпечують їх задоволенням від користування.

Основні принципи UX/UI-дизайну:

Пристосування інтерфейсу до потреб користувача є одним із ключових принципів UX/UI-дизайну. Це означає, що інтерфейс повинен бути створений з урахуванням потреб та очікувань цільової аудиторії. Мінімалізм і простота в дизайні також важливі. Спрощений та лаконічний дизайн полегшує сприйняття та навігацію для користувачів.

Ще одним важливим принципом є полегшення навігації та взаємодії з системою. Чітка та логічна структура допомагає користувачам швидко знаходити потрібну інформацію та функції [3]. Також важливо забезпечити доступність і використання стандартів, що забезпечить сумісність та зручність для широкого кола користувачів. Нарешті, використання психології користувача для поліпшення UX допомагає створювати більш ефективні та привабливі інтерфейси.

UX/UI-дизайн застосовується в розробці інформаційної системи для організації заходів. У розробці інформаційних систем для організації заходів критично важливо враховувати потреби аудиторії. Наприклад, система має бути спроєктована так, щоб легко знаходити інформацію про різні заходи, реєструватися на них та купувати квитки. Створення інтуїтивно зрозумілих інтерфейсів дає змогу користувачам легко та з комфортом користуватися системою [4].

Максимально ефективна взаємодія з системою є ключовою для зручного планування та участі у заходах. Наприклад, швидкий доступ до календаря подій,

можливість фільтрувати заходи за різними критеріями, як-от дата, тип події та ін., є важливими аспектами оптимізації [5].

Використання аналізу взаємодії користувачів із системою дає змогу збирати дані про їх поведінку та реакції на інтерфейс. Ці дані можуть бути використані для подальшого вдосконалення інформаційної системи.

Приклади успішного впровадження UX/UI-дизайну в системи організації заходів.

Один із прикладів успішного впровадження UX/UI-дизайну – це мобільні додатки для планування подій. Такі додатки забезпечують користувачам можливість швидко і зручно переглядати та реєструватися на заходи, отримувати сповіщення про них, додавати їх у свій календар тощо.

Іншим прикладом є вебплатформи з можливістю реєстрації на заходи та покупки квитків. Ці платформи забезпечують зручний інтерфейс для користувачів, що дає змогу швидко і легко здійснювати реєстрацію на заходи та оплату квитків.

Також існують розвинуті системи організації заходів для корпоративного сектору зі вдосконаленим UX/UI-дизайном. Ці системи допомагають організаторам подій ефективно планувати та керувати корпоративними заходами за допомогою зручного та інтуїтивно зрозумілого інтерфейсу.

UX/UI-дизайн має вплив на ефективність інформаційної системи для організації заходів. UI та UX сприяють безперебійному користувацькому досвіду для споживачів, що взаємодіють із продуктами або послугами. Важливо зазначити, що ці аспекти мають вплив не лише на поверхневому рівні, але й у контексті SEO [6]. SEO – це набір заходів, які включають у себе внутрішню та зовнішню оптимізацію з метою підвищення рейтингу сайта в пошукових системах [7].

Існує шість способів, як UX/UI можуть мати вплив на SEO:

- клікабельність (якісний UX/UI може підвищити ймовірність того, що користувачі натиснуть на результати пошуку);
- час витримки (чим привабливіший UX/UI, тим більше часу користувачі проводять на сайті перед поверненням до пошукових результатів);
- відвідування сторінок (ефективний UX/UI спонукає користувачів відвідувати більше сторінок сайта);
- конверсії (добре розроблений сайт із відмінним UX/UI може підвищити кількість конверсій, як-от підписки або продажі);
- якість контенту (UX/UI може впливати на сприйняття якості контенту користувачами, і відповідно на рейтинг сайта в пошукових системах);
- голосовий пошук (якісний UX/UI-дизайн може позитивно вплинути на ранжування в голосовому пошуку) [6].

Ефективний UX/UI-дизайн інформаційної системи для організації заходів може суттєво вплинути на її успішність і ефективність. Зокрема, забезпечення задоволення користувачів від взаємодії з системою, зменшення кількості помилок та непорозумінь під час користування нею, а також підвищення загального рівня використання системи є важливими аспектами.

Список використаних джерел

1. У чому різниця між UX і UI: вебсайт. *Redstone.* URL: https://redstone.media/shcho-take-ux-ui-dysayn#:~:text= (дата звернення: 02.05.2024).
2. Що таке UX-дизайн та його принципи. *Outsourcing Team.* 03.03.2024. URL: https://outsourcing.team/uk/blog/design/shho-take-ux-dizajn-ta-jogo-printsipi/ (дата звернення: 02.05.2024).
3. Norman D. A. The Design of Everyday Things. Basic Books, 2013.
4. Research-based, practitioner-focused. Your source for UX guidance and training. *NN/g.* URL: https://www.nngroup.com/ (дата звернення: 02.05.2024).
5. Kuniavsky M. Observing the User Experience: A Practitioner's Guide to User Research. Morgan Kaufmann, 2012.
6. Rose-Collins F. UX/UI та його вплив на SEO. *Ranktracker*. Jul 31, 2024. URL: https://www.ranktracker.com/uk/blog/ux-ui-and-its-impact-on-seo/ (дата звернення: 02.05.2024).
7. Зелінська О. В., Колосова К. К. Огляд методів UX-досліджень під час створення IT-продуктів. *Вісник студентського наукового товариства Донецького національного університету імені Василя Стуса*. 2022. Вип. 14, т. 1. С. 267–270. URL: https://jvestnik-sss.donnu.edu.ua/article/view/12827

УДК 519.87:004.93

Журовський Я. О., *здобувач 3 курсу спеціальності 122 Комп'ютерні науки,*
Ніколюк П. К., *д-р фіз.-мат. наук, професор, професор кафедри інформаційних технологій*

ЕКСТРАПОЛЯЦІЯ ЯК ЗАСІБ МОДЕЛЮВАННЯ ЗА ДОПОМОГОЮ ПРОГРАМИ MAPLE

Донецький національний університет імені Василя Стуса, м. Вінниця

У сучасному науковому та технічному середовищі моделювання і прогнозування великого обсягу даних відіграють ключову роль у розв'язанні різноманітних завдань. Одним з ефективних інструментів для цього є методи екстраполяції, які допомагають будувати моделі за допомогою наявних даних і розширювати їх за межі відомого діапазону.

Ця наукова робота присвячена вивченню та застосуванню методів екстраполяції з використанням програмного забезпечення Maple. Maple є потужним інструментом для обчислення та моделювання, що надає широкі можливості у сфері математичного моделювання. Використання методів екстраполяції за допомогою Maple дає змогу ефективно моделювати та прогнозувати значення функцій за їх відомими показниками.

Метою роботи є дослідження різних методів екстраполяції в програмі Maple, аналіз їх ефективності та застосування для моделювання різноманітних процесів і явищ. До того ж у межах роботи буде проведено порівняння різних підходів до екстраполяції та вивчено вплив параметрів на точність прогнозування.

Важливість цього дослідження полягає в його потенційній здатності допомогти у розв'язанні реальних завдань моделювання та прогнозування у наукових,

технічних та інженерних дисциплінах. Результати роботи можуть мати практичне значення для розробки нових методів аналізу даних, оптимізації процесів та управління проєктами.

У роботі буде розглянуто теоретичні аспекти екстраполяції, методи та їх використання у програмі Maple, досліджено вплив параметрів на результати моделювання, а також наведено приклади застосування екстраполяції в різних областях науки та техніки.

Отже, ця робота має на меті розширити наше розуміння екстраполяції як інструменту моделювання і продемонструвати його потенціал у контексті програмного забезпечення Maple для вирішення актуальних проблем наукового та технічного співтовариства.

Теоретичні основи екстраполяції включають низку ключових понять і методів, які використовуються для побудови моделей за допомогою наявних даних і розширення їх за межі відомого діапазону. Основні аспекти теорії екстраполяції включають:

1. Екстраполяція та інтерполяція:

- інтерполяція – це процес побудови нових даних у межах відомого набору даних, коли маємо точні значення функції у деяких точках;
- екстраполяція – це процес визначення значень функції за межами відомого діапазону на основі даних, які знаходяться в межах цього діапазону або поблизу нього.

2. Лінійна екстраполяція:

- використовує пряму лінію для продовження функції за межі відомих даних;
- підходить для випадків, коли залежність між змінними вважається лінійною.

3. Нелінійна екстраполяція:

- використовує нелінійні моделі для продовження функції за межі відомого діапазону;
- може бути більш точною, але вимагає додаткових даних та складніших обчислень.

4. Методи екстраполяції:

- метод найменших квадратів – використовуються для апроксимації функцій та екстраполяції даних шляхом знаходження найменшої розбіжності між функцією та даними;
- метод регресії – використовується для аналізу залежності між змінними та прогнозування значень за допомогою цієї залежності;
- метод найближчих сусідів – використовує найближчі точки даних для прогнозування значень функції.

5. Використання статистичних методів:

- можливість врахування статистичних показників (наприклад, середнього, дисперсії) для покращення точності екстраполяції.

6. Оцінка точності екстраполяції:

- важливість оцінки точності екстраполяції для визначення достовірності результатів інтерполяції за межами відомого діапазону.

Методи екстраполяції в програмі Maple можуть бути представлені різними підходами та функціями, які надаються цим програмним забезпеченням. Ось кілька типових методів екстраполяції, які можна реалізувати в Maple:

1. Лінійна екстраполяція: для лінійної екстраполяції можна використовувати функції CurveFitting:-PolynomialInterpolation або CurveFitting:-LinearFit, щоб побудувати пряму лінію, яка продовжиться за межі відомого діапазону даних.

```
with(CurveFitting):
data := [[x1, y1], [x2, y2], ...];  # Набір даних у форматі [x, y]
fit := LinearFit(data, x);          # Лінійна апроксимація
extrapolated_value := evalf(eval(fit, x = new_x));  # Екстраполяція для нового значення new_x
```

2. Нелінійна екстраполяція: для нелінійної екстраполяції можна використовувати функцію CurveFitting:-NonlinearFit, яка дає змогу побудувати нелінійну модель залежності між даними.

```
with(CurveFitting):
data := [[x1, y1], [x2, y2], ...];  # Набір даних у форматі [x, y]
fit := NonlinearFit(data, model, parameters, x);  # Нелінійна апроксимація за моделлю model
extrapolated_value := evalf(eval(fit, x = new_x));  # Екстраполяція для нового значення new_x
```

3. Інтерполяція: інтерполяція також може бути використана для екстраполяції за межами відомого діапазону даних. Для цього можна використовувати функцію Interpolation:-Interpolate, яка будує апроксимацію функції на основі інтерпольованих значень.

```
with(Interpolation):
data := [[x1, y1], [x2, y2], ...];  # Набір даних у форматі [x, y]
interp := Interpolate(data, x);     # Інтерполяція за допомогою зазначеного методу
extrapolated_value := evalf(eval(interp, x = new_x));  # Екстраполяція для нового значення
#new_x
```

Ці приклади демонструють основні підходи до екстраполяції, які можна використовувати в програмі Maple. Залежно від конкретних вимог і характеру даних можна вибрати відповідний метод екстраполяції для досягнення потрібної точності і результатів.

Дослідження впливу параметрів на результати екстраполяції важливо для розуміння та вдосконалення моделювання. Основні параметри для аналізу включають ступінь полінома апроксимації, метод апроксимації, кількість даних, шум та випадкові помилки, вибір методу екстраполяції та діапазон екстраполяції. Це дослідження допомагає визначити оптимальні налаштування моделі та її точність.

Прогнозування попиту на пальне може бути прикладом застосування екстраполяції в реальних задачах, де аналізуються історичні дані про продажі для оптимізації поставок і управління запасами. Наприклад, компанія енергетичної галузі може використовувати екстраполяцію для прогнозування попиту на пальне в майбутньому місяці на основі попередніх продажів, економічних показників та сезонних тенденцій.

Ще одним прикладом може бути використання екстраполяції в оборонній галузі для прогнозування кількості поранених у бойових діях. Військові медики можуть використовувати попередні дані про поранених та інші фактори, як-от інтенсивність бойових дій та умови бою, для прогнозування можливого обсягу медичних ресурсів, необхідних для надання допомоги.

Ці приклади демонструють, як екстраполяція може бути використана в різних сферах, від бізнесу до воєнних дій, для прогнозування та планування на основі наявних даних і тенденцій.

Список використаних джерел

1. Smith, J., Johnson, A. (2018). Introduction to Extrapolation Techniques. *Journal of Mathematical Modeling, 15*(2), 45–56.

2. Brown, R., Wilson, K. (2020). Applications of Extrapolation in Engineering Design. *Proceedings of the International Conference on Computational Methods,* 112–125.

3. Lee, S., Park, H. (2019). Utilizing Maple for Extrapolation Analysis: A Case Study in Financial Forecasting. *Maple Applications in Industry, 8*(4), 332–345.

4. Garcia, M., Martinez, P. (2017). Statistical Methods for Extrapolation: A Comprehensive Guide. Springer.

5. Chen, L., Wang, Q. (2022). Advanced Techniques in Extrapolation: Insights from Maple Simulations. *IEEE Transactions on Computational Modeling, 25*(3), 78–89.

УДК 004.94

Комар О. О., *здобувач вищої освіти,*
Ніколюк П. К., *д-р фіз.-мат. наук, професор, професор кафедри інформаційних технологій*

ПЕРСПЕКТИВИ СТВОРЕННЯ МОДЕЛЕЙ З ВИКОРИСТАННЯМ КЛІТИННИХ АВТОМАТІВ

Донецький національний університет імені Василя Стуса, м. Вінниця

У сучасному світі, коли складні системи та явища ставлять все нові виклики, зростає потреба в ефективних методах їх дослідження та моделювання. Одним із таких методів є метод клітинних автоматів (КА), який завдяки своїй простоті, універсальності та гнучкості може бути застосований у найрізноманітніших галузях науки.

Клітинні автомати вже давно зарекомендували себе як потужний інструмент для моделювання динамічних систем. Їх успішно використовують у сферах:

- фізики (моделювання фазових переходів, турбулентності, росту кристалів, поведінки газів і рідин);
- біології (моделювання еволюції, поширення хвороб, росту тканин, поведінки популяцій);
- економіки (моделювання динаміки ринків, поширення інновацій, поведінки економічних агентів);
- соціальних наук (моделювання поширення інформації, поведінки людей, розвитку міст, еволюції соціальних систем);
- інженерії (моделювання транспортних потоків, роботи роботів, поширення забруднень, поведінки складних систем);

Дослідження та створення моделей з використанням КА є перспективним напрямом. КА мають просту концепцію, що робить їх доступними для дослідни-

ків з різних галузей, і можуть бути використані для моделювання широкого спектру явищ.

Загалом КА – це впорядкований набір комірок. У загальному випадку розглядається n-вимірна решітка. Проте на практиці найчастіше для дослідження використовуються клітинні автомати малої розмірності з одно- або двовимірними решітками.

Структура просторової решітки залежить від форми комірок, з яких вона складається. Наприклад, у двовимірному випадку можна розглядати комірки трикутної, квадратної або шестикутної форм. Найбільшої популярності набули клітинні автомати, в яких клітки є квадратними, а решітка прямокутна.

Кожна комірка пам'яті клітинного автомата може зберігати одне значення із деякої скінченної множини значень. Час для клітинного автомата є дискретним (змінюється дискретними кроками – тактами) [1]. Зміна значень усіх комірок решітки відбувається синхронно і одночасно відповідно до правил переходу, за якими визначається нове значення кожної комірки як функція від поточних значень сусідніх комірок.

Класичні клітинні автомати характеризуються такими властивостями:

- *паралельність обчислень.* Класичний КА – це дискретна динамічна система з паралельним обчисленням значень комірок пам'яті;
- *властивість локальності.* Значення кожної комірки пам'яті на наступному такті роботи КА залежить від поточних значень комірок у деякому її околі (і, можливо, від значення власне у самій комірці);
- *властивість однорідності.* Правила переходу є однаковими для всіх комірок клітинного автомата.

Найбільш відомою інтерпретацією КА є «Гра життя», розроблена математиком Джоном Конвеєм [2]. Гра реалізується на площині, що складається з квадратних комірок. Кожна комірка має вісім сусідів (включно з тими, з якими вона стикається куточками). Кожна комірка може знаходиться у двох станах: «живому» (зайнятому, зазвичай показують чорним кольором) і «мертвому» (вільному, білому). Для запуску гри треба створити початкову конфігурацію («перше покоління»). Кожне наступне покоління визначається попереднім з урахуванням двох правил:

– «мертва» клітина, що має трьох «живих» сусідів, стає «живою»;

– «жива» клітина, що має менше двох або більше трьох «живих» сусідів, стає «мертвою».

Обчислювати наступні покоління є сенс, поки на полі залишаються «живі» клітини або поки система не входить у цикл, повторюючи один зі своїх попередніх станів. Вказаних правил достатньо для породження безлічі патернів, які демонструють складну поведінку. Багато конструкцій із клітин швидко деградують. Деякі виявляються стійкими. Існують «планери» («gliders») – патерни, здатні циклічно змінювати свою конфігурацію, необмежено переміщаючись по площині [3].

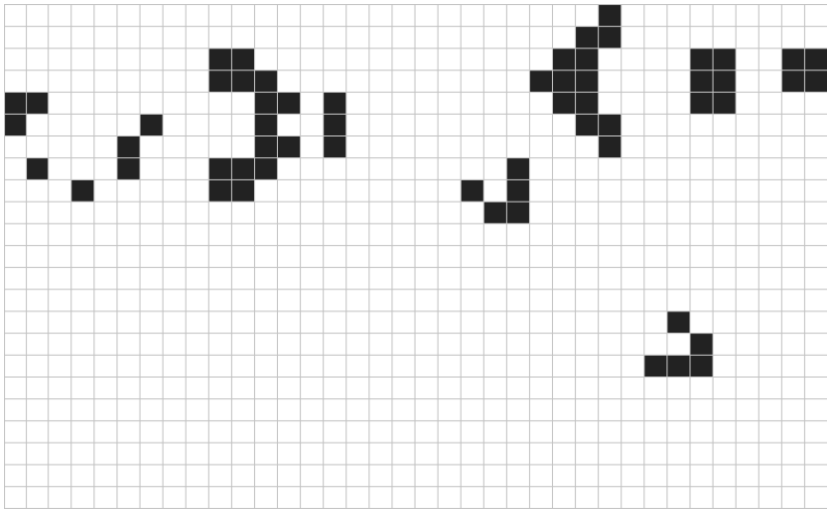


Рис. 1. Клітинний автомат – «Гра життя»

Під час створення моделей з використанням КА важливо враховувати низку ключових аспектів. По-перше, необхідно чітко визначити початкові умови системи, як-от розмірність клітинного поля, стан кожної клітини на початку моделювання та правила їх взаємодії. Далі необхідно обрати відповідний тип автомата залежно від характеру модельованої системи: одновимірний, двовимірний або багатовимірний. Під час розробки правил взаємодії між клітинами треба враховувати їх околицю, адаптувати правила до конкретності системи та цілей моделювання. До того ж важливо враховувати вплив параметрів моделі, як-от швидкість поширення інформації, ймовірність переходу між станами клітини та інші внутрішні параметри системи.

Щодо перспектив розвитку такого моделювання варто зазначити кілька ключових напрямів. Насамперед поєднання КА з іншими методами моделювання, як-от штучні нейронні мережі, може сприяти створенню більш складних та реалістичних моделей. Також важливим є розвиток адаптивних клітинних автоматів, які здатні самостійно змінювати свої правила залежно від змін у середовищі. Додатково, вивчення властивостей самоорганізації та емерджентності в системах на основі КА відкриває шлях до більш глибокого розуміння природних та соціальних процесів.

З урахуванням цих перспектив моделювання за допомогою КА відкриває широкі можливості для досліджень у різних галузях, від біології до комп’ютерних наук, сприяючи розвитку нових підходів та висновків у науковому дослідженні.

Підсумовуючи, визначимо, що КА – це потужний інструмент для моделювання складних систем та явищ. Моделювання за допомогою КА відкриває широкі можливості для досліджень у різних галузях – від біології до комп’ютерних наук. Цей метод може допомогти краще зрозуміти складні системи та явища, а також розробити нові підходи до їх дослідження й управління.

Список використаних джерел

1. Черепина Є. О. Моделювання складних процесів за допомогою клітинних автоматів: текстова частина до курсової роботи за спеціальністю «Інженерія програмного забезпечення» № 121. Київ, 2021. 14 с.

2. Ніколюк П. К. Моделювання систем: навчальний посібник для здобувачів вищої освіти спеціальності 122 Комп'ютерні науки: навч. посіб. Вінниця: ДонНУ імені Василя Стуса, 2023. 41 с.

3. Шабанов Д. А. Моделі на основі клітинних автоматів: онлайн-підручник за курсом «Сотворіння світів: імітаційне моделювання надорганізмових систем в електронних таблицях та R». Розділ 9. URL: https://batrachos.com/Simulation_Cell_Automates

УДК 519.651

Маруняк А. О., *здобувач 2 курсу спеціальності 122 Комп'ютерні науки,*
науковий керівник:
Сеник І. О., *асистент кафедри інформаційних технологій*

АПРОКСИМАЦІЯ ФУНКЦІЙ. МЕТОД НАЙМЕНШИХ КВАДРАТІВ

Донецький національний університет імені Василя Стуса, м. Вінниця

Вступ. Апроксимація функцій є важливою задачею в математичному аналізі та чисельних методах. Це дає нам змогу знаходити наближені значення функцій, що є особливо корисним у випадках, коли точне аналітичне представлення складне або неможливе. Метод найменших квадратів (МНК) є одним із найпопулярніших методів для апроксимації даних [1]. Його застосування варіюється від статистичного аналізу до інженерії та фізики.

Виклад основного матеріалу. Метод найменших квадратів полягає у знаходженні такої функції, яка мінімізує суму квадратів відхилень між експериментальними даними та значеннями апроксимуючої функції. Формально, якщо у нас є набір точок (x_i, y_i), де $i = 1,2,\dots,n$, то ми хочемо знайти функцію $f(x)$, яка мінімізує вираз:

$$S = \sum_{i=1}^{n} (y_i - f(x_i))^2.$$

Зазвичай у якості $f(x)$ вибирають лінійну функцію $f(x) = a_0 + a_1 x$ або поліном вищого степеня [2].

Найпростішим випадком є лінійна регресія, де функція має вигляд $f(x) = a + bx$. Для знаходження коефіцієнтів a і b використовують систему рівнянь:

$$\frac{\partial S}{\partial a} = -2 \sum_{i=1}^{n} (y_i - a - bx_i) = 0;$$

$$\frac{\partial S}{\partial b} = -2 \sum_{i=1}^{n} x_i (y_i - a - bx_i) = 0.$$

Розв'язуючи цю систему, отримуємо значення коефіцієнтів:

$$a = \frac{\sum y_i \sum x_i^2 - \sum x_i \sum (x_i y_i)}{n \sum x_i^2 - \sum (x_i)^2};$$

$$b = \frac{n\sum(x_i y_i) - \sum x_i \sum y_i}{n\sum x_i^2 - \sum(x_i)^2}.$$

Для поліноміальної регресії, наприклад, коли функція $f(x)$ є поліномом другого ступеня $f(x) = a + bx + cx^2$, розв'язання задачі МНК вимагає розв'язання системи лінійних рівнянь вищого порядку. Загальний підхід включає складання матриці Вандермонда [3] для системи рівнянь і розв'язання цієї системи за допомогою методів лінійної алгебри, як-от метод Гаусса [4].

Розглянемо приклад апроксимації набору даних за допомогою МНК. Припустимо, у нас є такі дані:

x	y
1	2,1
2	2,9
3	3,7
4	4,6
5	5,1

Ми можемо побудувати лінійну регресійну модель для цих даних. Використовуючи формули для коефіцієнтів a і b, обчислимо:

$$\sum x_i = 1 + 2 + 3 + 4 + 5 = 15;$$
$$\sum y_i = 2{,}1 + 2{,}9 + 3{,}7 + 4{,}6 + 5{,}1 = 18{,}4;$$
$$\sum x_i^2 = 1^2 + 2^2 + 3^2 + 4^2 + 5^2 = 55;$$
$$\sum (x_i\, y_i) = 1 \cdot 2{,}1 + 2 \cdot 2{,}9 + 3 \cdot 3{,}7 + 4 \cdot 4{,}6 + 5 \cdot 5{,}1 = 69{,}9.$$

Тепер підставимо ці значення у формули:

$$a = \frac{(18{,}4 \cdot 55) - (15 \cdot 69{,}9)}{5 \cdot 55 - 15^2} = \frac{1012 - 1048{,}5}{275 - 225} = \frac{-36{,}5}{50} = -0{,}73;$$

$$b = \frac{5 \cdot 69{,}9 - 15 \cdot 18{,}4}{5 \cdot 55 - 15^2} = \frac{349{,}5 - 276}{275 - 225} = \frac{73{,}5}{50} = 1{,}47.$$

Отже, апроксимуюча пряма має вигляд:

$$f(x) = -0{,}73 + 1{,}47x.$$

Висновки. Метод найменших квадратів є потужним інструментом для апроксимації функцій та аналізу даних. Його основна ідея полягає у мінімізації сумарного квадрата відхилень, що робить його ефективним для роботи з реальними даними, які часто мають випадкові шуми та похибки.

Список використаних джерел

1. Least Squares Method: What It Means, How to Use It, With Examples: вебсайт. URL: https://www.investopedia.com/terms/l/least-squares-method.asp#:~:text=The%20least%20squares%20method%20is%20a%20statistical%20procedure%20to%20find,the%20behavior%20of%20dependent%20variables (дата звернення: 02.05.2024).

2. Сорокін Ю. В. Методи чисельного аналізу. Київ: Вища школа, 2004.

3. Hughes T. The Vandermonde Determinant, A Novel Proof. 2020. URL: https://towardsdatascience.com/the-vandermonde-determinant-a-novel-proof-851d107bd728 (дата звернення: 02.05.2024).

4. Gauss elimination: вебсайт. URL: https://www.britannica.com/science/Gauss-elimination (дата звернення: 02.05.2024).

УДК 004.75:004.77: 004.8

Курдупов О. Л., *здобувач 1 курсу спеціальності 122 Комп'ютерні науки,*
Комаров В. Ф., *канд. техн. наук, старший викладач кафедри інформаційних технологій*

ВИКЛИКИ КІБЕРБЕЗПЕКИ У РОЗУМНИХ МЕДИЧНИХ СИСТЕМАХ

Донецький національний університет імені Василя Стуса, м. Вінниця

Сучасні технології, як-от штучний інтелект (AI) та інтернет речей (IoT), революціонізують медичну галузь, втілюючи концепцію смарт-медицини. Однак інтеграція цих технологій приносить нові виклики у сфері кібербезпеки. Дослідження зосереджується на аналізі викликів безпеки в системах, що використовують технології штучного інтелекту та інтернету речей (AIoT), і методах захисту даних у смарт-медицині.

Труднощі, з якими стикається медичний бізнес, впливають на стандарти лікування пацієнтів. Штучний інтелект у поєднанні з IoT може автоматизувати моніторинг стану здоров'я, оптимізувати роботу медичного персоналу та вдосконалити управління обслуговуванням пацієнтів. AIoT обіцяє трансформувати охорону здоров'я, забезпечуючи точнішу діагностику та персоналізоване лікування.

Для вирішення проблем, підвищення надійності та ефективності системи охорони здоров'я починають використовуватись технології Internet of Medical Thing (IoMT). Наприклад, носимі пристрої можуть надсилати дані про серцевий ритм та активність пацієнта безпосередньо до хмарної платформи для аналізу. Інформація про пацієнтів зберігається в хмарному центрі обробки даних або базі даних. До того ж хмара забезпечує доступ до численних медичних спеціалістів, наприклад до тих, хто проводить медичну діагностику для лікування пацієнтів [1]. Також додатки для охорони здоров'я дають змогу здійснювати віддалений моніторинг за допомогою інтелектуальних пристроїв, як-от смартфони та переносні сенсорні пристрої [2]. Традиційні способи догляду за пацієнтами швидко змінюються завдяки смарт-технологіям.

Однак із розширенням мережі IoT та збільшенням обсягу чутливих медичних даних зростає й потенціал кібератак. Відтак головною проблемою стає забезпечення надійності та конфіденційності медичних даних у розумних медичних системах. Без адекватних заходів безпеки така інтеграція вразлива до порушень даних.

Типовими задачами кібербезпеки у галузі застосування є:

1. Доступність даних: забезпечення безперервного доступу до медичних даних у разі кібератаки є критично важливим для здоров'я пацієнтів.

2. Конфіденційність: неавторизований доступ до медичних даних може призвести до порушення приватності та довіри пацієнтів.

3. Цілісність даних: неправдива інформація внаслідок злому може призвести до неправильного лікування, наражаючи пацієнтів на ризик.

Функції зберігання даних включають зберігання структурованих даних, відеоданих, даних зображень і напівструктурованих даних. Основними проблемами управління даними є індексація, об'єднання, аналіз і візуалізація даних. Важливо відокремлювати та шифрувати конфіденційні дані, а також контролювати доступ, керувати дозволами, робити резервні копії, відновлювати дані та зберігати журнали аудиту [3].

Забезпечити конфіденційність і безпеку даних складно, коли в системі наявний великий обсяг чутливої інформації. Переважна більшість медичного обладнання є вразливою до хакерських атак. IoMT може зберігати записи пацієнтів, контакти та інші клінічні записи, а це безліч пристроїв, наповнених клінічними даними. Медична карта пацієнта вразлива до можливості розголошення через недостатній або неактуальний рівень заходів безпеки.

Рішення IoMT повинні також враховувати загрози, які вони несуть не тільки пацієнтам, але й фінансовій підсистемі медичних установ [4].

Питання спільного використання даних завжди серйозні та мають всебічний вплив на конфіденційність і безпеку як системи, так і всіх зацікавлених сторін. Розмежування та спільне використання даних у смарт-системах є складною задачею в екосистемі AIoT через обсяг даних, що генерується сенсорними пристроями, постійний зв'язок між пристроями в системі та потребу в поєднанні інформації від пацієнтів для розкриття потенціалу смарт-систем [5].

Перспективною є рекомендація використання сучасних криптографічних методів, як-от шифрування даних, у поєднанні з технологіями блокчейн. Це може значно підвищити безпеку систем AIoT та IoMT. Імплементація стандартів і протоколів може вирішити багато зазначених проблем. Важливим в умовах швидкого розвитку смарт-технологій є також впровадження регулярних аудитів безпеки та навчання й актуалізація знань персоналу.

AIoT представляє величезні можливості для смарт-медицини, але супроводжується ризиками у сфері кібербезпеки. Розробка і реалізація комплексних стратегій захисту даних є ключовими для забезпечення успіху цих технологій у майбутньому.

Список використаних джерел

1. Privacy in the internet of things for smart healthcare / D. He, R. Ye, S. Chan, M. Guizani, and Y. Xu. *IEEE Communications Magazine*. 2018. Vol. 56, № 4.

2. Enabling Artificial Intelligence of Things (AIoT) / A. A. Pise, K. K. Almuzaini, T. A. Ahanger, A. Farouk, K. Pant, P. K. Pareek. *Healthcare Architectures and Listing Security Issues.* 2022. DOI: 10.1155/2022/8421434.

3. Artificial Intelligence of Medical Things for Medical Information Systems Privacy and Security / M. Abdulraheem, E. A. Adeniyi, J. B. Awotunde, A. L. Imoize, R. G. Jimoh, I. D. Oladipo, P. B. Falola. 2023. eBook ISBN9781003370321.

4. Al and IoT Enabled Smart Hospital Management Systems / M. K. Gourisaria, R. Agrawal, V. Singh, S. S. Rautaray, M. Pandey. 2022.

5. Ghosh A., Chakraborty D., Law A. Artificial intelligence in Internet of Things. *CAAI Trans.* Intell. Technol. 2018.

УДК 004.42:004.8:519.2

Уманська А. В., *здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,*
Комаров В. Ф., *канд. техн. наук, старший викладач кафедри інформаційних технологій*

ВИКОРИСТАННЯ ТЕОРІЇ ДЕМПСТЕРА–ШАФЕРА В ЗАДАЧАХ КЛАСИФІКАЦІЇ

Донецький національний університет імені Василя Стуса, м. Вінниця

Математична теорія Демпстера–Шафера (DST), яка відома також як теорія свідчень або теорія функцій довіри, надає математично обґрунтований метод для обчислення ймовірності події після комбінування окремих частин вихідної інформації про цю подію, яка може бути отримана з різних джерел.

З моменту появи теорії Демпстера–Шафера розвиваються та узагальнюються її базові положення і концептуальні основи, удосконалюються алгоритми, описуються нові сфери застосування, виявляються та долаються недоліки. Одним із перспективних напрямів є використання теорії DST в задачах класифікації.

Використання теорії Демпстера–Шафера у задачах класифікації може бути раціональним рішенням у разі, коли доводиться мати справу з суттєвою неоднозначністю. Припустимо, ми маємо множину об'єктів та множину всіх можливих властивостей, які вони можуть мати. Потрібно за відомим набором ознак визначити об'єкти, які мають задані властивості. Основна ідея DST полягає в тому, що деяка міра ймовірності може бути віднесена не тільки до окремих елементів множини подій у предметній галузі, але і загалом до певної підмножини цієї множини, коли більш докладний розподіл цієї часткової міри ймовірності за підмножиною невідомий, або є невідомими деякі міри ймовірності, які безпосередньо стосуються окремих елементів множини подій.

Закріплення загальної міри ймовірності за підмножинами множини об'єктів з відомими властивостями може відбуватись з таких причин:

- через неоднозначність рішення задачі експертної класифікації, коли експерт не в змозі точно визначити рівень значимості ознаки для одного, окремо взятого об'єкта;
- через неточність результатів визначення значення самої ознаки;
- через призначення загального рівня довіри для всіх ознак.

Ступінь впевненості в наявності заданої властивості у об'єктів можна висловити функцією довіри:

$$Bel(A_i) = \sum A_j \text{Ì } A_i m(A_j),$$

де A_j – подія, яка відповідає наявності заданої властивості в об'єкта або деякої множини об'єктів, $A_i \text{Ì } C$, C – повна множина подій, яка є результатом рішення задачі, $m(A_j)$ – базовий розподіл ймовірності подій A_j, $m(A_j) \in [0,1]$.

$Bel(A_i)$ має такі властивості: $Bel(\emptyset) = 0$, $Bel(A_i) \in [0,1]$ та $Bel(C) = 1$.

Ступінь наявності заданої властивості в об'єктів можна висловити за допомогою функції:

$$Pl(A_i) = 1 - Bel(A_i) = 1 - \sum m(A_j).$$

За аналогією з теорією ймовірності величини $Bel(A_i)$ та $Pl(A_i)$ можна прийняти як нижню та верхню границі ймовірності наявності заданої властивості об'єктів, які належать підмножині A_iÌ C, припускаючи існування деякої ймовірності $p(A_i)$:

$$Bel(A_i) \le p(A_i) \le Pl(A_i).$$

За правилом Демпстера, об'єднання різних подій з розподілом ймовірностей m_1 та m_2 виконується так:

$$m(C_n) = \frac{1}{1 - m(\emptyset)} \cdot \sum m_1(A_i) \cdot m_2(A_j),$$

де $m(C_n)$ – функція об'єднання подій A_i та A_j, які відповідають за наявність у об'єкта властивостей C_n.

Правило Демпстера асоціативне та комутативне, що дає змогу об'єднати аналогічно більшу кількість свідчень (оцінок).

Рішення, яке можна знайти за допомогою DST, являє собою повну групу подій, що, з одного боку, дає змогу краще оцінити реальне положення речей, з іншого боку, дає можливість уточнення рішення задачі під час появи додаткових ознак. Збільшення кількості ознак приводить до отримання більш точного результату.

Сфери застосування теорії DST дуже різноманітні. Вони простягаються від геоінформаційних до діагностичних систем. Функція довіри є природним інструментом у системах прийняття рішення за багатьма критеріями в умовах високої невизначеності та нестачі вихідних даних, використовування доволі рідкісних подій.

Тематика на напрям досліджень вміщують у себе розвиток поняття функції довіри, аналіз числових характеристик, використання емпіричних моделей, специфічні розширення теорії ймовірності.

У випадках, коли невизначеність пов'язана з варіативністю параметрів, а кількість вихідної інформації дає змогу експерту зробити припущення про тип імовірнісного розподілу, кориснішими можуть бути методи, засновані на вирішенні задач, що вміщують логіку немонотонних міркувань.

Незважаючи на певну особливість використання DST, в кожному конкретному випадку існує потреба в універсальній програмній реалізації. Прикладом такої реалізації може слугувати програмна бібліотека.

Перелік джерел, які можна віднести до розробок українських вчених, можна розширювати, але головні тренди, що пов'язані з DST, вже визначені. Актуальність розробок та застосування методів і алгоритмів з використанням DST у виробничих системах доволі зрозуміла. В якості прикладу можна навести наступні напрями:

- розробка моделей поведінки складних технічних систем;
- робототехніка;

- забезпечення надійності виробничих систем.

Аналіз експертних оцінок – це одна з найпоширеніших галузей застосування DST. Можна виокремити деякі показові напрями:

- аналіз групових експертних оцінок у критичних ситуаціях;
- обробка експертних суб'єктивних оцінок;
- дослідження проблем інтеграції лінгвістичної інформації;
- експертні системи для інформаційної підтримки операторів автоматизованих підприємств.

Також заслуговує на увагу робота, присвячена застосуванню методів контролю під час знаходження дефектів у матеріалах. Можна навести низку конкретних напрямів досліджень та розробок на основі DST в галузі діагностичних систем:

- діагностика хіміко-технологічних процесів;
- контроль виробів у машинобудуванні та енергетиці;
- контроль якості водорозподільних мереж;
- діагностика двигунів;
- діагностика багатоступеневих систем.

Останніми роками теорія DST має активний розвиток та затребуваність у вирішенні практичних задач. Це стосується великої кількості різних наукових та технологічних галузей і напрямів: аналіз та керування ризиками, розв'язання задач класифікації інтервальних даних, системи геолокації, вирішення завдань оптимізації в умовах невизначеності параметрів, гуманітарні сфери тощо.

Є підстави вважати, що розвиток теорії DST у перспективі може дати нові цікаві рішення в різноманітних задачах класифікації у поєднанні з сучасними методами машинного навчання. Отже, теорія залишається актуальною, особливо в галузі розширення граничних умов її застосування, та затребуваною для подальших досліджень.

Список використаних джерел

1. Альперт С. І. Сучасні критерії оцінки точності класифікації аерокосмічних зображень. Київ. 2020. 192 с.
2. Гладунський В. Н. Математика для економістів: означення, формули, приклади: навчальний посібник. Львів. 2023. 632 с.
3. Жлуктенко В. І., Наконечний С. І. Теорія ймовірностей і математична статистика: навчально-методичний посібник у 2-х ч. Ч. І: Теорія ймовірностей. Київ: КНЕУ. 2020. 304 с.
4. Копич І. М., Сороківський В. М. Теорія ймовірностей та математична статистика: навчальний посібник. Київ. 2020. 382 с.
5. Лозовий Б. Н., Пушак Я. С. Теорія ймовірностей і елементи математичної статистики: навчальний посібник. Київ. 2019. 276 с.

УДК 519.2

Чемес В. С., *здобувач 2 курсу спеціальності 122 Комп'ютерні науки,*
Волонтир Л. О., *канд. техн. наук, доцент, доцент кафедри інформаційних технологій*

АНАЛІЗ РИЗИКІВ У ФІНАНСОВОМУ СЕКТОРІ ЗА ДОПОМОГОЮ МЕТОДІВ ТЕОРІЇ ЙМОВІРНОСТЕЙ ТА СТАТИСТИКИ

Донецький національний університет імені Василя Стуса, м. Вінниця

Зростання економічної та фінансової нестабільності підкреслює необхідність проведення аналізу ризиків у фінансовому секторі. Але через їх складність і непередбачуваність розуміння та ефективне управління цими ризиками залишається складною задачею.

Метою цього дослідження є виявлення основних елементів ризиків, що виникають у фінансовому секторі, а також дослідження можливості використання методів, пов'язаних із теорією ймовірностей та статистикою, для аналізу та управління цими елементами.

Ризик – це ймовірність того, що небажана подія або втрата може вплинути на досягнення цілей чи результати певної діяльності. Коли справа доходить до фінансового сектору, ризик може складатися з різних елементів, наприклад, несплати кредиту, непередбачуваних витрат або навіть ймовірності фінансової катастрофи [1, 2].

Існує багато типів ризиків, зокрема кредитний, ринковий, операційний, ризик ліквідності, політичних змін, технологічний ризик та інші. Кожен із цих типів ризиків може мати власні унікальні характеристики та наслідки [2].

Теорія ймовірностей та математична статистика є незамінними інструментами для вирішення ключових завдань в аналізі ризиків. Однією з важливих задач є оцінка ймовірності настання ризикових подій. Використовуючи ймовірнісні моделі, можна визначити ймовірність виникнення конкретних ризикових ситуацій, як-от дефолти або ринкові кризи, що дає змогу фінансовим інститутам краще підготуватися до можливих сценаріїв.

Додатково, ці методи забезпечують кількісну оцінку ризиків. Математичні підходи, як-от Value at Risk (VaR) та Expected Shortfall (ES), дають змогу точно оцінити можливі збитки та ймовірність їх настання, що є важливим для прийняття рішень щодо управління капіталом та резервів.

VaR оцінює максимальний потенційний збиток за певний період із заданою ймовірністю. Формула для VaR для нормального розподілу виглядає так:

$$VaR = \mu - Z \cdot \sigma,$$

де μ – середнє повернення активу;

Z – квантиль нормального розподілу для заданого рівня довіри (наприклад, для 95 % рівня довіри $Z = 1.65$);

σ – стандартне відхилення повернень активу [3].

ES оцінює середній збиток у випадку, якщо збитки перевищують VaR. Для нормального розподілу ES обчислюється як:

$$ES = \frac{\sigma \cdot \phi(Z)}{1-\alpha} - \mu,$$

де $\phi(Z)$ – щільність нормального розподілу в точці Z;

α – рівень довіри (наприклад, для 95 % рівня довіри $\alpha = 0.05$) [4].

Аналіз та прогнозування волатильності є ще одним важливим завданням. Моделі, як-от GARCH, допомагають прогнозувати мінливість ринкових цін, що є ключовим для управління ринковими ризиками, та оцінки потенційної волатильності активів.

Базова GARCH(1,1) модель описується рівняннями:

$$h_t = \omega + \alpha\epsilon_{t-1}^2 + \beta h_{t-1},$$

де h_t – умовна дисперсія в момент часу t;

ω, α, β – параметри моделі;

ϵ_{t-1} – попередня залишкова величина.

До того ж статистичні методи сприяють оптимізації інвестиційного портфеля. Вони дають змогу визначити оптимальну структуру портфеля, яка мінімізує ризики за заданого рівня очікуваної дохідності, враховуючи кореляцію між активами.

Сценарний аналіз та стрес-тестування також є важливими аспектами, де теорія ймовірностей відіграє ключову роль. Моделювання різних економічних та ринкових сценаріїв дає змогу оцінити стійкість фінансових інститутів до зовнішніх шоків, що сприяє кращому розумінню впливу різних факторів на фінансову стабільність.

Нарешті, моніторинг та контроль ризиків є невід'ємною частиною фінансового аналізу. Використання статистичних методів дає змогу створювати системи раннього попередження, які допомагають виявляти та реагувати на ризики на ранніх етапах їх розвитку.

Приклад обчислення VaR та ES

Припустимо, у нас є портфель з історичними даними повернення активів за останній рік. Ми маємо такі щомісячні повернення (у відсотках):

–2 %, 1 %, 3 %, –1 %, 4 %, –3 %, 2 %, –2 %, 1 %, 5 %, –1 %, 3 %.

Крок 1: Обчислення середнього повернення та стандартного відхилення.

Середнє повернення (μ):

$$\mu = \frac{-2+1+3-1+4-3+2-2+1+5-1+3}{12} = \frac{10}{12} = 0.833\ \%.$$

Стандартне відхилення (σ) розраховується за формулою:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i-\mu)^2}{n}},$$

де x_i – кожне зі значень вибірки;

μ – середнє арифметичне вибірки;

n – кількість значень вибірки.

Тепер обчислимо сукупне значення цих відхилень та їх середнє значення:

$$\sigma = \sqrt{\frac{75.6667}{12}} = 2.511\ \%.$$

Крок 2: Розрахунок VaR.

Припустимо, ми хочемо розрахувати VaR для рівня довіри 95 %. Квантиль для 95 % рівня довіри нормального розподілу (Z) дорівнює 1.645:

$$VaR = \mu - Z \cdot \sigma = 0{,}833\ \% - 1{,}645 \cdot 2{,}511\ \% = -3{,}2974\ \%.$$

Це означає, що зі ймовірністю 95 % максимальний збиток за місяць не перевищить 3.2974 %.

Крок 3: Розрахунок Expected Shortfall (ES).

Для $Z = 1.645, \phi(1.645) \approx 0.103$.

Оскільки ми розглядаємо 95 % рівень довіри, $\alpha = 0.95$, то:

$$ES = 0{,}833\ \% - \frac{2{,}511\% \cdot 0{,}103}{0{,}05} = 0{,}833\ \% - 5{,}17\ \% = -4{,}3395\ \%.$$

Це означає, що в середньому збиток у випадку, якщо він перевищить VaR, становитиме 4.337 %.

Висновки. Ця робота покликана привернути увагу до різноманітних типів ризиків та демонструє практичну значимість ймовірнісних моделей для можливості оцінки ймовірності настання ризикових подій, як-от дефолти або ринкові кризи. На основі теоретичних відомостей на практичному прикладі продемонстровано використання теорії ймовірностей та математичної статистики для розв'язування спеціалізованих прикладних задач з оцінки ризиків у фінансовому секторі. Наведені в роботі приклади стосуються кількісної оцінки ризиків, як-от Value at Risk (VaR) та Expected Shortfall (ES).

Список використаних джерел

1. Що таке ризики і як вони впливають на якість роботи. *QATestLab.* 03.02.2022. URL: https://training.qatestlab.com/blog/technical-articles/what-are-the-risks-and-how-do-they-affect/ (дата звернення: 15.05.2024).
2. Жигірь А. А. Різновиди підприємницьких ризиків та їх класифікація. *Ефективна економіка.* 2012. № 4. URL: http://www.economy.nayka.com.ua/?op=1&z=1063 (дата звернення: 15.05.2024).
3. What Is Value at Risk (VaR)? How to Calculate It. Composer. Dec. 07 2023. URL: https://www.composer.trade/learn/value-at-risk (дата звернення: 16.05.2024).
4. Expected Shortfall closed-form for Normal distribution. Jérémie Smaga's personal blog. Nov. 6, 2016. URL: https://blog.smaga.ch/expected-shortfall-closed-form-for-normal-distribution/ (дата звернення: 17.05.2024).

УДК 004.6

Поліщук О. С., *здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,*
Поремський Ю. В., *канд. техн. наук, старший викладач кафедри інформаційних технологій*

ЧИСЕЛЬНІ МЕТОДИ РОЗВ'ЯЗАННЯ НЕЛІНІЙНИХ РІВНЯНЬ

Донецький національний університет імені Василя Стуса, м. Вінниця

Нелінійне рівняння – це рівняння, яке включає хоча б один член з нелінійною функцією від невідомої змінної або її похідних. Головна різниця між нелінійними та лінійними рівняннями полягає в тому, що в лінійних рівняннях усі члени є лінійними функціями невідомої змінної та її похідних. До найбільш відомих методів розв'язання нелінійних рівнянь належать метод Ньютона, метод простої ітерації та метод хорд.

Метод Ньютона, який також відомий як метод дотичних, застосовується для розв'язування нелінійних рівнянь. В цьому методі задається інтервал [a,b] з єдиним розв'язком, початкове наближення розв'язку $x \in$ [a,b] та задана точність $\varepsilon > 0$.

На кожній ітерації методу поточне наближення x_k обчислюється. Наступне наближення x_{k+1} визначається шляхом проведення дотичної до графіка функції $f(x)$, у точці $x_k f(x_k)$, і визначення точки перетину цієї дотичної з віссю x. Пошук розв'язку припиняється, коли значення функції $f(x_k)$ стає меншим за задану точність ε.

Для визначення цієї точки перетину використовуємо формулу, яка дає змогу знаходити корені рівняння:

$$x_{k+1} = x_k - \frac{f(x_k)}{f`(x_k)}, \qquad (k = 0, 1, 2, \dots).$$

Перед застосуванням методу Ньютона необхідно впевнитися в деяких умовах:

1) необхідно знайти відрізок, на якому знаходиться шуканий корінь рівняння;
2) переконатися, що на цьому відрізку знаки першої та другої похідних не змінюються. У випадку зміни знаків потрібно зменшити розмір початкового відрізка;
3) обрати будь-яку початкову точку в межах відрізка ізоляції, де виконується умова $f(x) * f''(x) > 0$.

Метод простої ітерації – метод обчислення нерухомої точки функції, один з методів наближеного розв'язування інтегральних лінійних рівнянь. Головна ідея методу полягає в тому, щоб перетворити нелінійне рівняння на ітераційний процес, у якому послідовно обчислюються значення змінної до досягнення необхідної точності або досягнення максимальної кількості ітерацій. Це досягається шляхом вибору початкового наближення та застосування ітеративної формули, яка в кожній ітерації покращує наближення до розв'язання рівняння. Такий процес дає змогу знаходити наближений розв'язок навіть для складних нелінійних рівнянь, які не мають аналітичних розв'язків.

Метод простої ітерації полягає в тому, що рівняння $f(x) = 0$ приводиться до вигляду:

$$x = \varphi(x),$$

тоді ітерації виконуються за правилом:

$$x_{n+1} = \varphi(x_n), \qquad (n = 0, 1, \dots),$$

причому задається початкове наближення x_0.

Коли обчислювальний процес збігається до кореня α рівняння $f(x) = 0$, тобто $\alpha = \varphi(\alpha)$, за умови, що функція $\varphi(x)$ визначена і неперервна на відрізку, де знаходиться корінь, можна встановити зв'язок між похибками обчислень на двох сусідніх ітераціях.

Різниця між наближенням x_{n+1} і справжнім значенням кореня α рівняння може бути приблизно виражена як $\varepsilon_{n+1} = \varphi(x_n) - \varphi(\alpha)$. Це може бути приблизно записано як $\varepsilon_{n+1} \approx \varphi`(x^*)x_n$, де x^* знаходиться між x_n і α.

Цей метод є простим у реалізації та ефективним для багатьох типів нелінійних рівнянь. Однак вибір підходящої функції перетворення може бути складним завданням, і неправильний вибір може призвести до незбіжності методу [2].

Приклад методу ітерації:

Маємо рівняння $x^2 - 4x + 3 = 0$. Ми можемо застосувати метод простої ітерації, переписавши це рівняння у вигляді $x = \varphi(x)$.

Запропонуємо таку функцію перетворення: $\varphi(x) = \frac{x^2+3}{4}$.

Розглянемо процес ітерації:

1) обираємо початкове наближення x_0;
2) застосовуємо функцію перетворення: $x_1 = \varphi(x_0)$;
3) повторюємо крок 2 до досягнення необхідної точності або досягнення максимальної кількості ітерацій.

Для початку виберемо $x_0 = 0$ як початкове наближення:

$$x_1 = \varphi(x_0) = \frac{0^2 + 3}{4} = \frac{3}{4} = 0{,}75;$$

$$x_2 = \varphi(x_1) = \frac{(0{,}75)^2 + 3}{4} = \frac{3{,}5625}{4} = 0{,}890625;$$

$$x_3 = \varphi(x_2) = \frac{(0{,}890625(^2 + 3}{4} = \frac{3{,}793212890625}{4} = 0{,}94830322265625.$$

Далі продовжуємо ітерувати, поки не досягнемо потрібної точності або не виконаємо задану кількість ітерацій.

Метод Ньютона та метод простої ітерації є двома ефективними числовими методами для розв'язання нелінійних рівнянь.

Метод Ньютона зазвичай збігається швидше за метод простої ітерації, особливо у випадках, коли корінь розташований близько до початкового наближення. Проте для його застосування потрібно обчислювати похідну функції, що може бути складним завданням, та враховувати особливості вибору початкового наближення, оскільки неправильний вибір може призвести до незбіжності методу.

З іншого боку, метод простої ітерації є більш простим у реалізації і не вимагає обчислення похідної функції. Він може збігатися повільніше, але зазвичай менш схильний до незбіжності через вибір відповідної функції перетворення та початкового наближення.

Отже, обидва методи мають свої переваги та недоліки, і вибір між ними зазвичай залежить від конкретних умов задачі та потреб точності.

Список використаних джерел

1. Чисельні методи: навчальний посібник / Л. О. Волонтир, О. В. Зелінська, Н. А. Потапова, І. А. Чіков. *Вінницький національний аграрний університет.* Вінниця: ВНАУ, 2020. 322 с.
2. Фельдман Л. П., Петренко Ф. І., Дмитрієва О. А. Чисельні методи в інформатиці. Київ. Видавнича група BNV, 2006. 480 с.

УДК 004.02

Бадіка М. М., *здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,*
науковий керівник:
Римар П. В., *старший викладач кафедри інформаційних технологій*

РОЛЬ СЕРВІСІВ ХМАРНИХ ОБЧИСЛЕНЬ У СУЧАСНІЙ ІНДУСТРІЇ

Донецький національний університет імені Василя Стуса, м. Вінниця

Сервіси хмарних обчислень набули значного значення в індустріальному ландшафті, надаючи компаніям можливість ефективно використовувати обчислювальні ресурси без необхідності власної інфраструктури. Актуальність цих досліджень щодо визначення ролі сервісів хмарних обчислень у сучасній індустрії обумовлена розвитком цифрового простору та сучасними інформаційними технологіями.

Дефініція «хмара» у системі прикладних та інформаційних технологій ототожнюється з хмарними обчисленнями або хмарними сервісами, що використовують обчислювальні ресурси і доступні через інтернет, зокрема обчислювальна потужність, зберігання даних та програмне забезпечення. У сфері хмарних обчислень користувачі не мають необхідності безпосередньо володіти або керувати фізичним обладнанням, натомість вони можуть споживати ресурси за потребою та оплачувати лише за фактичне їх використання.

Ідею хмарних обчислень можна простежити до 1960-х років, коли вчений-інформатик Дж. С. Р. Ліклайдер запропонував концепцію «Міжгалактичної комп'ютерної мережі». Ця ідея й заклала основу для майбутнього інтернету. Багато авторів визначають принцип поділу часу (англ. *time-sharing*) у використанні комп'ютерів у 1960–1970-х роках як ранні хмарні обчислення, однак він відрізняється від сучасного поняття. Хмарні обчислення стали можливими й набули популярності, коли інтернет став швидким і надійним [3].

Хмарні обчислення почали розвиватися у 2000-х роках, проте їх популярність значно зросла останніми десятиліттями. Вони забезпечують доступ до обчислювальних ресурсів через інтернет, що дає змогу компаніям масштабувати свої операції без значних витрат на обладнання та обслуговування.

Хмарні сервіси поділяються на кілька основних категорій залежно від наданих функцій:

1. Інфраструктура як сервіс (IaaS): включає надання обчислювальної потужності та інфраструктурних ресурсів через інтернет. Користувачі можуть отримувати віртуальні машини, зберігання даних та інші інфраструктурні ресурси за потребою.

2. Платформа як сервіс (PaaS): надає платформу для розробки, тестування та розгортання додатків. Користувачі можуть зосередитися на розробці програмного забезпечення, оскільки багато інфраструктурних аспектів вже реалізовано для них.

3. Програмне забезпечення як сервіс (SaaS): надає доступ до вже готового програмного забезпечення через Інтернет. Користувачі можуть використовувати програми без необхідності їх встановлення та оновлення.

4. Хмарні зберігальні системи: ці сервіси дають змогу зберігати та керувати даними в хмарі, забезпечуючи можливість резервного копіювання, синхронізації та спільного використання файлів.

5. Хмарні послуги обробки даних: дають змогу виконувати обробку та аналіз даних у хмарі, зазвичай використовуючи спеціалізовані інструменти та фреймворки.

6. Хмарні послуги для розробників (DevOps): дають змогу розробникам та командам керувати, автоматизувати та моніторити різні етапи розробки програмного забезпечення [1].

Розглянемо переваги та недоліки використання сервісів хмарних обчислень у сучасній індустрії. Перевагами є:

- зменшення витрат на обладнання та обслуговування інфраструктури;
- можливість швидко масштабувати ресурси залежно від потреб;
- підвищення доступності даних та послуг завдяки глобальній мережі серверів;
- забезпечення більшої гнучкості та мобільності для робочих процесів.

До недоліків треба віднести:

- збереження конфіденційності даних: передача та зберігання конфіденційних даних у хмарному середовищі може стати об’єктом атак з боку зловмисників. Недостатня захищеність може призвести до витоку конфіденційної інформації;
- вартість: хоча використання хмарних сервісів може допомогти зменшити витрати на обладнання та обслуговування, вартість підписки на ці послуги може стати значним фактором для деяких підприємств, особливо у довгостроковій перспективі;
- проблеми з безпекою даних: незважаючи на високий рівень безпеки, який забезпечують провайдери хмарних послуг, існують ризики порушення безпеки даних через помилки в конфігурації, соціальну інженерію або недостатні заходи безпеки з боку користувачів.

Хоча сервіси хмарних обчислень мають безліч переваг, вони стикаються з низкою ризиків. Забезпечення безпеки даних, надійності та доступності, управління обліковими записами та вартість використання є серйозними викликами, які вимагають уваги та вирішення.

З нескінченним розвитком технологій сервіси хмарних обчислень відкривають широкі перспективи для підприємств. Вони забезпечують гнучкість, мобіль-

ність, можливості масштабування, розширення географічного охоплення та стратегічне партнерство, сприяючи розвитку інноваційних рішень та стимулюючи ріст бізнесу.

Отже, сервіси хмарних обчислень відіграють ключову роль у сучасній індустрії, надаючи підприємствам можливість ефективно використовувати обчислювальні ресурси та підвищувати їх конкурентоспроможність. Незважаючи на це, використання хмарних сервісів вимагає уваги до викликів та можливостей, що можуть виникати під час їх впровадження та використання.

З розвитком технологій та вдосконаленням підходів до безпеки й оптимізації сервіси хмарних обчислень продовжать займати центральне місце в індустріальному ландшафті, впливаючи на розвиток бізнесу та інновацій.

Список використаних джерел

1. Що таке хмарні технології? Переваги та недоліки хмарних сервісів. *Офіційний сайт Edin.* URL: https://edin.ua/shho-take-xmarni-texnologi%D1%97-i-navishho-voni-potribni/ (дата звернення: 29.04.2024).

2. Бунке О. Переваги хмарних технологій при використанні у Internet of Things (IoT). *Технічні науки та технології.* 2019. № 1. С. 127–133.

3. Костін К. Хмарні обчислення: історія, можливості, перспективи. *Офіційний сайт Anywhereclub.* URL: https://aw.club/global/uk/blog/cloud-computing-history-opportunities-benefits (дата звернення: 29.04.2024).

4. Белов Д. Хмарні сервіси: які тренди впливатимуть на розвиток ринку у 2023 році. *Офіційний сайт SPEKA.* 25 січня 2023. URL: https://speka.media/trendi-yaki-vplivayut-na-rozvitok-rinku-xmarnix-servisiv-u-2023-9dyze9 (дата звернення: 29.04.2024).

УДК 004.6

Балюра Б. П., *здобувач 2 курсу спеціальності 122 Комп'ютерні науки,*
науковий керівник:
Горяшин А. С., *асистент кафедри інформаційних технологій*

ЗАСТОСУВАННЯ МЕТОДУ ХОРД У РОЗВ'ЯЗАННІ СИСТЕМ НЕЛІНІЙНИХ АЛГЕБРАЇЧНИХ РІВНЯНЬ У ФІНАНСОВІЙ АНАЛІТИЦІ

Донецький національний університет імені Василя Стуса, м. Вінниця

Фінансова аналітика в сучасному світі відіграє ключову роль у прийнятті стратегічних рішень, які мають значний вплив на економічні процеси та ринкові умови. Однак багато задач у цій області вимагають розв'язання складних систем нелінійних алгебраїчних рівнянь, що часто виникають у зв'язку з моделюванням фінансових процесів, оцінкою ризиків, управлінням портфелем та інвестиційним аналізом. Знаходження точних розв'язків цих систем є важливим завданням для фахівців у галузі фінансів, оскільки від цього залежить ефективність прийнятих рішень та рентабельність інвестицій. Однак через складність та неоднорідність таких систем традиційні аналітичні методи не завжди є дієвими для їх розв'язан-

ня. Тому великого значення набувають числові методи, які допомагають ефективно наближати розв'язки систем нелінійних рівнянь. Один із таких методів – метод хорд. Цей метод базується на ідеї лінійної апроксимації функції ітераційними процесами та дає змогу знаходити корені рівнянь, навіть у випадках, коли аналітичні методи стають непрактичними.

Метод хорд – це числовий метод для знаходження наближеного розв'язку нелінійних рівнянь. Основна ідея методу полягає в тому, щоб замінити нелінійне рівняння лінійним шляхом побудови секансів – прямих лінійних відрізків, які з'єднують дві точки на графіку функції. Далі метод хорд використовує ітераційний процес, щоб апроксимувати корінь шляхом обчислення перетину секансу з віссю абсцис.

Принцип роботи методу хорд такий:

1. Обираються дві початкові точки x_0 і x_1, які лежать по різні сторони від шуканого кореня рівняння.

2. Проводиться пряма, яка проходить через ці дві точки і перетинає вісь абсцис у точці x_2, що є наближенням кореня.

3. Повторюються кроки 1 і 2 з використанням нової пари точок x_1 і x_2, доки не досягнута задана точність або обмежена кількість ітерацій.

Порівняно з іншими числовими методами, як-от метод Ньютона або метод дотичних, метод хорд може бути менш ефективним у деяких випадках через його повільну збіжність. Однак він може бути більш стійким у випадках, коли функція має вузький діапазон збіжності.

Метод хорд базується на принципі інтерполяції Лагранжа. Згідно з цим принципом, секанс може бути розглянутий як лінійне наближення до функції між двома точками. Застосовуючи інтерполяційний метод, можна отримати аналітичний вираз для точки перетину секансу з віссю абсцис. Далі шляхом ітераційного підходу можна знаходити все ближчі наближення до кореня, збільшуючи точність розв'язку [1].

У фінансовій аналітиці часто виникають складні системи нелінійних рівнянь, які моделюють різноманітні фінансові процеси та інвестиційні стратегії. Наприклад, рівняння для обчислення вартості фінансових інструментів, як-от опціони чи облігації, можуть бути нелінійними через умови виплат та динаміку ринку. Також управлінням портфелем та ризиками часто супроводжуються складні рівняння, що враховують стохастичні процеси та оптимізаційні умови, які можуть бути нелінійними.

Приклади застосування методу хорд для розв'язання конкретних фінансових задач:

1. Оцінка вартості опціонів. Метод хорд може бути застосований для обчислення ціни опціонів, яка часто визначається шляхом розв'язання нелінійного рівняння, наприклад, рівняння Блека–Шоулза для європейських опціонів.

2. Моделювання фінансових потоків. У фінансових моделях часто виникають складні системи рівнянь для розрахунку дисконтування майбутніх грошових потоків чи визначення оптимального розподілу активів у портфелі.

3. Оптимізація інвестиційних стратегій. Метод хорд може бути використаний для розв'язання рівнянь, які виникають під час оптимізації інвестиційних

стратегій, як-от рівняння для максимізації очікуваної доходності за певних обмежень на ризик.

Переваги використання методу хорд у фінансовій аналітиці включають його простоту та інтуїтивність, що дають змогу швидко отримувати наближені розв'язки нелінійних рівнянь. До того ж метод хорд може бути менш чутливим до початкового наближення, порівняно з іншими методами, як-от метод Ньютона. Проте метод хорд може мати обмеження у швидкості збіжності, особливо у випадку, коли функція має складну структуру або вузький діапазон збіжності. Також для деяких складних фінансових моделей метод хорд може виявитися менш ефективним, порівняно з іншими числовими методами, як-от метод Ньютона чи метод дотичних [2].

Один із практичних прикладів застосування методу хорд у фінансовій аналітиці – це розрахунок внутрішньої ставки дохідності (IRR) для інвестиційного проєкту. Припустимо, ми маємо такі дані:

– інвестиція: \$100 000;

– очікувані виплати: \$30 000 на другий рік, \$40 000 на третій рік, \$50 000 на четвертий рік.

Ми хочемо знайти внутрішню ставку дохідності (IRR), яка робить чисту сучасну вартість (NPV) проєкту рівною нулю. Для цього ми можемо скористатися методом хорд. Почнемо з двох початкових значень ставки дисконту: $IRR_1 = 0.1$ (10 %) та $IRR_2 = 0.2$ (20 %). Застосовуємо метод хорд для знаходження кореня рівняння NPV, ітеративно обчислюючи нові значення ставки дисконту, що наближаються до значення IRR. Повторюємо цей процес до досягнення заданої точності або фіксованої кількості ітерацій. Коли досягнемо необхідної точності, значення ставки дисконту, за якого NPV стає нульовим, буде оцінкою внутрішньої ставки дохідності (IRR) для цього проєкту [3].

Приклад реалізованого процесу, написаний на мові Python, який обчислює NPV та застосування методу хорд для знаходження кореня рівняння, наведено на рис. 1.

```
def npv(rate, cashflows):
    return sum([cf / (1 + rate) ** i for i, cf in enumerate(cashflows)])
def irr(cashflows, guess1=0.1, guess2=0.2, tol=0.0001, max_iter=100):
    for i in range(max_iter):
        f_guess1 = npv(guess1, cashflows)
        f_guess2 = npv(guess2, cashflows)
        new_guess = guess2 - f_guess2 * (guess2 - guess1) / (f_guess2 - f_guess1)
        if abs(new_guess - guess2) < tol:
            return new_guess
        guess1 = guess2
        guess2 = new_guess
    return None
cashflows = [-100000, 30000, 40000, 50000]
irr_rate = irr(cashflows)
if irr_rate is not None:
    print("Внутрішня ставка дохідності (IRR): {:.2%}".format(irr_rate))
else:
    print("Не вдалося знайти внутрішню ставку дохідності.")
```

Рис. 1. Застосування методу хорд для знаходження кореня рівняння під час обчислення NPV

Результат:

```
Внутрішня ставка доходності (IRR): 8.90%
```

Метод хорд виявляється швидким та простим інструментом для розв'язання систем нелінійних алгебраїчних рівнянь у фінансовій аналітиці, здатним забезпечити швидке наближене розв'язання. Проте він може бути менш ефективним у складних моделях, тому рекомендується використовувати його разом з іншими методами для досягнення більшої точності.

Список використаних джерел

1. Kincaid D., Cheney W. Numerical Analysis: Mathematics of Scientific Computing. American Mathematical Society. 2002.
2. Brigham E. F., Houston J. F. Fundamentals of Financial Management. Cengage Learning. 2018.
3. Hull J. C. Options, Futures, and Other Derivatives. Pearson Education. 2017.

УДК 004.6

Клименко А. Р., *здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,*
науковий керівник:
Фриз І. В., *канд. фіз.-мат. наук, старший викладач кафедри інформаційних технологій*

НАБЛИЖЕНІ МЕТОДИ РОЗВ'ЯЗАННЯ СИСТЕМ ЛІНІЙНИХ РІВНЯНЬ У ПРИКЛАДНИХ ДОСЛІДЖЕННЯХ

Донецький національний університет імені Василя Стуса, м. Вінниця

Системи лінійних рівнянь є основними складниками різних галузей прикладних досліджень, включно з інженерією, фізикою, економікою та ін. Ці системи часто є великими та складними, що робить точні методи розв'язання непрактичними. Відповідно, щоб полегшити роботу з великою кількістю даних, були розроблені наближені методи, які ефективно надають розв'язання з прийнятною точністю.

Найбільш приваблива особливість ітераційних (наближених) методів полягає в тому, що весь обчислювальний процес зводиться до повторюваної послідовності відносно простих дій (ітераційних кроків), як-от множення матриці на матрицю або матриці на вектор.

Основною ідеєю методів наближеного розв'язання є те, що рішення системи:

$$Ax = b,$$

де A – матриця коефіцієнтів, x – вектор невідомих, b – вектор вільних членів, знаходиться як границя послідовних наближень $x^{(n)}$ при $n \to \infty$, де n – номер ітерації. Щоб застосувати ітераційні методи, необхідно задати початкове значення невідомих $x^{(0)}$ і точності обчислень $\varepsilon > 0$. Обчислення проводяться доти, поки не буде виконана оцінка:

$$\left\| x^{(n)} - x^{(n-1)} \right\| \leq \varepsilon.$$

Головна перевага наближених методів – точність шуканого рішення задається.

Для досягнення заданої точності ε, необхідне число ітерацій $n = n(\varepsilon)$. Воно є основною оцінкою якості методу, оскільки за допомогою цього числа порівнюють різні методи за часовими характеристиками.

Основний недолік цих методів полягає в питанні збіжності ітераційного процесу, оскільки це вимагає окремого дослідження. Наразі доведені теореми про збіжність наближених методів дійсні лише для систем, що мають обмеження на матриці [1].

Основні ітераційні методи включають: метод Якобі, метод Гауса–Зейделя і послідовної верхньої релаксації. В їх основі лежить систематичне уточнення значень змінних, що були задані на початку розрахунку [2].

Кожен із цих методів має свої особливості та застосування. Тож давайте розглянемо їх точніше.

Метод Якобі оновлює кожен компонент вектора розв'язку незалежно, на основі попередньої ітерації. На початку розв'язання маємо вже раніше згадану систему лінійних рівнянь $Ax = b$. Розділимо матрицю коефіцієнтів A на діагональну (D), нижню трикутну (L) та верхню трикутну (U) частини:

$$A = D + L + U.$$

Переписуємо систему рівнянь, виокремлюючи діагональну частину (D):

$$Dx = b - (L + U)x.$$

З огляду на це ітераційна формула розв'язання для кожного компонента вектора x на ітерації матиме вигляд:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} (b_i - \sum_{j \neq i} a_{ij} x_j^{(k)}).$$

Метод Якобі простий у реалізації, але він повільно збігається і вимагає діагональної домінантності матриці A для гарантованої збіжності.

Метод Гауса–Зейделя є удосконаленням методу Якобі і оновлює кожен компонент послідовно, використовуючи останні доступні значення у поточній ітерації. Вектор невідомих значень x можна виразити через попередні та поточні наближення:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} (b_i - \sum_{j<i} a_{ij} x_j^{(k+1)} - \sum_{j>i} a_{ij} x_j^{(k)}).$$

Цей метод загалом збігається швидше за метод Якобі, але також вимагає діагональної домінантності матриці для забезпечення збіжності.

Метод послідовної верхньої релаксації (SOR) є розширенням методу Гауса–Зейделя, вводячи фактор релаксації ω для прискорення збіжності. Правильний вибір ω є критичним для оптимізації продуктивності, і він ефективний для певних типів матриць. Вектор невідомих знаходиться за формулою:

$$x_i^{(k+1)} = (1 - \omega) x_i^{(k)} + \frac{\omega}{a_{ii}} (b_i - \sum_{j<i} a_{ij} x_j^{(k+1)} - \sum_{j>i} a_{ij} x_j^{(k)}).$$

Метод SOR може значно прискорити збіжність, порівняно з методом Гауса–Зейделя, завдяки введенню параметра релаксації ω, який оптимізує процес ітерацій. Значення ω зазвичай вибирається експериментально або за допомогою аналізу [3].

У прикладних дослідженнях наближені методи широко застосовуються в різних галузях. У структурній інженерії вони є важливими для аналізу великих структур, як-от мости та будівлі. Методи, як-от Гауса–Зейделя або SOR, використовуються для моделювання потоку рідини в реальному часі, надаючи критичні дані для аеродинаміки та прогнозування погоди. В економіці моделі, що включають кілька змінних і обмежень, часто приводять до великих лінійних систем, де ітераційні методи допомагають у наближеному розв'язанні задач рівноваги, оптимізації розподілу ресурсів та прогнозуванні економічних тенденцій. У машинному навчанні, особливо під час навчання великомасштабних лінійних моделей, приблизні методи застосовуються для обробки масивних наборів даних.

Методи наближеного розв'язання мають свої переваги та обмеження. Основні переваги включають: *масштабованість*, оскільки вони можуть обробляти дуже великі системи, розв'язання яких є нездійсненною задачею для точних методів; *ефективність,* оскільки ітераційні методи можуть збігатися до розв'язання швидше, ніж прямі методи, особливо для розріджених систем; *гнучкість*, адже ці методи можуть бути адаптовані та поєднані з різними техніками для покращення продуктивності.

Основні обмеження включають: *проблеми збіжності*, оскільки не всі ітераційні методи гарантують збіжність, і їх продуктивність сильно залежить від властивостей матриці; *точність,* адже наближені методи надають розв'язки з компромісом у точності, що може бути неприйнятним для застосування; *чутливість до параметрів,* оскільки методи, як-от SOR, вимагають ретельного налаштування параметрів для досягнення оптимальної продуктивності, додаючи складності до їх реалізації [4].

Наближені методи розв'язання систем лінійних рівнянь є незамінними в прикладних дослідженнях завдяки їх здатності ефективно обробляти великі та складні задачі. Ітераційні техніки, як-от методи Якобі та Гауса–Зейделя, пропонують надійні розв'язання у різних галузях. Хоча ці методи мають свої обмеження, їх переваги в масштабованості й ефективності роблять їх важливими інструментами в обчислювальному арсеналі дослідників та практиків.

Список використаних джерел

1. Квєтний Р. Н. Методи комп'ютерних обчислень: навчальний посібник. Вінниця: ВДТУ, 2001. 148 с.

2. Чисельні методи: навчальний посібник / Л. О. Волонтир, О. В. Зелінська, Н. А. Потапова, І. А. Чіков. *Вінницький національний аграрний університет.* Вінниця: ВНАУ, 2020. 322 с.

3. Шахно С. М., Дудикевич А. Т., Левицька С. М. Практична реалізація чисельних методів лінійної алгебри: навчальний посібник. Львів: Видавничий центр ЛНУ імені Івана Франка, 2009. 137 с.

4. Чисельні методи: конспект лекцій / О. В. Шебаніна, С. І. Тищенко, І. І. Хилько, В. О. Крайній. Миколаїв: МНАУ, 2023. 100 с.

СЕКЦІЯ 2
АЛГОРИТМІЗАЦІЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

УДК 004.42:519.2

Шевцов М. В., *здобувач вищої освіти 2 курсу спеціальності 122 Комп'ютерні науки,*
Комаров В. Ф., *канд. техн. наук, старший викладач кафедри інформаційних технологій*

ГЕНЕРАЦІЯ ПСЕВДОВИПАДКОВИХ ЧИСЕЛ У СУЧАСНІЙ C++

Донецький національний університет імені Василя Стуса, м. Вінниця

У сучасному світі програмування генерація псевдовипадкових чисел відіграє ключову роль у різноманітних областях, від розробки ігор до криптографії та моделювання. Їх використання є необхідним елементом для створення випадкових подій та реалізації алгоритмів, які вимагають елемента випадковості.

Стандартна можливість генерування випадкових чисел з'явилася ще в мові програмування C, а потім була успадкована C++. Але можна зробити припущення, що зі стрибком технологій вимоги до стандартного алгоритму стали вищими. Це призвело до того, що функцію стали вважати застарілою і недостатньо ефективною. Але що саме не влаштувало сучасних кодерів?

Припустимо, що розробник хотів обрахувати суму мільйона випадкових елементів і був обмежений використанням функції std::rand(). В такому випадку розробник зіткнувся б із кількома проблемами. Запустивши цей код декілька разів, він міг помітити, що сума не дуже змінюється від запуску до запуску і завжди дорівнює значенню, близькому до 16'000, яке є достатньо малим для суми з 1 000 000 елементів. Проблема полягала у значенні макросу RAND_MAX, який визначає обмеження на максимальне число, яке дорівнює 32'767 (максимальному значенню змінної типу short integer). Це означає, що випадкове значення буде повторюватись кожні 32'767 ітерацій. До того ж сам алгоритм використовував лінійний конгруентний метод для обрахування наступного числа [1]. Такий метод є легко передбачуваним, тобто, принаймні, деякі біти згенерованого за його допомогою наступного числа доволі просто передбачити.

Усвідомивши перераховані проблеми, розробники почали шукати альтернативу, і так з'явився файл <random>, який став стандартною бібліотекою у стандарті C++11. Це значно потужніший інструмент, який містить 3 класи рушіїв випадкових чисел, кожен з яких реалізує в собі алгоритм, за допомогою якого обраховується наступний елемент. Для створення об'єкта рушія зазвичай необхідно вказувати багато шаблонних параметрів, що є максимально непрактичним. Тому були створені спеціальні псевдоніми зі вже підставленими найбільш використовуваними параметрами, які отримали назву генераторів.

На прикладі генератора std::mt19937 розглянемо переваги використання файлу <random>:

1. Максимальне значення, яке може бути згенеровано в такий спосіб, дорівнює 4'294'967'295 (тобто максимальному значенню 32-бітного числа) [2].

2. Обрахування наступного елементу відбувається значно швидше, ніж у std::rand().

3. Цей генератор використовує для генерації наступного числа алгоритм «Вихор Мерсенна», що ускладнює ймовірність передбачити це число [2].

4. Період повторення згенерованих чисел дорівнює $2^{19937} - 1$ [2].

Але одного лише генератора не достатньо: також потрібно обрати певний закон, що описуватиме значення випадкової величини та ймовірності їх появи [3]. Цей закон називається розподілом, і файл <random> містить реалізації найбільш популярних його видів, до яких входять 4 різновиди розподілу Бернуллі, 5 різновидів розподілів Пуассона, розподіли Фішера, Ст'юдента, Коші та багато інших. Є також 2 універсальні розподіли, які створюють цілі або дійсні значення, рівномірно розподілені в діапазоні. Щоб результати генерації не повторювались, як і у випадку функції std::rand(), треба задати «зерно», від якого буде відштовхуватись алгоритм під час генерації наступного числа. У мові С та версіях С++ до 11-го стандарту для цього часто використовували бібліотеку <ctime>, але після 11-го стандарту був доданий файл для сучасної роботи з часом під назвою <chrono>. В якості зерна можна використовувати звичайну константу типу unsigned, але підхід з файлом <chrono> підвищує непередбачуваність результатів.

Із прикладом застосування генераторів для рандомізованого алгоритму пошуку опорного елементу у сортуванні QuickSort, розробленого автором цієї роботи, можна ознайомитись у репозиторії [4].

Отже, ми розглянули методи генерування псевдовипадкових чисел у застарілому та сучасному стилях і визначили плюси та мінуси використання обох із них. Очевидно, що генерація з використанням std::rand() є значно передбачуванішою, має суттєві обмеження на період повторення та розмір генерованих чисел. З іншого боку, метод із використанням сучасних методів із файлів <random> та <chrono> надає істотно новий рівень контролю та непередбачуваності в процесі генерації випадкових чисел та робить період генерації повторень майже недосяжним.

Список використаних джерел

1. C++23 standard(ISO/IEC JTC1 SC22 WG21 N 4860).26.6 Random number generation.
2. Жильцов О. Б. Теорія ймовірностей та математична статистика у прикладах і задачах. Київський столичний університет імені Бориса Грінченка. 2015, 336 с.

УДК 004.42

Балюра Б. П., *здобувач 2 курсу спеціальності 122 Комп'ютерні науки,*
Вєтров О. С., *старший викладач кафедри інформаційних технологій*

ПОРІВНЯЛЬНИЙ АНАЛІЗ ЕФЕКТИВНОСТІ АЛГОРИТМІВ СОРТУВАННЯ QUICK SORT, HEAP SORT, SHELL SORT

Донецький національний університет імені Василя Стуса, м. Вінниця

Сучасний світ інформаційних технологій невіддільний від обробки та аналізу даних. Алгоритми сортування є одними з основних інструментів у цьому процесі, які забезпечують ефективне управління даними у різних застосуваннях, від програмування до наукових досліджень. Серед численних алгоритмів сортування вирізняються Quick sort, heap sort та shell sort, які пропонують різні підходи до вирішення завдань сортування.

Метою цього дослідження є проведення порівняльного аналізу ефективності алгоритмів сортування Quick sort, heap sort та shell sort. Цей аналіз дасть змогу з'ясувати, який із цих алгоритмів краще справляється зі сортуванням різних типів даних за визначеними критеріями. Об'єктом дослідження є визначення особливостей кожного алгоритму та їх вплив на швидкодію й ефективність сортування.

Алгоритми сортування відіграють ключову роль у різних областях комп'ютерних наук і програмування, забезпечуючи ефективне управління даними.

Короткий опис та принцип дії трьох алгоритмів сортування: Quick sort, heap sort та shell sort.

1. Quick sort (Швидке сортування) є одним з найпопулярніших алгоритмів сортування. Він базується на принципі «розбиття та коригування». Принцип дії: алгоритм обирає певний елемент масиву, який називається опорним (pivot), та розбиває масив на два підмасиви: один з елементами менше опорного, а інший – з елементами більше опорного. Потім рекурсивно сортується кожен із цих підмасивів.

2. Heap sort (Сортування купою) базується на використанні структури даних, відомої як купа (heap). Алгоритм будує максимальну купу (для сортування у порядку зростання) або мінімальну купу (для сортування у порядку спадання), а потім послідовно видаляє корінь купи (найбільший або найменший елемент) і перебудовує купу [1].

3. Shell sort (Сортування Шелла) є розширенням алгоритму вставки (insertion sort) і належить до категорії алгоритмів інкрементного сортування. Алгоритм використовує послідовність «кроків», що визначають, як відбувається процес перегляду та перестановки елементів. Спочатку він застосовує звичайне сортування вставками для елементів, що знаходяться на відстані одного кроку один від одного, а потім зменшує цей крок і повторює процес доти, поки весь масив не буде відсортований [2].

Порівняємо три алгоритми сортування – Quick sort, Heap sort та Shell sort – у одній ситуації. Розглянемо ситуацію, коли потрібно відсортувати великий обсяг даних, наприклад, інформацію про транзакції великого банку.

Приклад порівняння трьох алгоритмів сортування на мові Python:

```
import random
import time
def quick_sort(arr):
    if len(arr) <= 1:
        return arr
    pivot = arr[len(arr) // 2]
    left = [x for x in arr if x < pivot]
    middle = [x for x in arr if x == pivot]
    right = [x for x in arr if x > pivot]
    return quick_sort(left) + middle + quick_sort(right)
def heap_sort(arr):
    import heapq
    heapq.heapify(arr)
    return [heapq.heappop(arr) for _ in range(len(arr))]
def shell_sort(arr):
    n = len(arr)
    gap = n // 2
    while gap > 0:
        for i in range(gap, n):
            temp = arr[i]
            j = i
            while j >= gap and arr[j - gap] > temp:
                arr[j] = arr[j - gap]
                j -= gap
            arr[j] = temp
        gap //= 2
    return arr
# Генеруємо великий список транзакцій (100 000)
transactions = [random.randint(1000, 10000) for _ in range(100000)]
# Тестуємо та виводимо результати
for algorithm in [quick_sort, heap_sort, shell_sort]:
    start_time = time.time()
    sorted_transactions = algorithm(transactions.copy())
    elapsed_time = time.time() - start_time
    print(f"{algorithm.__name__} зайняв {elapsed_time:.6f} секунд")
```

У цьому коді генерується великий список транзакцій (випадкові цілі числа) та порівнюється швидкодія кожного алгоритму для сортування цих транзакцій. Потім виводиться час виконання кожного алгоритму [3].

Внаслідок порівняльного аналізу ефективності алгоритмів сортування для великого списку транзакцій у банківській системі отримані такі результати:

Quick sort зайняв 0.127742 секунд;

Heap sort зайняв 0.042595 секунд;

Shell sort зайняв 0.398416 секунд.

У межах дослідження було проведено порівняльний аналіз трьох алгоритмів сортування – Quick sort, Heap sort та Shell sort – у контексті їх ефективності для великих обсягів даних, як-от транзакції в банківській системі. Heap sort виявився найшвидшим серед трьох алгоритмів, зайнявши найменше часу на сортування великого списку транзакцій. Quick sort також продемонстрував гарні результати, ефективно працюючи з великими обсягами даних. У той час як Shell sort показав

найгірші результати серед трьох алгоритмів для великих списків транзакцій. З цих результатів можна зробити висновок, що для оптимальної швидкодії сортування великих обсягів даних у банківській системі рекомендується використовувати Heap sort.

Список використаних джерел

1. Introduction to Algorithms / T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein. MIT Press, 2009.
2. Sedgewick R., Wayne K. Algorithms, Addison-Wesley Professional, 2011.
3. Weiss M. A. Data Structures and Algorithm Analysis in Java, Pearson, 2011.

УДК 004.94

Войтенко М. О., *здобувач 1 курсу ОС «Магістр» спеціальності 122 Комп'ютерні науки,*
Бабаков Р. М., *д-р техн. наук, доцент, професор кафедри інформаційних технологій*

ПРЕДСТАВЛЕННЯ РОЗВ'ЯЗКУ ЗАДАЧІ АЛГЕБРАЇЧНОГО СИНТЕЗУ МІКРОПРОГРАМНОГО АВТОМАТА У ВИГЛЯДІ ГРАФА

Донецький національний університет імені Василя Стуса, м. Вінниця

Сучасні технології виробництва цифрових систем досягли рівня, за якого проєктування обчислювальних пристроїв здійснюється в автоматичному або автоматизованому (напівавтоматичному) режимі на основі певних вхідних даних. Автоматизація процесу розробки цифрових систем та їх складників дає змогу не тільки зменшити час виготовлення пристроїв, а й провести оптимізацію характеристик пристроїв, як-от вартість, енергоспоживання, надійність тощо [1].

Одним із центральних складників сучасних цифрових систем є пристрій керування, який координує роботу усіх блоків системи і припускає різні пособи структурної організації [2]. Одним із таких способів є мікропрограмний автомат з операційним автоматом переходів (МПА з ОАП), у якому перетворення кодів станів під час автоматних переходів здійснюється за допомогою певної множини арифметико-логічних операцій [1].

Одним із етапів синтезу МПА з ОАП є так званий алгебраїчний синтез автомата [3]. В процесі алгебраїчного синтезу відбуваються такі дії:

1. Станам автомата ставляться у відповідність унікальні коди із певної множини кодів станів. Коди станів можуть розглядатись як алфавіти різних алгебр та інтерпретуватись як цілі числа, бітові вектори або в інший спосіб. Оскільки схемна реалізація кодів станів потребує їх представлення у двійковій формі, інтерпретація у вигляді бітових векторів є обов'язковою.

2. Автоматним переходам ставляться у відповідність певні операції переходів (ОП) із певної множини ОП. Використання однієї ОП для реалізації декількох

автоматних переходів є припустимим і сприяє зменшенню загальної кількості використовуваних ОП та відповідно зменшенню складності схеми автомата.

Спосіб виконання цих пунктів називають задачею алгебраїчного синтезу. Сьогодні розв'язання цієї задачі не є чітко формалізованим і не дає змогу автоматизувати процес алгебраїчного синтезу МПА з ОАП. Це ускладнює практичне застосування МПА з ОАП під час проєктування пристроїв керування сучасних цифрових систем. Отже, актуальною науково-практичною проблемою є розробка алгоритмів і методів розв'язання цієї задачі, а також автоматизація процесу алгебраїчного синтезу МПА з ОАП шляхом програмної реалізації відповідних алгоритмів.

МПА з ОАП може бути заданий у різний спосіб: за допомогою граф-схем алгоритмів, у вигляді операційної таблиці переходів, файлом у форматі «kiss2» тощо [1, 3]. Однак ці способи містять інформацію, зайву для опису функції переходів автомата (зокрема мікрооперації, що формуються автоматом). Для того, щоб представити у наочній формі лише функцію переходів автомата, пропонується використати форму орієнтованого графу. В такому графі кожна вершина відповідатиме окремому стану автомата, кожна дуга – автоматному переходу.

В якості прикладу задамо функцію переходів МПА з ОАП графом, наведеним на рис. 1.

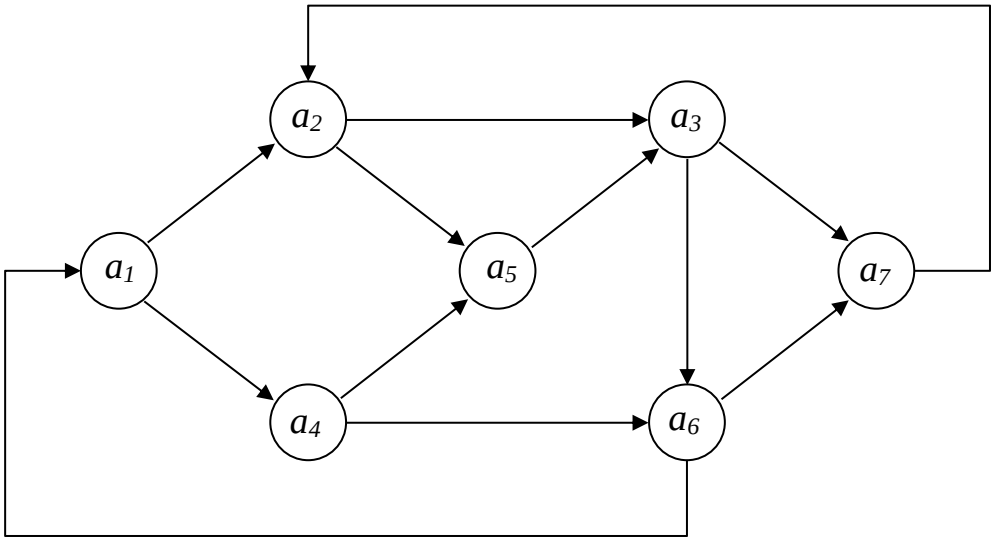


Рис. 1. Завдання функції переходів автомата орієнтованим графом

Аналіз графу показує, що автомат містить 7 станів a_1–a_7, для кодування яких достатньо трьох двійкових розрядів [2]. Нехай для МПА з ОАП задані три операції переходів, аргументом яких є код поточного стану автомата. Перша операція виконує додавання до коду поточного стану числа 1, друга – додавання числа 2, третя – додавання числа 4. Будемо позначати ці операції як «+1», «+2» та «+3» відповідно.

Оскільки ці операції передбачають подальшу схемну реалізацію, будемо вважати, що кожна з операцій виконується в межах трьох двійкових розрядів. Це означає, що аргументи операції (код поточного стану) і її результат (код стану переходу) завжди лежать у діапазоні [0; 7], тобто в діапазоні трирозрядних двійкових чисел. Наприклад, під час додавання до числа 7 одиниці ми отримуємо 0 та ін. Це називається виконанням операцій «за модулем 8».

Виконаємо розглянуті вище дії, передбачені процесом алгебраїчного синтезу МПА з ОАП. Зіставимо станам a_1–a_7 числові коди із діапазону [0; 7], а автоматним переходам – операції «+1», «+2», «+3». Спосіб, у який це зроблено, у цій роботі не розглядається. Результат, який є розв'язком задачі алгебраїчного синтезу МПА з ОАП, наведений на рис. 2.

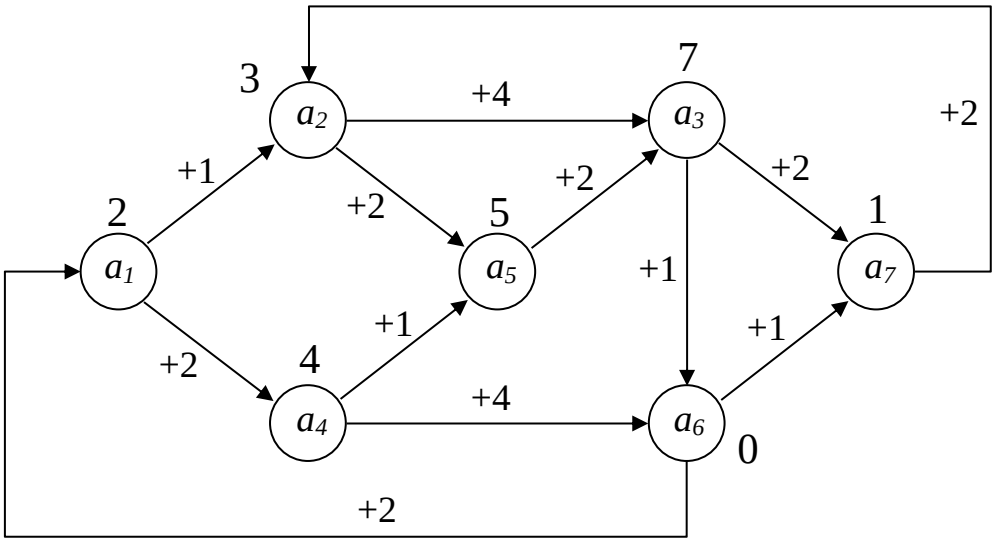


Рис. 2. Результат алгебраїчного синтезу

Аналіз рис. 2 дає змогу переконатись, що при обраних кодах станів усі автоматні переходи можуть бути реалізовані за допомогою трьох зазначених операцій. Наприклад, перехід зі стану a_3 з кодом 7 у стан a_7 з кодом 1 виконується за допомогою операції «+2». Дійсно: (7 + 2) mod 8 = 9 mod 8 = 1.

Розглянуте представлення розв'язку задачі алгебраїчного синтезу МПА з ОАП у вигляді графу дає змогу відобразити лише ту інформацію, яка стосується функції переходів автомата. Графове представлення надає потенційну можливість застосування різноманітних відомих методів із теорії графів для створення методів алгебраїчного синтезу МПА з ОАП, що в підсумку сприятиме вирішенню проблеми автоматизації синтезу даного класу цифрових пристроїв керування.

Список використаних джерел

1. Бабаков Р. М. Структури і методи синтезу мікропрограмних автоматів з операційним перетворенням кодів станів: дис. ... д-ра техн. наук: 05.13.05. Харків, 2020. 399 с.

2. Logic synthesis for FPGA-based finite state machines: monograph / A. A. Barkalov, L. A. Titarenko, R. M. Babakov, A. V. Baiev. Vinnitsya: DonNU Vasyl' Stus. 2016. 195 p.

3. Barkalov A. A., Titarenko L. A., Babakov R. M. Synthesis of VHDL Model of a Finite State Machine with Datapath of Transitions. *Радіоелектроніка. Інформатика. Управління*. 2023. Vol. 4. C. 135–147.

УДК 004.1

Просянніков А. В., *здобувач 4 курсу спеціальності 122 Комп'ютерні науки,*
Антонов Ю. С., *канд. фіз.-мат. наук, доцент, доцент кафедри інформаційних технологій*

РОЗРОБКА ПРОГРАМИ ДЛЯ ІНДИВІДУАЛЬНОЇ КОНФІГУРАЦІЇ АВТОМОБІЛЯ З ВИКОРИСТАННЯМ ПЛАТФОРМИ .NET ТА WINDOWS FORMS

Донецький національний університет імені Василя Стуса, м. Вінниця

У сучасному світі існує велика кількість виробників авто [1]. У кожного з них є власні інструменти для індивідуальної конфігурації авто для клієнта [2–4]. Вони відрізняються інтерфейсом і дизайном, функціональністю, доступними опціями, наявністю або відсутністю інтерактивності та технологічними можливостями. Не завжди зручно і інтуїтивно зрозуміло як користуватись кожним із них, тому тема роботи є актуальною.

Метою роботи є розробка програми для індивідуальної конфігурації автомобіля на основі платформи .NET з використанням мови програмування C# та Windows Forms [**Помилка! Джерело посилання не знайдено.**–**Помилка! Джерело посилання не знайдено.**].

Створення десктоп-додатків та пов'язані з ними теми підкреслюють важливість використання потужних інструментів та технологій, як-от Windows Forms, платформи .NET та мова програмування C#, для створення якісних десктоп-додатків, які задовольняють потреби користувачів. У роботі Еріка Брауна «Windows Forms Programming With C#» описана розробка додатків на Windows Forms, вимоги до них та приклади таких додатків [5].

Внаслідок дослідження був розроблений десктоп-додаток для індивідуальної конфігурації автомобіля. У процесі розробки було реалізовано інтерфейс користувача, база даних та робота з базою на основі фреймворку Entity Framework. Використовувався шаблон програмування MVP.

Робота з Entity Framework детально описана у роботі Джона Сміта «Entity Framework Core in Action» [6]. У першій частині він детально описав налаштування фреймворка, описав роботу з LINQ для запитів до даних, роботу з міграціями. Вона адресована як початківцям, так і досвідченим розробникам, які хочуть поглибити свої знання про EF.

Використання шаблонів програмування – дуже важливий аспект у програмуванні. Шаблони програмування описує робота Мартіна Фаулера «Patterns of Enterprise Application Architecture», вона вважається одною з основних книг із шаблонів для програмістів [7].

Windows Forms забезпечує можливість створення інтуїтивно зрозумілого інтерфейсу, що сприяє зручності взаємодії з програмою. Завдяки гнучкості Windows Forms можна реалізувати різноманітні елементи керування, від тексто-

вих полів до випадних списків, що дає змогу користувачеві легко налаштувати параметри автомобіля за власними потребами.

Entity Framework дає змогу зручно взаємодіяти з базою даних MS SQL Server, що використовується для зберігання інформації про автомобілі та їх конфігурації. Завдяки Entity Framework можна легко виконувати операції читання, запису та оновлення даних, що забезпечує швидкий та ефективний доступ до необхідної інформації.

Використання шаблону MVP допомагає розділити програму на модель (Model), вигляд (View) та пред'явник (Presenter), що сприяє розділенню обов'язків і полегшує розробку та підтримку коду. Модель відповідає за роботу з даними, вигляд відображає інтерфейс користувача, а пред'явник виконує логіку взаємодії між моделлю та виглядом. Це дає змогу зберігати код чистим, організованим та легко змінювати і розширювати.

Отже, використання платформи .NET, Windows Forms, Entity Framework, MS SQL Server та шаблону MVP дає змогу створити програму для індивідуальної конфігурації автомобіля, яка буде забезпечувати зручність, швидкість, безпеку та надійність у роботі з користувачем і його даними.

Список використаних джерел

1. Largest. Найбільші автовиробники світу: вебсайт. URL: https://largesthq.com/najbilshi-avtovirobniki-svitu/ (дата звернення: 09.05.2024).
2. Конфігуратор Porsche: вебсайт. URL: https://porsche.ua/modelstart/all (дата звернення: 09.05.2024).
3. Конфігуратор BMW: вебсайт. URL: https://www.bmw.ua/uk/configurator.html (дата звернення: 09.05.2024).
4. Конфігуратор Mercedes-Benz: вебсайт. URL: https://www.mercedes-benz.co.uk/passengercars/configurator.html?group=all&subgroup=see-all&view=BODYTYPE (дата звернення: 09.05.2024).
5. Браун Е. Windows Forms Programming With C#. 2002. 695 с. URL: https://www.academia.edu/36682764/Windows_Forms_Programming_With_C_ (дата звернення: 09.05.2024).
6. Сміт Д. Entity Framework Core in Action, 2018. 520 с.
7. Фаулер М. Patterns of Enterprise Application Architecture, 2002. 560 с.

УДК 004.4

Капля Г. О., *здобувач 4 курсу спеціальності 122 Комп'ютерні науки,*
Бабаков Р. М., *д-р техн. наук, доцент, професор кафедри інформаційних технологій*

ВИКОРИСТАННЯ ТЕХНОЛОГІЇ BLUEPRINT У РОЗРОБЦІ ІГРОВИХ ДОДАТКІВ

Донецький національний університет імені Василя Стуса, м. Вінниця

Завдяки широкому доступу та прогресу в області комп'ютерних технологій індустрія відеоігор продовжує залучати все більше уваги. Вже у 2021 році світова

аудиторія геймерів сягнула позначки в 3,2 мільярди осіб [1]. Через різноманітність уподобань серед такої величезної аудиторії гравців індустрія постійно працює над розробкою великої кількості унікальних відеоігрових проєктів, щоб задовольнити запити кожного. Однак створення відеоігрового додатка є складним і тривалим процесом, який вимагає значних зусиль та часу, де ключову роль відіграють інноваційність та креативність для створення неповторних ігрових механік, захопливих світів та сюжетних ліній.

Технології візуального програмування, як-от технологія Blueprint в Unreal Engine, надають розробникам потужні інструменти для втілення їх ідей з меншими технічними бар'єрами та витратами часу. Ці інструменти допомагають як досвідченим розробникам, так і початківцям, які роблять свої перші кроки у відеоігровій індустрії.

Blueprint – це інтегрована в Unreal Engine система візуального програмування, яка дає змогу створювати ігрову логіку, сценарії та взаємодії без необхідності писати код на мовах програмування, як-от C++ [2]. Замість написання програмного коду розробники використовують інтуїтивний графічний інтерфейс, у якому можна не лише змінювати та додавати певні параметри класів, а й візуально зв'язувати різноманітні вузли для створення бажаної поведінки та функціоналу.

Технологія візуального програмування Blueprint забезпечує легкий вхід у світ розробки завдяки великій кількості навчальних ресурсів, простому графічному інтерфейсу та виключенні виникнення синтаксичних помилок під час розробки. Проте навіть із такими перевагами ця технологія все ж таки вимагає базових знань програмування та об'єктно-орієнтованого підходу для створення повноцінного функціонального контенту.

На відміну від традиційного програмування мовою C++, у Blueprint можна вносити зміни безпосередньо під час роботи над проєктом без потреби у компіляції коду, яка у проєктах Unreal Engine може тривати понад 15 хвилин. Це значно пришвидшує цикл розробки, дає змогу оперативно тестувати та коригувати ігрову логіку, сприяючи ітеративному підходу.

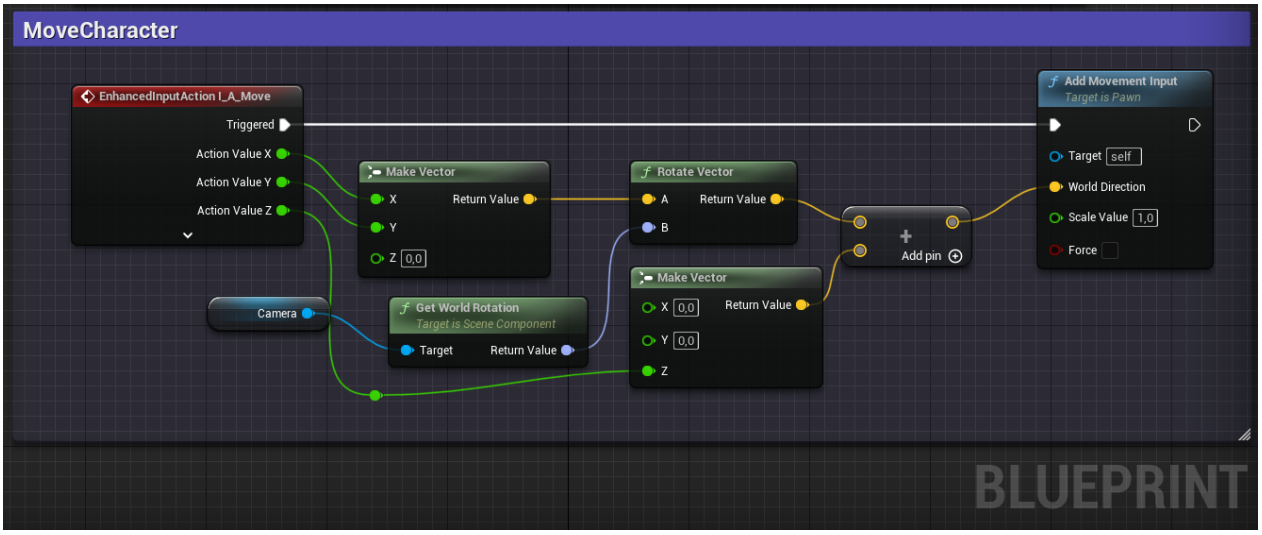


Рис. 1. Інтерфейс Blueprint

Однак, незважаючи на зазначені переваги, Blueprint має певні недоліки. Код, написаний на Blueprint, може мати суттєво нижчу продуктивність, порівняно з

рівноцінним кодом, написаним на C++. Різниця може сягати десятків та сотень разів, хоча розробники Blueprint постійно працюють над зменшенням цього розриву.

До того ж функціонал Blueprint є обмеженим, порівняно з можливостями мов програмування, як-от C++. Тому розробники мають можливість використовувати додаткові плагіни, розроблені спільнотою, або писати певний функціонал мовою C++, оскільки Unreal Engine дає змогу використовувати гібридну розробку із застосуванням обох технологій.

Враховуючи розглянуті переваги та недоліки, можна зробити висновок, що Blueprint, хоч і менш функціональний та менш швидкий, порівняно з C++, є відмінним інструментом для початківців та дизайнерів без глибоких технічних знань у програмуванні чи технології Unreal Engine. Досвідчені розробники використовують обидві технології: Blueprint – для швидкого написання та тестування ігрових механік, C++ – для написання ресурсномістких систем та забезпечення оптимальної продуктивності кінцевого продукту.

Список використаних джерел

1. Share of video gamers worldwide in 2022: вебсайт. URL: https://www.statista.com/statistics/297874/number-mobile-gamers-region/ (дата звернення: 09.05.2024)

2. Blueprint documentation: вебсайт. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/introduction-to-blueprints-visual-scripting-in-unreal-engine?application_version=5.3 (дата звернення: 09.05.2024)

УДК 004.056.5

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

ВИПАДКОВІ ВЕЛИЧИНИ ТА РОЗПОДІЛИ ЙМОВІРНОСТЕЙ У КІБЕРБЕЗПЕЦІ

Донецький національний університет імені Василя Стуса, м. Вінниця

Проблеми захисту інформації стають все більш актуальними на сучасному етапі розвитку суспільства, коли загрози для безпеки інформації та приватності постійно зростають. Застосування математичного апарату в інформаційній та кібернетичній безпеці є ключовими аспектами для створення надійних систем захисту даних та мереж. Розглянемо це на прикладі випадкових величин та розподілів ймовірностей.

Одним з основних застосувань випадкових величин у кібербезпеці є генерація криптографічних ключів. Випадковість ключів є важливою для забезпечення стійкості криптографічних алгоритмів шифрування та автентифікації. Застосування різних розподілів ймовірностей, як-от рівномірний, нормальний або експо-

ненціальний, може забезпечити відповідний рівень випадковості генерованих ключів [1].

Випадкові величини та розподіли ймовірностей використовуються для аналізу вразливостей та прогнозування ризиків у кібербезпеці [2]. Моделювання випадкових процесів може допомогти ідентифікувати потенційні загрози та виявляти слабкі місця в інформаційних системах і мережах. Використання випадкових величин дає змогу проводити статистичний аналіз даних щодо вразливості програмного забезпечення та мереж [3]. Це може допомогти в розумінні розподілу часу між виявленням вразливості та випуском виправлення, що в свою чергу допомагає в управлінні ризиками та розробці стратегій захисту.

Використання випадкових величин та розподілів ймовірностей у кібербезпеці є важливим для створення надійних систем захисту даних та мереж. Розглянемо код на C# (рис. 1), який використовує випадкові величини для створення мережі з 50 вузлів, де кожен вузол має певну ймовірність атаки. Симуляція атаки на мережу дає змогу обчислити кількість вразливих вузлів та відсоток вразливості, що допомагає аналізувати стійкість мережі до потенційних загроз.

```
using System;
using System.IO;
using System.Text;

class Program
{
   static void Main()
   {
      // Параметри симуляції
      int networkSize = 50; // Розмір мережі (кількість вузлів)
      double attackProbability = 0.1; // Ймовірність атаки на кожен вузол

      // Створення мережі
      bool[] network = new bool[networkSize]; // Масив для представлення стану кожного вузла (true - вразливий, false - невразливий)

      // Ініціалізація мережі: встановлення вразливості для певної частини вузлів
      Random rand = new Random();
      for (int i = 0; i < networkSize; i++)
      {
         network[i] = rand.NextDouble() < 0.5; // 50% вузлів вважається вразливими
      }

      // Симуляція атаки на мережу
      int vulnerableNodes = 0; // Лічильник вразливих вузлів
      for (int i = 0; i < networkSize; i++)
      {
         if (network[i] && rand.NextDouble() < attackProbability)
         {
            // Якщо вузол вразливий і атака відбувається
            Console.WriteLine($"Вузол {i} був атакований!");
            vulnerableNodes++;
         }
      }
```

```
        // Виведення результатів симуляції
        Console.WriteLine($"У мережі з {networkSize} вузлів {vulnerableNodes} вразливих.");
        double vulnerabilityRatio = (double)vulnerableNodes / networkSize;
        Console.WriteLine($"Відсоток вразливих вузлів: {vulnerabilityRatio:P}");

        // Збереження у файл з кодуванням UTF-8
        string filePath = "output.txt";
        using (StreamWriter writer = new StreamWriter(filePath, false, Encoding.UTF8))
        {
            writer.WriteLine($"У мережі з {networkSize} вузлів {vulnerableNodes} вразливих.");
            writer.WriteLine($"Відсоток вразливих вузлів: {vulnerabilityRatio:P}");
        }
    }
}
```

Рис. 1. Код дослідження вразливостей на C#

Після проведення симуляції програма обчислює кількість вразливих вузлів та відсоток вразливості мережі (рис. 2).

```
Вузол 32 був атакований!
Вузол 35 був атакований!
Вузол 40 був атакований!
У мережі з 50 вузлів 3 вразливих.
Відсоток вразливих вузлів: 6,00%
Press any key to continue . . .
```

Рис. 2. Результати роботи коду

Ці результати важливі для аналізу та управління ризиками, а також для вдосконалення системи захисту. Збереження результатів у файл дає змогу вести запис та дослідження вразливостей, що сприяє подальшому аналізу та поліпшенню системи.

Отже, використання випадкових величин та розподілів ймовірностей у кібербезпеці допомагає не лише оцінювати стійкість системи, але й ідентифікувати потенційні загрози та виявляти слабкі місця у мережі, що робить її більш надійною та захищеною.

Список використаних джерел

1. Гулак Г. М. Методологія захисту інформації. Аспекти кібербезпеки: навчальний посібник. Київ: Видавництво НА СБ України. 2020. 256 с.
2. Sánchez-García I. D., Mejía J., San Feliu T. Cybersecurity Risk Assessment: A Systematic Mapping Review, Proposal, and Validation. *Applied Sciences.* 2023. № 13(1). 29 p. DOI: 10.3390/app13010395 (дата звернення 18.05.2024).
3. Дудикевич В. Б. Основи інформаційної безпеки: навч. посібник. Вінниця: ВНТУ. 2018. 316 с.

УДК 004.1

Оврамець І. В., *здобувач 4 курсу спеціальності 122 Комп'ютерні науки,*
Антонов Ю. С., *канд. фіз.-мат. наук, доцент, доцент кафедри інформаційних технологій*

АРХІТЕКТУРНІ ОСОБЛИВОСТІ МЕСЕНДЖЕРА З БАГАТОРІВНЕВИМ ШИФРУВАННЯМ

Донецький національний університет імені Василя Стуса, м. Вінниця

У сучасному світі питання шифрування, захисту інформації та протидії кіберзагрозам є доволі актуальними [1–2], оскільки кожна людина постійно комунікує зі своїми колегами, друзями або близькими, використовуючи для цього месенджери або інші засоби зв'язку.

Під час розробки різноманітних додатків важливим складником є архітектура програмного забезпечення [3–4], особливо коли йдеться про великі проєкти зі значною кількістю кодів. Необхідно зробити функціонал достатньо простим для розширення та виправлення помилок. Можливість повторного використання коду в інших проєктах можна реалізувати за допомогою багатошарової архітектури.

Цю роботу присвячено створенню кросплатформного месенджера з багаторівневим шифруванням та особливостям його архітектурної реалізації.

Для коректної роботи месенджера виникає потреба уніфікувати методи шифрування, тобто зробити так, щоб за потреби додати новий тип шифрування в месенджер було достатньо реалізувати відповідний інтерфейс і додати певні налаштування у файл конфігурації. Вирішення цього завдання можна продемонструвати у вигляді схеми взаємодії класів між собою.

На рис. 1 наведено схему діаграми класів для реалізації криптографічних алгоритмів [5] у месенджері з багаторівневим шифруванням. Вона складається з декількох класів та інтерфейсів:

ICrypto – основний інтерфейс, який містить методи, що дають змогу здійснювати шифрування (Crypt) та дешифрування (Decrypt) повідомлень.

SymmetricCryptoAlg та AsymmetricCryptoAlg – абстрактні класи, що забезпечують базове функціонування для симетричних та асиметричних криптографічних алгоритмів відповідно.

PermutationCryptoAlg – конкретна реалізація алгоритму перестановки для реалізації шифрування повідомлень на основі класу SymmetricCryptoAlg.

ShanonFanoAlg – реалізація симетричного алгоритму шифрування, яка використовує реалізовані класи та методи, що безпосередньо не знаходяться в цьому класі, проте викликаються, обгортка реалізується на основі класу SymmetricCryptoAlg.

RSACryptoAlg – реалізація асиметричного алгоритму шифрування на основі абстрактного класу AsymmetricCryptoAlg.

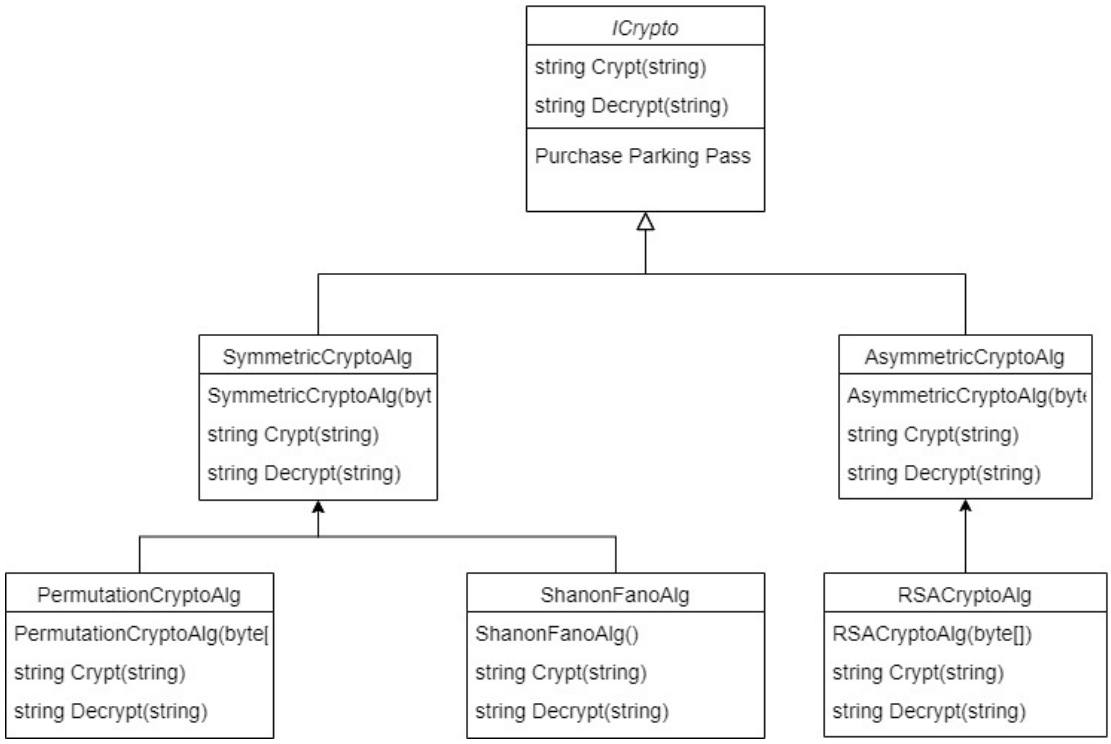


Рис. 1. Діаграма класів криптографічної компоненти

На рис. 2 наведено діаграму фабричних методів, що складається з таких класів:

ICryptoFactory – визначає інтерфейс для створення криптографічних об’єктів через два методи:

– ICrypto Create() – створює об’єкт без додаткових параметрів;

– ICrypto Create(string[] param) – створює об’єкт, приймаючи масив параметрів.

CryptoPermutationFactory – створює об’єкти, що реалізують криптографію з використанням перестановок.

ShanonFanoCryptoFactory – створює об’єкти, що реалізують криптографію з використанням методу Шеннона-Фано.

RSACryptoFactory – створює об’єкти, що реалізують RSA-криптографію.

Отже, ця діаграма демонструє застосування шаблону Factory Method для створення різних видів криптографічних об’єктів через абстрактну фабрику та її конкретні реалізації.

Ця архітектура забезпечує кілька важливих переваг:

Розширюваність – завдяки використанню інтерфейсу ICrypto й абстрактних класів SymmetricCryptoAlg та AsymmetricCryptoAlg додавання нового алгоритму шифрування стає простим завданням. Потрібно лише створити новий клас, який реалізує відповідний інтерфейс, і програма автоматично зможе використовувати цей новий алгоритм.

Гнучкість конфігурації – параметри алгоритмів та послідовність їх застосування може бути індівідуальними для кожного абонента.

Надійність шифрування – для шифрування повідомлення, може використовуватись одночасно декілька різних алгоритмів, що буде підвищувати захищеність повідомлення загалом.

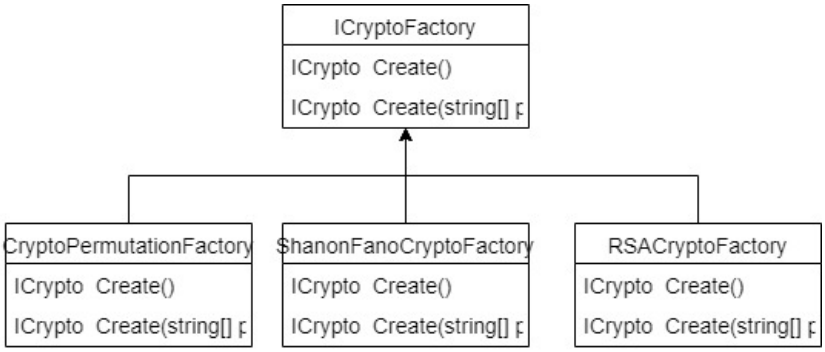


Рис. 2. Діаграма фабричних класів

Повторне використання коду – оскільки алгоритми шифрування реалізовані як окремі класи, їх можна легко повторно використовувати в інших проєктах.

Завдяки цій архітектурі розробники можуть легко додавати, видаляти або змінювати алгоритми шифрування без необхідності внесення змін у весь код програми. Це робить систему гнучкою, масштабованою та простою у підтримці, особливо для великих проєктів з багатьма залежностями.

Список використаних джерел

1. Демашкевич А. В., Антонов Ю. С. Розробка програмного забезпечення для приховування повідомлення в цифрових зображеннях за допомогою методу LSB. Матеріали IV Всеукраїнської науково-практичної конференції *«Комп'ютерні технології обробки даних»*. Вінниця, ДонНУ імені Василя Стуса. 2023. С. 242–245.

2. Антонов Ю. С., Римар П. В., Антонова О. Г. Проблема DoS/DDoS атак навчальних ресурсів здобувачами. *Сучасний захист інформації*. 2019. № 4(40). С. 52–62.

3. Tune N., Perrin J. Architecture Modernization. 2024. 488 с.

4. Design Patterns. URL: https://refactoring.guru/design-patterns

5. Wong D. Real-World Cryptography. 2021. 400 с.

УДК 004.1

Явгусішин Б. А., *здобувач 4 курсу спеціальності 122 Комп'ютерні науки,*
Антонов Ю. С., *канд. фіз.-мат. наук, доцент, доцент кафедри інформаційних технологій*

РЕАЛІЗАЦІЯ КАЗУАЛЬНОЇ ГРИ ДЛЯ МОБІЛЬНИХ ПРИСТРОЇВ З ВИКОРИСТАННЯМ ПЛАТФОРМИ .NET

Донецький національний університет імені Василя Стуса, м. Вінниця

Сучасна ігрова індустрія залучає мільйони гравців з усього світу та отримує великі прибутки. Попри глобальну рецесію та зменшення ігрового ринку через введення обмежень у Китаї, ринок був у передбаченому падінні після двох років зростання через пандемію. Наразі ринок стабілізується, що є гарною можливістю для просунення власних ігрових додатків [1–2]. Враховуючи те, що на світовому ринку ігрової індустрії найуспішнішим та найприбутковішим сектором є ігри для мобільних пристроїв, тема роботи є актуальною.

Метою роботи є розробка казуальної гри "Code Frost: Cybernetic Battle" на основі платформи .NET з використанням Unity та мови програмування C#.

Останні дослідження вказують на популярність ігрової індустрії та розробки ігрових додатків. Наприклад, у роботі «Розробка мобільної комп'ютерної гри "Морський бій" під платформу Android за допомогою Java» досліджується процес створення гри з використанням мови програмування Java та висвітлюються проблеми, що можуть виникнути під час розробки гри [3]. Інша робота, «Створення інтелектуальної мережевої гри ШАШКИ», досліджує аспекти розробки мережевих ігор, що включають роботу між web-сервером та desktop-додатками користувачів [4].

Внаслідок дослідження була розроблена казуальна гра «Code Frost: Cybernetic Battle». У процесі розробки гри «Code Frost: Cybernetic Battle» було реалізовано ігрові об'єкти з їх логікою взаємодії, ігрові механіки і можливості та інтерфейс користувача, за допомогою яких гравець може взаємодіяти з грою.

З основних ігрових об'єктів, що були реалізовані, варто виділити:

- ігровий персонаж, яким керує користувач;
- різні види ворогів, що з'являються навколо гравця;
- зброя, якою атакують як вороги, так і ігровий персонаж.

Кожен з ігрових об'єктів має власну логіку взаємодії з іншими. Наприклад, гравець може замінити зброю, яку використовує його персонаж, підібравши інший тип зброї з ігрової площини. Зброя знаходиться в рандомізованих місцях цієї площини.

Щодо реалізованих ігрових механік та можливостей варто зазначити такі: переміщення ігрового персонажа за допомогою екранного стіка, що з'являється при дотику до нижньої половини екрана; автоматична атака персонажа по ворогах наявною зброєю; зона хабу, де можна витратити внутрішньоігрову валюту для покращення показників зброї; зменшення здоров'я ігрового персонажа з плином часу для обмеження часу гри в одній сесії; нескінченна ігрова площина, по якій може переміщуватися гравець.

Також важливою частиною реалізації гри є використання патернів проєктування. Головною перевагою є забезпечення легкої підтримки та супроводу створеного програмного продукту. Також їх використання пришвидшує тестування та виявлення багів. Основними використаними патернами були:

- dependency injection – у програмі було використано фреймворк Zenject, який реалізовує цей патерн;
- state – було реалізовано state machine, що контролює основний процес гри. Наприклад, запускає появу ворогів на ігровому полі;
- Model-View-Controller (MVC) – використовується у представленні інтерфейсів та вікон користувачу;
- Observer – поведінковий патерн, який дає можливість об'єктам стежити й реагувати на події, які відбуваються в інших об'єктах. Реалізовано в деяких класах для кращої незв'язності коду;
- Object pool – використовується для появи нових ворогів на ігровій площині.

Було розглянуто основні реалізовані ігрові об’єкти, ігрові механіки і можливості та основні використані патерни під час реалізації казуальної гри «Code Frost: Cybernetic Battle» на основі платформи .NET з використанням Unity та мови програмування C#, та реалізовано програму (рис. 1).

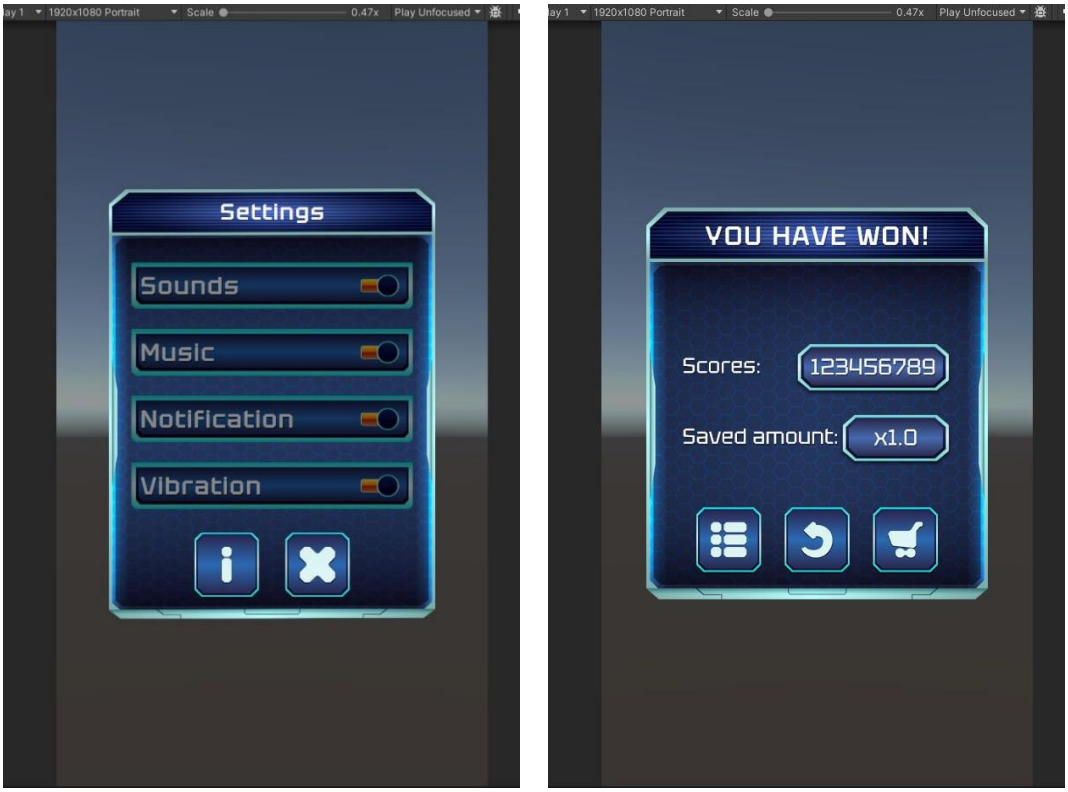


Рис. 1. Приклад інтерфейсу програми

Список використаних джерел

1. Knezovic A. 200+ Mobile Games Statistics: Market & Revenue Report. 2024. URL: https://www.blog.udonis.co/mobile-marketing/mobile-games/mobile-gaming-statistics (дата звернення: 15.05.2024).

2. Hawkins A. Chinese gaming sector in turmoil as regulators announce new proposals. 2023. URL: https://www.theguardian.com/world/2023/dec/27/chinese-gaming-shares-fall-as-regulators-announce-new-proposals (дата звернення: 15.05.2024).

3. Гнатюк М. А., Антонов Ю. С. Розробка мобільної комп’ютерної гри «Морський бій» під платформу Android за допомогою Java. 2024. URL: https://jpvs.donnu.edu.ua/article/view/7242 (дата звернення: 17.05.2024).

4. Сірков О. О., Антонов Ю. С. Створення інтелектуальної мережевої гри ШАШКИ. 2024. URL: https://jait.donnu.edu.ua/article/view/8942 (дата звернення: 17.05.2024).

5. ModestTree. Zenject: вебсайт. https://github.com/modesttree/Zenject (дата звернення: 16.05.2024).

УДК 004.422.5

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

ОСОБЛИВОСТІ ВИКОРИСТАННЯ СТРУКТУРИ ДАНИХ «ЧЕРГА» У ПРОГРАМУВАННІ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЯХ

Донецький національний університет імені Василя Стуса, м. Вінниця

У світі комп'ютерних технологій черга – це важлива концепція, яка допомагає у впорядкуванні та управлінні набором елементів. Вона використовується для зберігання та обробки даних у такий спосіб, що перший елемент, який додається до черги, також є першим, що видаляється з неї. Цей принцип, відомий як «першим прийшов – першим пішов (FIFO), дає змогу ефективно використовувати чергу в різних областях інформатики, включно з операційними системами, алгоритмами обробки даних та ін. [1].

Черги відіграють важливу роль у багатьох аспектах програмування, від управління завданнями до обробки подій. Вони є важливим інструментом для створення ефективних та надійних програм, які працюють із послідовністю дій або подій. Черги можуть мати різні реалізації, як-от черги на базі масивів або черги на базі зв'язних списків, і вони знаходять своє застосування в різних областях інформатики, від розробки програмного забезпечення до оптимізації роботи систем [2].

Черги також можна знайти в реальних сценаріях за межами цифрового світу. Уявіть, наприклад, чергу людей, які чекають на посадку в автобус. Перша людина, яка прибуває на зупинку, першою сідає в автобус, коли він прибуває. Аналогічна ситуація і з квитковою системою: хто перший у черзі, той обслуговується першим [3].

Розуміння того, як працюють черги та їх застосування, має вирішальне значення в інформатиці та розробці програмного забезпечення. Це дає змогу програмістам розробляти ефективні алгоритми та системи, які можуть виконувати завдання структуровано та організовано. Отже, наступного разу, коли ви стоятимете в черзі або працюватимете з комп'ютерною програмою, подумайте про концепцію черг і про те, як вони допомагають операціям працювати безперебійно [4]. А тепер давайте розглянемо на практиці чергу.

Програма нижче моделює роботу банківської черги, де клієнти заходять у чергу та обслуговуються у порядку їхнього приходу. Використовуючи мову програмування C#, код демонструє структуру черги, реалізованої за допомогою стандартного класу «Queue», а також механізм додавання клієнтів до черги та їх обслуговування.

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading;

0 references
class Program
{
    0 references
    static void Main()
    {
        // Встановлюємо кодування UTF-8 для консолі
        Console.OutputEncoding = Encoding.UTF8;

        // Створюємо чергу для банку
        var bankQueue = new BankQueue();

        // Додаємо клієнтів до черги
        bankQueue.Enqueue("Клієнт 1");
        bankQueue.Enqueue("Клієнт 2");
        bankQueue.Enqueue("Клієнт 3");

        // Обслуговуємо клієнтів та виводимо чергу
        Thread.Sleep(2000); // Симулюємо час обслуговування
        while (bankQueue.Count > 0)
        {
            Console.WriteLine($"{bankQueue.Dequeue()} був обслужений та вийшов з черги.");
            Thread.Sleep(1000); // Симулюємо час обслуговування
        }
    }
}
```

Рис. 1. Код роботи банківської черги

```
}

1 reference
class BankQueue
{
    private Queue<string> queue = new Queue<string>();

    1 reference
    public int Count => queue.Count;

    3 references
    public void Enqueue(string customer)
    {
        queue.Enqueue(customer);
        Console.WriteLine($"{customer} вступив до черги.");
    }

    1 reference
    public string Dequeue()
    {
        if (queue.Count > 0)
            return queue.Dequeue();
        else
        {
            Console.WriteLine("Черга порожня.");
            return null;
        }
    }
}
```

Рис. 2. Продовження коду

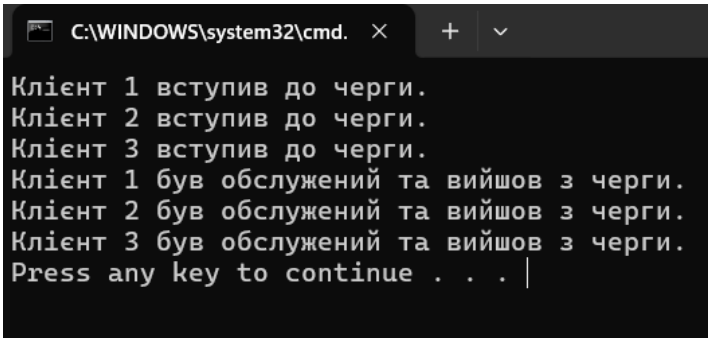


Рис. 3. Результат коду

Цей код пропонує просту реалізацію черги у мові програмування C#, яка дає змогу ефективно додавати клієнтів до черги, обслуговувати їх у порядку черги та виводити повідомлення про їх обслуговування. Черги відіграють важливу роль у програмуванні та інформаційних технологіях, забезпечуючи правильне упорядкування та обробку даних у процесі виконання програм. Вони знаходять застосування у широкому спектрі задач, від управління завданнями до оптимізації роботи систем, завдяки принципу «першим прийшов – першим обслугований», який є ключовим для багатьох алгоритмів та програмних рішень.

Список використаних джерел

1. Черга. Способи реалізації черги. Представлення черги у вигляді динамічного масиву. *BestProg:* вебсайт. 2019. URL: https://www.bestprog.net/uk/2019/09/26/c-queue-general-concepts-ways-to-implement-the-queue-implementing-a-queue-as-a-dynamic-array-ua/ (дата звернення: 30.04.2024).

2. Жмурко О. О. Урок: Роль інформаційних технологій у житті сучасної людини. *Vseosvita:* вебсайт. 2024. URL: https://vseosvita.ua/lesson/rol-informatsiinykh-tekhnolohii-u-zhytti-suchasnoi-liudyny-229384.html (дата звернення 30.04.2024).

3. Основи інформатики та технологій програмування: навчальний посібник / М. Є. Рогоза, С. К. Рамазанов, А. В. Велігура, С. М. Танченко. Луганськ: Вид-во СНУ ім. В. Даля, 2012. 568 с.

4. Програмна реалізація та дослідження алгоритмів паралельного швидкого сортування / В. О. Денисюк, Н. А. Потапова, О. В. Зелінська, М. Б. Тарасюк. *Вісник Хмельницького національного університету. Технічні науки.* 2023. № 4. С. 95–105. URL: http://journals.khnu.km.ua/vestnik/?page_id=41

УДК 004.6

Сидорчук Р. В., *здобувач*
1 курсу ОС «Магістр»
спеціальності 122 Комп'ютерні науки,
Потапова Н. А., *канд. екон. наук,*
доцент, доцент кафедри
інформаційних технологій

СУЧАСНІ ТЕХНОЛОГІЇ РОЗРОБКИ МОБІЛЬНИХ ЗАСТОСУНКІВ

Донецький національний університет імені Василя Стуса, м. Вінниця

Програмні застосунки – це програми, розроблені для виконання конкретних завдань або вирішення певних проблем користувачів. Вони можуть працювати на різних платформах, зокрема на персональних комп'ютерах, серверах та мобільних пристроях. Мобільні програмні застосунки (або мобільні додатки) – це програми, спеціально створені для смартфонів і планшетів. Ці додатки оптимізовані для роботи на мобільних операційних системах, як-от iOS та Android, забезпечуючи користувачам доступ до різноманітних функцій та сервісів безпосередньо зі своїх портативних пристроїв.

Створення програмних застосунків для мобільних платформ має велике значення в сучасному світі. З поширенням мобільних технологій і збільшенням кількості користувачів смартфонів мобільні додатки стали невід'ємною частиною повсякденного життя. Вони допомагають виконувати різні завдання – від комунікацій і розваг до бізнесу та навчання. Розробка мобільних застосунків дає змогу забезпечити максимальну продуктивність, стабільність і безпеку, а також використовувати повний спектр можливостей пристроїв, що робить їх ключовим елементом у досягненні ефективності та задоволення потреб користувачів. Розробка мобільних додатків є надзвичайно актуальною в сучасному цифровому світі. Зростаюча кількість користувачів смартфонів і планшетів стимулює попит на високоякісні, функціональні та зручні мобільні додатки. Бізнеси, які інвестують у мобільні рішення, отримують конкурентні переваги, підвищують лояльність клієнтів та оптимізують внутрішні процеси.

Сучасні технології значно спрощують процес створення мобільних додатків. Ось перелік деяких із таких технологій:

- React Native дає змогу розробляти нативні додатки для iOS та Android, використовуючи одну кодову базу на JavaScript.
- Progressive Web Apps (PWA) поєднують можливості вебсайтів і мобільних додатків, забезпечуючи швидкий доступ і функціональність без необхідності встановлення.

Flutter від Google активно використовуються для створення високопродуктивних і кросплатформних рішень, що робить розробку мобільних додатків доступною та ефективною.

- Cordova часто використовують разом з Angular, що дає змогу створювати гібридні мобільні додатки, які можуть працювати на різних платформах із використанням вебтехнологій.

React Native – це фреймворк для розробки мобільних додатків, створений компанією Facebook. Він дає змогу використовувати JavaScript і React для створення нативних додатків для iOS та Android з однією кодовою базою. React Native забезпечує швидку розробку та можливість повторного використання коду між платформами [1].

Переваги React Native: висока продуктивність, можливість повторного використання коду, активна спільнота та підтримка.

Недоліки React Native: обмежений доступ до деяких нативних API, можливі проблеми з продуктивністю у складних додатках, залежність від сторонніх бібліотек.

Progressive Web Apps (PWA) – це вебдодатки, які використовують сучасні вебтехнології для надання досвіду, схожої на нативні мобільні додатки. Вони можуть працювати офлайн, надсилати push-повідомлення і встановлюватися на домашній екран, забезпечуючи швидкий доступ до функцій додатка без необхідності встановлення через магазини додатків [2].

Переваги Progressive Web Apps: відсутність необхідності у встановленні, швидкий доступ, менші витрати на розробку та підтримку, кросплатформність.

Недоліки Progressive Web Apps: обмежені можливості доступу до нативних функцій пристрою, залежність від браузера, менша продуктивність, порівняно з нативними додатками.

Flutter – це відкритий фреймворк для розробки мобільних додатків, створений компанією Google. Він дає змогу розробляти нативні додатки для iOS, Android, веб- і десктопних платформ із єдиною кодовою базою на мові Dart. Flutter використовує власний графічний рушій для рендерингу, що забезпечує високу продуктивність і гнучкість у розробці інтерфейсів користувача [3].

Переваги Flutter: висока продуктивність завдяки нативному рендерингу, можливість створення кросплатформних додатків із єдиною кодовою базою, багата бібліотека віджетів, активна підтримка і велика спільнота, швидкий цикл розробки та тестування завдяки функції «hot reload», яка дає змогу моментально бачити внесені зміни).

Недоліки Flutter: великий розмір кінцевих додатків, обмежена кількість готових бібліотек і пакетів, порівняно з іншими фреймворками, необхідність вивчення мови Dart, потенційні проблеми з сумісністю під час оновлення платформ.

Apache Cordova – це платформа для створення гібридних мобільних додатків, яка дає змогу використовувати вебтехнології, як-от HTML, CSS і JavaScript. Вона обгортає вебдодаток у нативний контейнер, надаючи доступ до функцій пристрою через JavaScript API. Cordova часто використовується в поєднанні з фреймворками, як-от Angular, для створення багатофункціональних і кросплатформних мобільних додатків [4].

Переваги Cordova: легкість у використанні для веброзробників, кросплатформна розробка з однією кодовою базою, доступ до нативних функцій через плагіни, підтримка багатьох платформ, швидка розробка і низькі витрати на підтримку.

Недоліки Cordova: менша продуктивність, порівняно з нативними додатками, обмежена можливість оптимізації, залежність від плагінів для доступу до на-

тивних функцій, можливі проблеми з сумісністю між різними платформами, складність у підтримці та оновленні великих додатків.

Отже, зараз існує дуже багато технологій для розробки мобільних застосунків. Вибір технології залежить від конкретних потреб проєкту. Нативна розробка для кожної платформи може забезпечити найвищий рівень продуктивності і доступу до всіх можливостей пристрою, але вона також потребує більших зусиль у розробці та підтримці.

Список використаних джерел

1. Meta Platforms. React Native Architecture Overview: вебсайт. URL: https://reactnative.dev/architecture/overview

2. LePage Pete, Richard Sam. What are Progressive Web Apps? 2020. URL: https://web.dev/articles/what-are-pwas

3. Dembny Michal. What is Flutter and How Can It Benefit Your Business? 2024. URL: https://www.thedroidsonroids.com/blog/what-is-flutter-app-development

4. Petkovski Filip. Developing Mobile Applications with Cordova. 2014. URL: https://www.toptal.com/app/developing-mobile-applications-with-apache-cordova

УДК 004.738.5

Лаптєва М. А., *здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,*
Зелінська О. В., *канд. техн. наук, доцент, доцент кафедри інформаційних технологій*

СУЧАСНІ ТЕНДЕНЦІЇ BACKEND МОБІЛЬНИХ ДОДАТКІВ

Донецький національний університет імені Василя Стуса, м. Вінниця

З розвитком мобільних технологій та зростанням попиту на мобільні додатки виникла потреба у створенні ефективних та надійних бекенд-рішень. Бекенд мобільних додатків забезпечує обробку даних, управління користувачами, інтеграцію з різними сервісами та гарантує безпеку. У цій статті розглядаються основні сучасні тенденції в цій галузі.

Однією з найпопулярніших тенденцій у розробці бекенду є серверлесс-архітектура. Ця модель дає змогу розробникам зосередитися на написанні коду, не турбуючись про управління серверами. Використання сервісів, як-от AWS Lambda, Google Cloud Functions та Azure Functions, значно спрощує процес масштабування та знижує витрати на інфраструктуру. Це також зменшує час на розгортання нових функцій, оскільки розробники можуть швидко реалізовувати та тестувати нові ідеї [1].

Мікросервісна архітектура стає дедалі популярнішою завдяки своїй гнучкості та масштабованості. Вона передбачає розділення додатка на незалежні модулі, кожен з яких виконує окрему функцію і може бути розгорнутий та масштабований окремо. Це дає змогу командам працювати автономніше та швидше реагува-

ти на зміни у вимогах. До того ж це сприяє покращенню надійності додатка, оскільки збій одного мікросервісу не призводить до відмови всього додатка.

GraphQL, розроблений Facebook, є сучасною альтернативою REST API. Він дає змогу клієнтам запитувати лише ті дані, які їм потрібні, що зменшує обсяг переданої інформації та покращує продуктивність. GraphQL також забезпечує більш гнучкий та ефективний підхід до обробки запитів, що особливо важливо для мобільних додатків з обмеженою пропускною здатністю мережі. Цей підхід дає змогу розробникам створювати більш чіткі та структуровані API, що полегшує їх підтримку та розвиток [2].

API Gateway стає невід'ємною частиною сучасної архітектури бекенду. Цей сервіс виконує роль єдиного входу для всіх клієнтських запитів до бекенду, забезпечуючи балансування навантаження, кешування, автентифікацію та моніторинг. Популярні рішення в цьому сегменті включають Amazon API Gateway, Apigee та Kong. Використання API Gateway дає змогу зменшити складність управління API та підвищити безпеку завдяки централізованому контролю доступу та моніторингу запитів.

Використання кешування даних, наприклад, Redis або Memcached, значно покращує продуктивність мобільних додатків, зменшуючи час відповіді сервера та знижуючи навантаження на базу даних. Це особливо важливо для додатків з високою інтенсивністю запитів, як-от соціальні мережі або платформи електронної комерції. Кешування дає змогу зберігати часто запитувані дані в пам'яті, що значно скорочує час доступу до них і покращує загальну швидкодію додатка [3].

Безпека залишається критичним аспектом бекенду мобільних додатків. Сучасні підходи включають використання OAuth для автентифікації та авторизації, шифрування даних під час передачі та зберігання, а також регулярні перевірки безпеки коду. Впровадження DevSecOps допомагає інтегрувати безпеку на всіх етапах розробки. Це означає, що безпека стає частиною процесу розробки від початку, що знижує ризики вразливостей і забезпечує захист даних користувачів.

Важливість обробки даних у реальному часі зростає, особливо для додатків, які потребують миттєвого оновлення інформації, як-от месенджери або фінансові сервіси. Використання WebSocket та технологій, як-от Firebase Realtime Database або AWS AppSync, дає змогу забезпечити миттєву синхронізацію даних між клієнтом і сервером. Це допомагає користувачам отримувати оновлення в режимі реального часу, що підвищує зручність використання додатків і їх функціональність [4].

Інтеграція штучного інтелекту (AI) та машинного навчання (ML) стає все більш поширеною у мобільних додатках. Це дає змогу створювати інтелектуальні функції, як-от рекомендаційні системи, розпізнавання зображень або обробка природної мови. Платформи, як-от TensorFlow та PyTorch, спрощують розробку та впровадження таких функцій. Впровадження AI та ML дає змогу створювати більш персоналізовані та інтерактивні додатки, що підвищує залученість користувачів і задовольняє їх потреби більш ефективно.

Сучасні тенденції у розробці бекенду мобільних додатків зосереджені на підвищенні продуктивності, гнучкості та безпеки. Серверлесс-архітектура, мікросервіси, GraphQL, API Gateway, кешування даних, безпека, обробка в реальному

часі й інтеграція AI та ML є ключовими елементами, що формують майбутнє мобільних додатків. Врахування цих тенденцій допоможе створювати більш ефективні, надійні та масштабовані рішення для користувачів.

Список використаних джерел

1. Kreibich J. A. Redis: the Definitive Guide: Data Modeling, Caching, and Messaging. O'Reilly Media, Incorporated, 2015. 340 p.

2. Biswas N. Practical GraphQL. Berkeley, CA: Apress, 2023. DOI: 10.1007/978-1-4842-9621-9 (дата звернення: 13.05.2024).

3. Lewis J., Flowler M. Microservices. 2014. URL: https://martinfowler.com/articles/microservices.html (дата звернення: 15.05.2024).

4. What is Amazon API Gateway? URL: https://docs.aws.amazon.com//developerguide/welcome.html (дата звернення: 11.05.2024).

УДК 004.738.5

Суліма В. К., *здобувач 2 курсу спеціальності 122 Комп'ютерні науки,*
Зелінська О. В., *канд. техн. наук, доцент, доцент кафедри інформаційних технологій*

СУЧАСНІ ТЕНДЕНЦІЇ FRONTEND МОБІЛЬНИХ ДОДАТКІВ

Донецький національний університет імені Василя Стуса, м. Вінниця

З розвитком мобільних технологій та зростанням попиту на мобільні додатки значно змінилися підходи до розробки frontend-частини. Frontend мобільних додатків відповідає за інтерфейс користувача та взаємодію з додатком, забезпечуючи зручність, швидкодію та привабливість. У цій публікації розглядаються основні сучасні тенденції у цій галузі.

Однією з найважливіших тенденцій є використання фреймворків та бібліотек для кросплатформенної розробки, як-от React Native та Flutter. Ці інструменти дають змогу створювати додатки для iOS та Android з використанням єдиного коду, що значно знижує витрати на розробку та підтримку. React Native, розроблений Facebook, і Flutter, створений Google, забезпечують високу продуктивність та гарний вигляд додатків [1].

Прогресивні вебдодатки (PWA) стають дедалі популярнішими завдяки своїй здатності забезпечувати функціональність, подібну до нативних додатків, використовуючи вебтехнології. PWA дають змогу встановлювати додатки безпосередньо з веббраузера, що зменшує залежність від магазинів додатків та спрощує процес оновлення. Вони також можуть працювати офлайн, що підвищує зручність для користувачів у місцях з обмеженим інтернет-з'єднанням.

Адаптивний дизайн та гнучкі макети стають стандартом у розробці мобільних додатків. Використання CSS Grid та Flexbox дає змогу створювати інтерфейси, які автоматично підлаштовуються під різні розміри екранів та орієнтацію пристроїв. Це забезпечує оптимальний користувацький досвід, незалежно від того, який пристрій використовується [2].

Поява та розвиток технологій доповненої реальності (AR) і віртуальної реальності (VR) відкриває нові можливості для мобільних додатків. AR та VR дають змогу створювати захопливі та інтерактивні користувацькі інтерфейси, що особливо актуально для ігор, навчальних додатків і додатків для електронної комерції. Використання платформ, як-от ARKit від Apple та ARCore від Google, спрощує інтеграцію цих технологій у мобільні додатки.

Використання компонентного підходу у розробці інтерфейсів набуває все більшої популярності. Бібліотеки та фреймворки, як-от React та Vue.js, дають змогу розробникам створювати повторно використовувані компоненти, що значно прискорює розробку та підвищує якість коду. Компонентний підхід також сприяє більшій модульності та зручності в обслуговуванні додатків [3].

Покращення продуктивності мобільних додатків є пріоритетом для розробників. Використання технологій, як-от Lazy Loading, дає змогу завантажувати лише необхідні частини додатка за потреби, що знижує час завантаження та покращує швидкодію. Оптимізація графіки та зменшення обсягу передаваних даних також сприяють кращому користувацькому досвіду.

Безпека мобільних додатків стає дедалі важливішою, особливо з урахуванням зростання кількості кіберзагроз. Впровадження практик безпечного кодування, використання HTTPS, шифрування даних та автентифікація користувачів є необхідними заходами для забезпечення безпеки користувацьких даних. Регулярні аудити безпеки та тестування на вразливості допомагають виявляти та усувати потенційні загрози [4].

Інтеграція штучного інтелекту (AI) та машинного навчання (ML) у мобільні додатки відкриває нові можливості для персоналізації та автоматизації.

AI та ML дають змогу створювати інтелектуальні функції, як-от рекомендаційні системи, розпізнавання облич та голосу, а також автоматичне виправлення помилок. Використання хмарних сервісів, як-от Google ML Kit та AWS AI, спрощує інтеграцію цих технологій у мобільні додатки.

Сучасні тенденції у розробці frontend мобільних додатків зосереджені на покращенні продуктивності, гнучкості та безпеки. Використання кросплатформенних фреймворків, прогресивних вебдодатків, адаптивного дизайну, технологій AR та VR, компонентного підходу, оптимізація продуктивності, забезпечення безпеки та інтеграція AI і ML є ключовими елементами, що формують майбутнє мобільних додатків. Врахування цих тенденцій дасть змогу створювати більш ефективні, привабливі та надійні рішення для користувачів.

Список використаних джерел

1. Hume D. A. Progressive Web Apps. Manning Publications, 2017. 200 p.

2. Posnett D., Filkov V., Devanbu P. An empirical study on the influence of pattern roles on change-proneness. *Proceedings of the 2013 International Conference on Software Engineering*. 2013. С. 683–692.

3. Overview of ARCore and supported development environments. URL: https://developers.google.com/ar/develop (дата звернення: 10.05.2024).

4. Get Started with React Native? URL: https://reactnative.dev/docs/environment-setup (дата звернення: 10.05.2024).

УДК 004.428/.43

Шевцов М. В., *здобувач вищої освіти 2 курсу ОС «Бакалавр» спеціальності 122 Комп'ютерні науки,*
Зелінська О. В., *канд. техн. наук, доцент, доцент кафедри інформаційних технологій*

ВЕБПРОГРАМУВАННЯ НА C++: БІБЛІОТЕКА WEB TOOLKIT

Донецький національний університет імені Василя Стуса, м. Вінниця

Сфера використання мови програмування C++ вражає: від розробки нових мов програмування, компіляторів та програмного забезпечення до розробки ігор, криптовалюти, дизайну баз даних. Як не дивно, синтаксис мови C++ також використовується у розробці вебсторінок. Однією з бібліотек для вебпрограмування на C++ є Wt (Web Toolkit) [1].

Web Toolkit (Wt) – це кросплатформна бібліотека графічного інтерфейсу в сучасному C++. За допомогою особливих графічних елементів інтерфейсу (віджетів), як-от кнопки чи текстові поля, спроєктовані для взаємодії з користувачем та відображення інформації, ви можете швидко розробляти інтерактивні вебінтерфейси, не пишучи жодного рядка JavaScript.

Розглянемо більш детально особливості цього інструменту. Wt підтримує створення додатків, які автоматично перемикаються між Ajax-версією (Asynchronous JavaScript and XML) та аналогічним HTML-кодом у разі відсутності JavaScript [1]. Якщо браузер користувача підтримує JavaScript, то Wt використовує технологію Ajax (асинхронний обмін даними між браузером і сервером без перезавантаження сторінки), що дає змогу створювати багатофункціональні вебдодатки, які працюють плавно та без перезавантаження всієї сторінки. У іншому випадку Wt генерує аналогічний HTML-код, що робить додаток незалежним від JavaScript. Ще одною приємною особливістю є те, що додатки цієї бібліотеки можуть бути підключені через клієнт-серверні протоколи, як-от HTTP, FastCGI або ISAPI, а також підтримують нововведення HTML5 (а саме переписування URL для створення REST-додатків) [2]. До того ж Wt використовує бібліотеку boost.asio, що забезпечує асинхронність операцій введення / виведення та обробки подій. У контексті першої операції, фреймворк розділяє потоки (streams), даючи змогу більш ефективного використання часу виконання, а в контексті другої – операція відбувається без створення / завершення потоків виконання (threads), забезпечуючи асинхронну взаємодію та реагування на події, як-от клік користувача чи відповідь від сервера. Також для кожної сесії на сервері зберігається певний об'єкт, який являє собою «зліпок» поточної сесії, що сприяє розділенню ресурсів між різними сесіями та користувачами, утримуючи їхні дані та взаємодію в безпечному середовищі. Бібліотека має вбудований захист від різних видів кібератак, як-от XSS, SQL-injection та DDoS [3].

З інших особливостей можна зазначити можливість запускати вебдодатки навіть на пристроях з обмеженими ресурсами, вбудовану підтримку Unicode та

локалізацію, яка полегшує розробку додатків для різних мов. Також фреймворк включає в себе систему малювання, побудови діаграм, ORM (відображення класів C++ на структури баз даних) та систему автентифікації, розширюючи можливості розробників.

Бібліотека Wt є специфічною, тому її немає у стандартному пакеті бібліотек, а отже, її треба встановити. Найзручніше це робиться на Unix-подібних системах. На Linux Ubuntu фреймворк встановлюється за допомогою пакетного менеджера apt-get командою sudo apt-get install witty witty-dev witty-doc witty-dbg witty-examples.

Універсальним є варіант клонування Wt з віддаленого репозиторію командою git clone https://github.com/emweb/wt.git. Після цього функціонал бібліотеки може використовуватись на вашому пристрої.

У висновку зазначимо, що бібліотека Wt (Web Toolkit) відкриває широкі можливості для розробки високоякісних та інтерактивних вебдодатків на мові програмування C++, а також підтримує розробку на Python та Java. Вона забезпечує розробників широким набором графічних елементів інтерфейсу та інструментів для створення багатофункціональних вебзастосунків без необхідності писати JavaScript код. До того ж Wt підтримує різні способи підключення додатків за допомогою сучасних мережевих технологій, що робить його гнучким та пристосованим до різних середовищ використання. Підтримка технологій HTML5 та асинхронний обмін даними дають змогу створювати динамічні та високоефективні вебдодатки. Найвідомішими користувачами бібліотеки Wt є компанії Intel, Philips, Epic Games та інші.

Отже, завдяки вбудованим засобам безпеки, відкритому коду та активній спільноті розробників Wt є потужним інструментом для будь-якого, хто прагне створювати сучасні та надійні вебдодатки на мові програмування C++. Його можливості поширюються від простих інтерфейсів до складних бізнес-застосунків, що робить його важливим елементом у розробці програмного забезпечення.

Список використаних джерел

1. Розробка вебдодатків на C++: вебсайт. URL: https://www.webtoolkit.eu/wt (дата звернення: 10.05.2024).

2. Web Development Essentials. 2 Linux Professional Institute. 2022. URL: https://learning.lpi.org/pdfstore/LPI-Learning-Material-030-100-uk.pdf

3. Веб-програмування. jQuery у веб-додатках. URL: https://webstudio2u.net/ua/programming/120-jquery.html (дата звернення: 10.05.2024).

УДК 004.1

Яценко В. В., *здобувач 2 курсу спеціальності 122 Комп'ютерні науки,*
Сеник І. О., *асистент кафедри інформаційних технологій*

АНАЛІЗ АЛГОРИТМІВ ІНТЕРПОЛЯЦІЇ ТА ЇХ ЗАСТОСУВАННЯ

Донецький національний університет імені Василя Стуса, м. Вінниця

Обробка й аналіз значень функції є основною задачею математичного аналізу [1]. Математичний аналіз – це розділ математики, що вивчає властивості функцій і пов'язані з ними поняття та методи. Для дослідження властивостей функції збирається певний набір даних. Набір даних – це структурована однотипна інформація, взята з, метою аналізу чи дослідження певної предметної області. Зібрані набори даних відіграють ключову роль в аналізі в різних галузях науки і подальшому прийнятті рішень. Від якості початкового набору даних залежить достовірність і точність дослідження певного явища. Набір даних може містити неповний або розріджений опис функції, що негативно вплине на якість подальшого дослідження. З метою підвищення якості результатів остаточного набору даних були створені методи, що дають змогу знаходити проміжні значення між відомими точками інтервалу, тобто методи інтерполяції [2]. Інтерполяція – це процес пошуку проміжних даних функції у відомих інтервалах. Лінійна, квадратична, Ньютона – основні види інтерполяції, що зараз застосовуються.

У статтях «Reconstruction and representation of 3D objects with radial basis functions» і «Radial basis functions for the multivariate interpolation of large scattered data sets» подається новий сучасний метод вирішення задачі інтерполяції [3, 4]. Інтерполяція за допомогою радіально-базисних функцій має значно вищу точність, і водночас оптимальну для задач швидкість. Цей метод буде використовуватись все частіше і може замінити класичні методи завдяки свій високій ефективності.

Метою дослідження є аналіз та порівняння різних методів інтерполяції для визначення оптимального для різноманітних задач. У межах роботи необхідно створити програму, що вираховуватиме час виконання алгоритмів інтерполяції і похибку. До того ж необхідно визначити переваги і недоліки алгоритмів та підсумувати відповідну інформацію.

Необхідно визначитись із функцією, на прикладі якої буде відбуватись порівняння алгоритмів. Для порівняння можна використати періодичну функцію. Можна додавати певний рівень шуму для перевірки стабільності методів. Можна обрати лінійну інтерполяцію, квадратичну інтерполяцію, кубічну інтерполяцію, інтерполяцію за методом Лагранжа, поліноміальну інтерполяцію ступеня 5 та інтерполяцію за допомогою кубічних сплайнів. Для реалізації алгоритмів будуть використані вбудовані модулі numpy і scipy. Підрахунок часу буде здійснюватись за допомогою методу timeit() модуля timeit. Для підрахунку похибки буде використовуватись середньоквадратична похибка, тобто середня квадратична різниця між інтерполяційними методами і реальними значеннями функцій.

```
import numpy as np
from scipy.interpolate import lagrange, interp1d, CubicSpline
from numpy.polynomial import Polynomial
import timeit

def f1(x):
    return np.sin(x)

def add_noise(y, noise_level=0.1):
    noise = noise_level * np.random.normal(size=y.shape)
    return y + noise

def linear_interpolation(x, y, x_new):
    linear_interp = interp1d(x, y, kind='linear')
    return linear_interp(x_new)

def quadratic_interpolation(x, y, x_new):
    quadratic_interp = interp1d(x, y, kind='quadratic')
    return quadratic_interp(x_new)

def cubic_interpolation(x, y, x_new):
    cubic_interp = interp1d(x, y, kind='cubic')
    return cubic_interp(x_new)

def lagrange_interpolation(x, y, x_new):
    lagrange_poly = lagrange(x, y)
    return lagrange_poly(x_new)

def cubic_spline_interpolation(x, y, x_new):
    cubic_spline = CubicSpline(x, y)
    return cubic_spline(x_new)

def polynomial_interpolation(x, y, x_new, degree):
    p = Polynomial.fit(x, y, degree)
    return p(x_new)

def rmse(y_true, y_pred):
    return np.sqrt(np.mean((y_true - y_pred) ** 2))
```

Рис. 1. Тестова функція, інтерполяційні методи, формула обчислення похибки

```
x = np.linspace(-10, 10, 100)
methods = [linear_interpolation, quadratic_interpolation, cubic_interpolation,
           lagrange_interpolation, cubic_spline_interpolation, polynomial_interpolation]
method_names = ["Лінійна", "Квадратична", "Кубічна", "Лагранжа", "Кубічні сплайни", "Поліноміальна"]

x_new = np.linspace(-10, 10, 1000)
degree = 5

y = f1(x)
y_new_true = f1(x_new)
print()

for method, name in zip(methods, method_names):
    print(f"Метод: {name}")

    if name == "Поліноміальна":
        execution_time = timeit.timeit(lambda: method(x, y, x_new, degree), number=1000) / 1000
        y_new = method(x, y, x_new, degree)
    else:
        execution_time = timeit.timeit(lambda: method(x, y, x_new), number=1000) / 1000
        y_new = method(x, y, x_new)

    error = rmse(y_new_true, y_new)
    print(f"Похибка: {error:.6f}, Час виконання: {execution_time:.9f} секунд")

    y_noisy = add_noise(y)
    if name == "Поліноміальна":
        execution_time = timeit.timeit(lambda: method(x, y_noisy, x_new, degree), number=1000) / 1000
        y_new_noisy = method(x, y_noisy, x_new, degree)
    else:
        execution_time = timeit.timeit(lambda: method(x, y_noisy, x_new), number=1000) / 1000
        y_new_noisy = method(x, y_noisy, x_new)
        error_noisy = rmse(y_new_true, y_new_noisy)
    print(f"Похибка з шумом: {error_noisy:.6f}")
```

Рис. 2. Побудова інтерполяційних методів, підрахунок часу і похибок

```
Метод: Лінійна
Похибка: 0.002570, Час виконання: 0.000041054 секунд
Похибка з шумом: 0.084022
Метод: Квадратична
Похибка: 0.000049, Час виконання: 0.000171076 секунд
Похибка з шумом: 0.093005
Метод: Кубічна
Похибка: 0.000003, Час виконання: 0.000149511 секунд
Похибка з шумом: 0.092741
Метод: Лагранжа
Похибка: 139887023127877113317674263384050567741440.000000, Час виконання: 0.168490793 секунд
Похибка з шумом: 2818614356871660642670012855784516773478 4.000000
Метод: Кубічні сплайни
Похибка: 0.000003, Час виконання: 0.000127358 секунд
Похибка з шумом: 0.097703
Метод: Поліноміальна
Похибка: 0.643340, Час виконання: 0.000109295 секунд
Похибка з шумом: 0.097703
```

Рис. 3. Лістинг результату роботи програми

Проаналізувавши вихідні дані, можна дійти висновків, що лінійна інтерполяція є найшвидшим методом інтерполяції, але похибка, порівняно з іншими методами, достатньо велика. Квадратична і кубічна інтерполяція мають нижчі похибки відповідно, але більший час роботи і меншу стабільність під час роботи з функцією з шумами. Метод Лагранжа показує аномально велику похибку. Це пов'язано з феноменом Рунге, коли зі зростанням ступеня полінома з'являється похибка на кінцях інтервала [5]. Ступінь полінома Лагранжа в цьому прикладі дуже великий, відтак похибка прямує до нескінченності. У випадку поліноміальної інтерполяції похибка не є настільки значимою через відносно низький ступінь полінома, але все одно значима, порівняно з іншими методами. Інтерполяція кубічними сплайнами – інтерполяція, яка використовує шматкові поліноми для кожного сегмента, – показала достатньо точний результат за оптимальний час. Ключова різниця цього методу від інших полягає у тому, що сплайни залежать лише від найближчих точок, отже, цей метод забезпечує більш гладкий і плавний графік.

У підсумку, кожен алгоритм має свої переваги і недоліки. У випадку, коли користувач працює з поліномами низького ступеня і швидкість є основним пріоритетом, лінійна інтерполяція є найкращим варіантом. Поліноміальна інтерполяція й інтерполяція методом Лагранжа достатньо повільні, але ефективні і максимально точні у простих функціях. Квадратична і кубічна інтерполяції забезпечують оптимальну швидкість із достатньо низьким рівнем відхилення даних від істинних. Метод кубічних сплайнів використовується, коли необхідна гладка і точна апроксимація функції з достатньо оптимальною швидкістю. Вибір методу повністю залежить від вимог користувача.

Список використаних джерел

1. Analysis. Mathematics. *Britannica.* URL: https://www.britannica.com/science/analysis-mathematics (дата звернення: 16.05.2024).

2. Інтерполяція. *Byjus.* URL: https://byjus.com/maths/interpolation/ (дата звернення: 16.05.2024).

3. Reconstruction and representation of 3D objects with radial basis functions / J. C. Carr; R. K. Beatson, J. B. Cherrie, T. J. Mitchell; W. R. Fright, B. C. McCallum, T. R. Evans. URL: http://mesh.brown.edu/DGP/pdfs/Carr-sg2001.pdf (дата звернення: 16.05.2024).

4. Lazzaro D., Montefusco L. B. Radial basis functions for the multivariate interpolation of large scattered data sets. *Journal of Computational and Applied Mathematics.* 2002. № 140. P. 521–536. URL: https://core.ac.uk/download/pdf/82502771.pdf (дата звернення: 16.05.2024).

5. Runge's phenomenon. *Wolfram Demonstrations Project.* URL: https://demonstrations.wolfram.com/RungesPhenomenon/ (дата звернення: 16.05.2024).

УДК 004.421:519.61](043.2)

Левченко М. Р., *здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,*
Сеник І. О., *асистент кафедри інформаційних технологій*

ПОРІВНЯЛЬНИЙ АНАЛІЗ МЕТОДІВ РОЗВ'ЯЗАННЯ СИСТЕМИ ЛІНІЙНИХ РІВНЯНЬ. МЕТОД ГАУСА ТА СІМПСОНА

Донецький національний університет імені Василя Стуса, м. Вінниця

У цьому дослідженні проводиться порівняльний аналіз ефективності і точності різних методів розв'язання систем лінійних рівнянь, зокрема методу Гауса та методу Сімпсона. Результати дослідження дають змогу виявити переваги та недоліки кожного методу залежно від характеристик системи лінійних рівнянь, як-от розмір матриці, співвідношення між числом рівнянь та невідомих, а також структура матриці. Ця робота має на меті визначити найбільш підходящий метод для конкретних умов задачі, що є важливим кроком у покращенні ефективності обчислення та точності результатів у чисельних обчисленнях.

Метод Гауса – його суть полягає в тому, що шляхом елементарних перетворень СЛАР приводять до еквівалентної системи трикутного або трапецієвидного вигляду, з якої послідовно, починаючи з останніх (за номером) змінних, знаходять усі невідомі [1].

Метод Сімпсона – це чисельний метод інтегрування, який використовується для наближеного обчислення визначених інтегралів функцій. Основна ідея полягає в апроксимації площі під кривою за допомогою квадратичних апроксимаційних функцій, відомих як параболи [2].

Порівняємо основні переваги та недоліки двох методів.

Таблиця 1 – Переваги методів

Метод Гауса	**Метод Сімпсона**
Універсальність: може застосовуватися для розв'язання широкого спектра систем лінійних рівнянь з різними характеристиками	*Висока точність:* часто дає дуже точні результати, особливо під час обчислення великих інтегралів
Стійкість: має добру стійкість до помилок округлення, що робить його надійним для використання в чисельних обчисленнях	*Простота реалізації:* має просту інтуїтивну інтерпретацію, що робить його доступним для реалізації навіть без великого досвіду в чисельних обчисленнях
Ефективність: для деяких типів систем лінійних рівнянь може бути дуже ефективним і швидким	*Можливість адаптації:* може бути легко адаптований для обчислення невизначених інтегралів та інших специфічних випадків

Таблиця 2 – Недоліки методів

Метод Гауса	Метод Сімпсона
Помірна складність: у деяких випадках може мати помірну обчислювальну складність, особливо для великих систем рівнянь	*Обмеження на класи функцій:* може бути менш ефективним для деяких класів функцій, зокрема для функцій з великими похідними або розривами
Чутливість до особливостей матриці: у разі певних особливостей матриці, як-от великі коефіцієнти або велика розбіжність між масштабами елементів, може втратити ефективність та стійкість	*Обчислювальна складність:* для деяких випадків може вимагати більше обчислювальних ресурсів, особливо для дуже складних функцій або великих інтервалів

Розглянемо практичну роботу методів у розв’занні системи лінійних рівнянь у мові Python.

```
import numpy as np
import sys

n = int(input('Введіть кількість невідомих: '))

a = np.zeros((n, n + 1))

x = np.zeros(n)

print('Введіть коефіцієнти розширеної матриці:')
for i in range(n):
    for j in range(n + 1):
        a[i][j] = float(input('a[' + str(i) + '][' + str(j) + ']='))

for i in range(n):
    if a[i][i] == 0.0:
        sys.exit('Ділення на нуль!')

    for j in range(i + 1, n):
        ratio = a[j][i] / a[i][i]

        for k in range(n + 1):
            a[j][k] = a[j][k] - ratio * a[i][k]

x[n - 1] = a[n - 1][n] / a[n - 1][n - 1]

for i in range(n - 2, -1, -1):
    x[i] = a[i][n]

    for j in range(i + 1, n):
        x[i] = x[i] - a[i][j] * x[j]

    x[i] = x[i] / a[i][i]
print('\nРозв’язок: ')
for i in range(n):
    print('X%d = %0.2f' % (i, x[i]), end='\t')
```

Рис. 1. Лістинг програми методу Гауса

```
Введіть кількість невідомих: 2
Введіть коефіцієнти розширеної матриці:
a[0][0]=1
a[0][1]=4
a[0][2]=5
a[1][0]=2
a[1][1]=8
a[1][2]=5
Ділення на нуль!
```

Рис. 2. Результат програми методу Гауса. Ділення на нуль

```
Введіть кількість невідомих: 2
Введіть коефіцієнти розширеної матриці:
a[0][0]=1
a[0][1]=2
a[0][2]=3
a[1][0]=5
a[1][1]=8
a[1][2]=1

Розв'язок:
X0 = -11.00 X1 = 7.00
```

Рис. 3. Результат програми методу Гауса

Програма реалізує метод Гаусса для розв'язання систем лінійних рівнянь. Користувач вводить кількість невідомих і коефіцієнти розширеної матриці, після чого код перетворює матрицю у верхньотрикутну форму через прямий хід, а потім виконує зворотній хід для обчислення розв'язків, які виводяться на екран.

```python
def f(x):
    return 1 / (1 + x ** 2)

def simpson13(x0, xn, n):
    h = (xn - x0) / n

    integration = f(x0) + f(xn)

    for i in range(1, n):
        k = x0 + i * h

        if i % 2 == 0:
            integration = integration + 2 * f(k)
        else:
            integration = integration + 4 * f(k)

    integration = integration * h / 3

    return integration


lower_limit = float(input("Введіть нижню межу інтеграції: "))
upper_limit = float(input("Введіть верхню межу інтеграції: "))
sub_interval = int(input("Введіть кількість підінтервалів: "))

result = simpson13(lower_limit, upper_limit, sub_interval)
print("Результат інтегрування за методом Сімпсона: %0.6f" % (result))
```

Рис. 4. Лістинг програми методу Сімпсона

```
Введіть нижню межу інтеграції: 0
Введіть верхню межу інтеграції: 1
Введіть кількість підінтервалів: 3
Результат інтегрування за методом Сімпсона: 0.720513
```

Рис. 5. Результат програми методу Сімпсона

Програма використовує метод Сімпсона для чисельного інтегрування, апроксимуючи функцію параболами на підінтервалах і сумуючи ці площі для обчислення інтегралу. Він визначає функцію $f(x)$, запитує у користувача межі інтеграції та кількість підінтервалів, обчислює значення функції у вузлових точках і використовує формулу методу Сімпсона для отримання результату.

Підсумовуючи це дослідження, зазначимо, що було проведено порівняльний аналіз методу Гаусса та методу Сімпсона, які застосовуються для розв'язання систем лінійних рівнянь та чисельного інтегрування відповідно. Метод Гауса показав високу універсальність та стійкість до помилок округлення, але може виявитися менш ефективним у разі великих систем або наявності особливостей у матриці. Метод Сімпсона натомість забезпечує високу точність інтегрування та простоту реалізації, але може бути обчислювально складним для складних функцій або великих інтервалів. Практичні приклади, реалізовані на Python, демонструють специфічні переваги та недоліки кожного методу, що підкреслює важливість вибору відповідного методу залежно від конкретних умов задачі.

Список використаних джерел

1. Верещак Р. Метод Гауса для розв'язання систем лінійних рівнянь: повний огляд. *Методи розв'язування систем лінійних алгебраїчних рівнянь*. 29.11.2023. URL: https://www.mathros.net.ua/metod-gaussa-rozvjazok-systemy-linijnyh-rivnjan-metodom-gaussa.html (дата звернення: 17.05.2024).
2. Метод Сімпсона: Основи та практичне застосування. *mathros.net.ua*. Mar. 3, 2024. URL: https://medium.com/@MathrosNetUa/метод-сімпсона-основи-та-практичне-застосування-e5803a0e9cc1 (дата звернення: 17.05.2024).
3. Gauss elimination method python program. *Codesansar*. URL: https://www.codesansar.com/numerical-methods/gauss-elimination-method-python-program.htm (дата звернення: 17.05.2024).
4. Simpson's Rule. *Mathematical Python*. URL: https://patrickwalls.github.io/mathematical-python/integration/simpsons-rule/ (дата звернення: 17.05.2024).

УДК 004.6

Кизь О. А., *здобувач 2 курсу спеціальності 122 Комп'ютерні науки,*
науковий керівник:
Якубич К. О., *асистент кафедри інформаційних технологій*

МЕТОД ГРАДІЄНТНОГО СПУСКУ

Донецький національний університет імені Василя Стуса, м. Вінниця

Методи оптимізації широко використовуються в обчислювальних задачах для пошуку найкращих рішень чи найоптимальніших параметрів. Вони допомагають знаходити мінімуми або максимуми функцій, розв'язувати задачі лінійного та нелінійного програмування, задачі розподілу ресурсів та багато інших. Для вирішення різних завдань у дослідженнях використовуються різні методи. Є алгоритми, які завершують свою роботу після певної кількості кроків, та інші, які пристосовуються до знаходження рішення на певних типах задач, і також є еврис-

тичні підходи, які можуть дати наближені розв'язки для деяких завдань, хоча не завжди точні. Ці методи мають різні алгоритми, один із яких – методи градієнтного спуску, який ми більш детально розглянемо.

Градієнтний спуск – це алгоритм оптимізації. Він використовується для покращення продуктивності нейронної мережі шляхом налаштування параметрів мережі так, щоб різниця між прогнозами мережі та фактичними / очікуваними значеннями мережі (що називаються втратою) була якомога меншою. Градієнтний спуск приймає початкові значення параметрів і використовує операції, засновані на численні, щоб налаштувати їх значення до значень, які зроблять мережу максимально точною [2]. Інакше кажучи, це один із методів оптимізації, який дає змогу нейронній мережі вчитися, а кінцева мета оптимізації – знайти такі значення параметрів, за яких функція помилки досягне свого мінімуму (рис. 1).

Розглянемо суть методу детальніше. Градієнт вказує напрям найшвидшого зменшення помилки. У контексті нейронних мереж він допомагає визначити, як зміна параметрів моделі впливає на функцію помилки. Метод використовує градієнт для пошуку шляху з найшвидшим зменшенням помилки, оновлюючи параметри моделі до досягнення мінімуму помилки. Етапами алгоритму градієнтного спуску є:

- Ініціалізація параметрів: початкові значення параметрів моделі обираються у випадковий спосіб або за певними правилами.
- Обчислення градієнта: обчислюється градієнт функції в поточній точці, вказуючи напрям найшвидшого зростання:

$$\nabla f(x_1^0, x_2^0, x_n^0) = \left(\frac{df}{dx_1}, \frac{df}{dx_2}, \frac{\partial f}{dx_n}\right),$$

де $\frac{df}{dx_i}$ – часткова похідна функції f за змінною x_i, обчислена у точці (x_1^0, x_2^0, x_n^0).

- Оновлення параметрів: параметри моделі оновлюються у напрямі, протилежному градієнту, за допомогою кроку (швидкості навчання), щоб зменшити значення функції в цій точці.
- Перевірка зупинки: проводиться перевірка критерію зупинки, який може бути досягненням максимальної кількості ітерацій, досягненням необхідної точності або іншими умовами зупинки.
- Повторення ітерацій: якщо критерій зупинки не виконано, процес повторюється, починаючи з обчислення градієнта в новій точці.

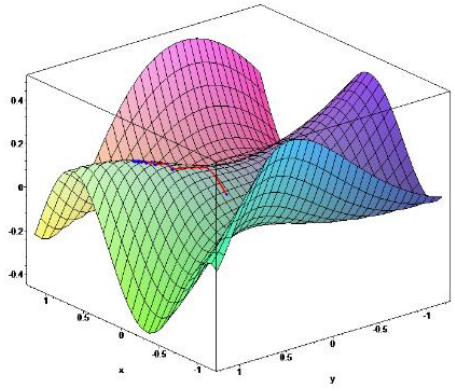


Рис. 1. Градієнтний спуск

Реалізація методу градієнтного спуску в MATLAB:

Згенеруємо випадкові дані для навчального набору:

```
X = randn(100, 2); % 100 прикладів з двома ознаками
y = randi([0, 1], 100, 1); % бінарна цільова змінна
```

Ініціалізуємо ваги нейронної мережі випадковими значеннями:

```
W = randn(2, 1);
```

Швидкість навчання:

```
learning_rate = 0.01;
```

Кількість ітерацій:

```
num_iterations = 1000;
```

Зберігаємо значення функції помилки на кожній ітерації:

```
error_history = zeros(num_iterations, 1);
```

Алгоритм градієнтного спуску для корекції ваг:

```
for i = 1:num_iterations
```

Обчислюємо прогнози моделі:

```
predictions = X * W;
```

Обчислюємо помилку (загальний квадрат помилки):

```
error = mean((predictions - y).^2);
error_history(i) = error;
```

Обчислюємо градієнт функції помилки:

```
gradient = 2 * mean(X .* repmat(predictions - y, 1, size(X, 2)), 1)';
```

Оновлюємо ваги:

```
W = W - learning_rate * gradient;
end
```

Виводимо остаточні ваги:

```
disp('Остаточні ваги:');
disp(W);
```

Виводимо графік зміни значення функції помилки на кожній ітерації:

```
plot(1:num_iterations, error_history, '-o');
xlabel('Ітерація');
ylabel('Значення функції помилки');
title('Зміна значення функції помилки');
```

Методи оптимізації в обчислювальних задачах є важливим інструментом для пошуку оптимальних рішень. Вони використовуються для знаходження мінімумів або максимумів функцій, розв'язання задач лінійного та нелінійного програмування, а також для інших завдань. Одним із методів оптимізації обчислювальних задач є метод градієнтного спуску, який дає змогу навчати нейронні мережі та знаходити оптимальні параметри. Він використовує градієнт функції помилки для оновлення параметрів моделі у напрямку найшвидшого зменшення помилки, що збільшує точність моделі та зменшує обчислювальні втрати.

Список використаних джерел

1. Градієнтний спуск: алгоритм та приклад на Python. *Robotdreams.* URL: https://robot-dreams.cc/uk/blog/331-gradiyentniy-spusk-algoritm-ta-priklad-na-python (дата звернення 28.04.2024).
2. Нельсон Д. Що таке градієнтний спуск? *UniteAI.* Серпень 23, 2020. URL: https://www.unite.ai/uk/what-is-gradient-descent/ (дата звернення 28.04.2024).
3. Лекції з дисципліни «Обчислювальна математика та програмування» для студентів денної та заочної форм навчання напряму підготовки 6.051301 «Хімічна технологія» / укл. Г. М. Дмитрієнко. Східноукраїнський національний університет імені Володимира Даля, 2010. 88 с.

УДК 004.9

Ілик В. В., *здобувач 2 курсу спеціальності 122 Комп'ютерні науки,*
науковий керівник:
Якубич К. О., *асистент кафедри інформаційних технологій*

ТЕОРІЇ АЛГОРИТМІВ У ЗАДАЧАХ ШТУЧНОГО ІНТЕЛЕКТУ

Донецький національний університет імені Василя Стуса, м. Вінниця

Вступ. Штучний інтелект (ШІ) є однією з найбільш динамічних галузей сучасної науки та техніки, яка швидко розвивається. В основі багатьох досягнень ШІ лежать теорії алгоритмів, які забезпечують ефективне розв'язання складних задач, аналіз великих обсягів даних та прийняття рішень. Розвиток алгоритмів для ШІ є критичним для прогресу у сферах обробки природної мови, розпізнавання образів, автоматизації та робототехніки.

Розробка ефективних алгоритмів для ШІ має велике значення для багатьох галузей економіки та суспільства. Зокрема, вони дають змогу автоматизувати рутинні процеси, підвищувати продуктивність, створювати нові можливості для аналізу даних та прогнозування. В умовах стрімкого зростання обсягів інформації, що потребує обробки, та складності задач, які потрібно розв'язувати, вдосконалення алгоритмів стає ключовим фактором успішного розвитку ШІ [1].

Аналіз останніх досліджень і публікацій. Останні дослідження в галузі теорії алгоритмів для ШІ зосереджені на таких напрямах:

1. Машинне навчання та глибоке навчання (алгоритми навчання на основі великих наборів даних, як-от нейронні мережі та методи підсилення).
2. Алгоритми оптимізації (методи оптимізації, включно з градієнтними методами, еволюційними алгоритмами та стохастичною оптимізацією, використовуються для покращення продуктивності моделей ШІ).
3. Алгоритми обробки природної мови (NLP) (розробка моделей для розуміння та генерації людської мови, зокрема трансформерів і моделей на основі BERT).
4. Алгоритми для автономних систем (включають плани дій і системи прийняття рішень для автономних транспортних засобів та роботів).

Метою цієї роботи є аналіз сучасних теорій алгоритмів, що використовуються в задачах штучного інтелекту, зокрема розгляд методів машинного навчан-

ня, оптимізації та обробки природної мови, а також визначення напрямів подальшого розвитку та вдосконалення алгоритмів для підвищення їх ефективності.

Виклад основного матеріалу. Основні задачі дослідження зазвичай такі:

1) визначення основних типів алгоритмів, що застосовуються в ШІ;

2) аналіз ефективності та застосовності різних алгоритмів у конкретних задачах ШІ;

3) визначення напрямів для покращення наявних алгоритмів та розробки нових методів [2, 3].

Машинне навчання (ML) є одним із основних підходів у ШІ, що базується на розробці алгоритмів, які можуть навчатися з даних. Глибоке навчання (DL) є підмножиною ML, що використовує багатошарові нейронні мережі для автоматичного вилучення характеристик з даних. Основні алгоритми включають:

– нейронні мережі (використовуються для обробки складних вхідних даних, як-от зображення та звук).

– методи підсилення (адаптивні алгоритми, що комбінують прості моделі для підвищення загальної точності).

Оптимізація є критичним компонентом для навчання моделей ШІ. Основні методи включають:

– градієнтний спуск (використовується для мінімізації функцій помилки в нейронних мережах);

– еволюційні алгоритми (застосовуються для розв’язання задач з великою кількістю локальних мінімумів).

Найсучасніші моделі NLP використовують:

– трансформери (моделі на основі архітектури трансформера, як-от BERT та GPT, які забезпечують високу точність у задачах розуміння тексту та генерації мови);

– рекурентні нейронні мережі (RNN) (використовуються для обробки послідовних даних, як-от текст або часові ряди).

Алгоритми для автономних систем включають методи планування дій та прийняття рішень:

– плани дій на основі пошуку (використовуються для визначення оптимальних маршрутів та стратегій);

– розподілені алгоритми (застосовуються в системах із багатьма агентами, наприклад, у робототехніці та автономних транспортних засобах) [4, 5].

Висновки. Алгоритми є невід’ємною частиною розвитку штучного інтелекту, забезпечуючи ефективне розв’язання складних задач. Сучасні дослідження спрямовані на вдосконалення наявних методів та розробку нових підходів для покращення продуктивності моделей ШІ. Основні напрями включають оптимізацію алгоритмів машинного навчання, розвиток нових методів обробки природної мови та впровадження ефективних алгоритмів для автономних систем.

Список використаних джерел

1. Goodfellow I., Bengio Y., Courville A. Deep Learning. 2016. MIT Press.
2. Bishop C. M. Pattern Recognition and Machine Learning. 2006. Springer.
3. Vaswani A., et al. Attention is All You Need. In Advances in Neural Information Processing Systems. 2017.

4. Russell S., Norvig P. Artificial Intelligence: A Modern Approach. 2010. Prentice Hall.
5. Sutton R. S., Barto A. G. Reinforcement Learning: An Introduction. 2018. MIT Press.

УДК 512.54:004.738:519.17

Коновалюк І. Л., *здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,*
Вєтров О. С., *старший викладач кафедри інформаційних технологій*

НУМЕРАЦІЯ ПЕРЕСТАНОВОК

Донецький національний університет імені Василя Стуса, м. Вінниця

У світі математики перестановки є однією з ключових тем, яка відіграє важливу роль у різних математичних теоріях і практичних застосуваннях. Поняття перестановки виникає в контексті упорядкування скінченних множин, де кожен елемент займає своє визначене місце. Ця тема має глибокі корені і знаходиться у центрі досліджень комбінаторики, теорії ймовірностей, теорії графів та ін.

Перестановкою (the permutation) із n елементів називається будь-яка скінченна послідовність, яка одержується внаслідок упорядкування деякої скінченної множини, складеної з n елементів. Число всіх можливих перестановок із n елементів позначається як P_n [1]. Це число дорівнює добутку всіх цілих чисел від 1 до n. Число можливих перестановок із n елементів певного числового набору дорівнює добутку натуральних чисел від 1 до n.

Наприклад, якщо маємо множину з 3 елементів, кількість можливих перестановок буде:

$$P_3 = 3! = 3 \times 2 \times 1 = 6.$$

Розрахунок перестановок дає нам змогу з'ясувати різні варіанти розташування елементів набору, забезпечуючи водночас незмінну їх кількість. Кількість перестановок позначається як P_n, де n – кількість елементів множини. Перестановки обчислюються за формулою $P_n = n!$:

$$\begin{pmatrix} 1\,4\,3\,2 \\ 3\,1\,2\,4 \end{pmatrix} \quad \begin{pmatrix} 1\,2\,3\,4\,5 \\ 3\,1\,2\,4\,5 \end{pmatrix}.$$

Рис. 1. Приклад перестановок двох наборів чисел

Перестановки – це аранжування елементів у певному порядку. Вони можуть бути корисні в різних областях математики, комп'ютерних наук, криптографії та багатьох інших сферах. Один із видів перестановок – нумерація перестановок. Це спосіб присвоєння унікального числа кожній можливій перестановці набору елементів.

Наприклад, розглянемо набір {1, 2, 3}. Його можна переставити у шість можливих способів: {1, 2, 3}, {1, 3, 2}, {2, 1, 3}, {2, 3, 1}, {3, 1, 2}, {3, 2, 1}. Нумерація перестановок дає змогу присвоїти кожній із цих перестановок унікальний номер або індекс, що дуже корисно під час обчислень та аналізу.

Формули для обчислення числа можливих перестановок та для визначення індексу конкретної перестановки можуть змінюватися залежно від конкретного

методу нумерації. Однак у загальному випадку число можливих перестановок n елементів може бути обчислене як n факторіал, тобто $n!$.

Нумерація перестановок знайде застосування у криптографії для генерації ключів шифрування, у комбінаториці для розв'язання задач на обчислення кількості можливих розташувань, у програмуванні для створення ефективних алгоритмів сортування та багато інших областях, де потрібно аналізувати або працювати зі змінними наборами елементів.

Одна з ключових формул для обчислення індексу перестановки у нумерації перестановок – це формула Лейхарда–Роуза. Для заданої перестановки π довжини n ця формула визначає її індекс $P(\pi)$ за таким правилом:

$$P(\pi) = \sum_{i=1}^{n} ((\pi(i) - 1) \cdot (n - i)!) .$$

Тут $\pi(i)$ – це i-тий елемент перестановки a $(n - i)!$ $(n - i)!$, факторіал решти незафіксованих елементів після i-того.

Ця формула є ключовою у роботі з нумерацією перестановок, оскільки вона дає змогу ефективно перетворювати перестановки у відповідні числові індекси та навпаки.

Поняття множини – одне з невизначених понять у математиці (наприклад, множина натуральних чисел). Об'єкти, з яких складається множина, називаються елементами множини. Множина, що складається зі скінченної кількості елементів, називається скінченною множиною. Такими множинами є: множина всіх двоцифрових чисел, множина вершин даного многокутника і множина його діагоналей. Множина, що містить нескінченну кількість елементів, називається нескінченною множиною. Нескінченна множина – це множина всіх натуральних чисел. Множина, яка не містить елементів, називається порожньою.

Множина – одне з найважливіших понять сучасної математики. Поняття множини введено аксіоматично як сукупність певних об'єктів довільної природи [2], і тому множину не можна означити, застосовуючи інші означені поняття. Навпаки, поняття «множина» охоплює безліч інших понять, що виходять за межі сфери математики. Ми можемо посилатися на набір книг у конкретній бібліотеці, набір літер, із яких складається український алфавіт, набір усіх розв'язків даного рівняння, набір різноманітних геометричних фігур або навіть набір, що складається з інших наборів.

Одним із найважливіших застосувань перестановок є їх застосування в криптографії. У криптографії перестановки використовуються для створення паролів, які захищають конфіденційні дані. Наприклад, шифр перестановки змінює порядок символів у повідомленні на основі певного ключа, що ускладнює доступ до нього без ключа.

Ще однією важливою областю застосування перестановок є комбінаторика. У комбінаториці перестановки використовуються для вирішення різноманітних задач щодо розташування об'єктів.

У сфері алгоритмів перестановки використовуються для сортування даних, пошуку оптимальних рішень та інших операцій обробки даних. Наприклад, алгоритм швидкого сортування використовує перестановки для швидкого сортування

масиву. До того ж перестановки використовуються в алгоритмах, які оптимізують розташування даних у пам'яті комп'ютера.

У математичному моделюванні перестановки використовуються для аналізу різних систем і процесів. Наприклад, їх можна використовувати для вивчення маршрутизації, аналізу графів та інших математичних структур у комп'ютерних мережах.

До того ж аранжування можуть знайти застосування в теорії музики, для аналізу та створення музичних творів. Наприклад, порядок нот у мелодії або розташування музичних фраз можна відобразити аранжуваннями.

Усі ці приклади демонструють важливість і універсальність механізмів у різних галузях науки і техніки, що робить їх незамінними інструментами для аналізу, обробки та захисту даних, а також для вирішення різноманітних математичних і практичних задач. Отже, перестановки виявляються важливим та універсальним інструментом у багатьох галузях науки та технологій. Розуміння їх концепції та застосування дає змогу розв'язувати різноманітні математичні та практичні задачі, що робить їх незамінним елементом у сучасному світі.

Список використаних джерел

1. Васильків І. М. Основи теорії ймовірностей і математичної статистики: навч. посібник. Львів: ЛНУ імені Івана Франка, 2020.

2. Дороговцев А. Я. Математичний аналіз. Частина 1. Київ: Либідь, 1993.

3. Ferreiros J. Labyrinth of Thought: A History of Set Theory and Its Role in Modern Mathematics. Birkhäuser Basel, 2007.

УДК 519.642 004.424.5.021

Левченко М. Р., *здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,*
Вєтров О. С., *старший викладач кафедри інформаційних технологій*

ІСТОРІЯ ТА РОЗВИТОК АЛГОРИТМУ ШВИДКОГО СОРТУВАННЯ

Донецький національний університет імені Василя Стуса, м. Вінниця

Швидке сортування (QuickSort) – це один із найпотужніших та найпоширеніших алгоритмів сортування, який має широке застосування у комп'ютерних науках та програмуванні. Саме його унікальна комбінація простоти реалізації та високої ефективності робить QuickSort об'єктом численних досліджень і вдосконалень з моменту його створення [1]. Алгоритм QuickSort, або ж qsort, був розроблений понад 40 років тому британським комп'ютерним вченим Тоні Хоаром у 1960 році під час його роботи в московському університеті. На той час основною задачею було створення алгоритму сортування, який був би ефективнішим за наявні методи. Хоар спочатку розробив QuickSort для сортування слів у словнику англійської мови, що стало основою для подальших досліджень та вдосконалень алгоритму [2].

Алгоритм працює за принципом «розділяй і володарюй» – потрібно розбивати масив і застосовувати той самий алгоритм до кожної частини, які поступово зменшуватимуться. Основна ідея алгоритму така:

- З масиву вибирається елемент, який називається опорним (pivot).
- Потім масив ділиться на дві частини: одна частина містить усі елементи, менші або рівні опорному елементу, а друга – всі елементи, більші за опорний елемент.
- Для кожного підмасиву, якщо в ньому більше двох елементів, рекурсивно виконується та сама процедура. Якщо в підмасиві два елементи, вони порівнюються між собою і, якщо потрібно, міняються місцями.
- Після виконання цих кроків ми отримаємо повністю відсортований масив [3].

Протягом років алгоритм зазнав численних вдосконалень:

- Вибір середнього елемента, медіани трьох або випадкового елемента для уникнення найгірших випадків.
- Комбінація QuickSort з іншими алгоритмами, як-от вставка (Insertion Sort), для покращення продуктивності на малих підмасивах.

Розробка багатопоточних реалізацій QuickSort застосовується для ефективного використання багатоядерних процесорів [1]. У цього методу, як і в будь-якого іншого методу сортування, є свої переваги та недоліки.

Перевагами QuickSort є:

- висока швидкодія на практиці. QuickSort є одним із найшвидших алгоритмів внутрішнього сортування: потребує лише O(n) пам’яті, для покращеної версії – O(1), у найкращому випадку складність становить $\Omega(n \log n)$;
- дає змогу розпаралелювання для сортування підмасивів;
- працює на пов’язаних списках.

Серед характерних недоліків можна зазначити нестійкість (не зберігає відносний порядок однакових елементів) та сильну деградацію обчислювальної складності за умови невдалих вхідних даних, $O(n^2)$.

Розглянемо приклад використання алгоритму QuickSort у мові програмування Python [1]. Цей код реалізує алгоритм швидкого сортування (QuickSort) для сортування списку чисел. Спочатку визначається опорний елемент (pivot), і масив розділяється так, що всі елементи, менші або рівні опорному, переміщаються ліворуч, а всі більші – праворуч. Потім функція QuickSort рекурсивно сортує підмасиви ліворуч і праворуч від опорного елемента. Основна програма ініціалізує список чисел, виводить його, виконує сортування за допомогою QuickSort і виводить відсортований список.

Підсумовуючи, можемо зазначити, що історія QuickSort демонструє, як інноваційні ідеї та постійне вдосконалення можуть зробити алгоритм ключовим інструментом у комп’ютерних науках. Завдяки своїй ефективності та гнучкості QuickSort залишається актуальним і незамінним для вирішення завдань сортування в різних галузях.

Список використаних джерел

1. Розбираємо швидке сортування. *EPAM. Campus.* 27.09.2021. URL: https://training.epam.ua/ua/blog/483 (дата звернення: 20.05.2024).

2. Python Program for QuickSort. *geeksforgeeks.org.* 28.08.2024. URL: https://www.geeksforgeeks.org/python-program-for-quicksort/ (дата звернення: 20.05.2024).

3. Швидке сортування. *Wikiwand.* URL: https://www.wikiwand.com/uk/Швидке_сортування (дата звернення: 20.05.2024).

УДК 004.41

Токар С. С., *здобувач 2 курсу спеціальності 122 Комп'ютерні науки,*
Вєтров О. С., *старший викладач кафедри інформаційних технологій*

РЕАЛІЗАЦІЯ МЕТОДУ ШУЛЬЦЕ МОВОЮ ПРОГРАМУВАННЯ JAVASCRIPT

Донецький національний університет імені Василя Стуса, м. Вінниця

Метод Шульце – це система голосування, розроблена Маркусом Шульце у 1997 році, яка обирає одного переможця за допомогою голосів, які виражають переваги. Метод також можна використовувати для створення відсортованого списку переможців [1, 2]. У цій роботі ми розглянемо імплементацію методу Шульце з використанням мови програмування JavaScript.

Цей метод базується на ідеї попарного порівняння кандидатів. У ньому визначається «сила» взаємного переважання кожного кандидата над іншими. З цієї інформації формується граф переваг, який використовується для визначення переможця.

Кожен бюлетень містить повний список усіх кандидатів. Кожен виборець ранжує цих кандидатів у порядку переваги. Окремий виборець може віддати однакову перевагу кільком кандидатам і залишити кандидатів без рейтингу. Якщо даний виборець не ранжує всіх кандидатів, то вважається, що цей виборець віддає перевагу всім кандидатам із рейтингом перед усіма кандидатами без рейтингу, і що цей виборець байдужий до всіх кандидатів без рейтингу.

Розглянемо кроки методу Шульце:

1. Створення матриці парних порівнянь. Кожен виборець виражає свої переваги щодо кандидатів. Ця інформація збирається і утворює матрицю, де кожен елемент [i] [j] відповідає кількості виборців, які віддали перевагу кандидату i над кандидатом j.

2. Побудова графу переваг. На основі матриці парних порівнянь будується граф, де ребра вказують на те, які кандидати мають більшість переваг над іншими.

3. Визначення найсильніших шляхів. З використанням алгоритму Флойда–Уоршелла знаходяться найсильніші шляхи між усіма парами кандидатів у графі.

4. Визначення переможця. Найкращий кандидат визначається як той, у якого немає конкурента, що має сильніший шлях відносно всіх інших кандидатів.

У мові програмування JavaScript ми можемо реалізувати цей метод так. Спочатку створюємо функцію для обчислення методу Шульце. Потім створюємо початкову структуру даних (матрицю) для зберігання найсильніших шляхів між кандидатами в методі Шульце на основі вхідних даних.

Використовуємо алгоритм Флойда–Уоршелла для знаходження найсильніших шляхів між кандидатами у графі. Цей процес повторюється для всіх комбінацій кандидатів i, j та k, щоб гарантувати, що найсильніші шляхи враховують усі можливі проміжні кандидати. В результаті отримується матриця найсильніших шляхів між усіма кандидатами. Зрештою, програмно визначаємо переможця виборів за методом Шульце на основі матриці найсильніших шляхів між кандидатами.

Отже, метод Шульце є ефективним способом визначення переможця у виборах з великою кількістю кандидатів та виборців. Його реалізація за допомогою мови програмування JavaScript дає змогу легко використовувати цей метод у вебдодатках та інших проєктах, що вимагають проведення голосування.

Список використаних джерел

1. Schulze method. *Electowiki.* 14 July 2023. URL: https://electowiki.org/wiki/Schulze_method (дата звернення: 20.05.2024).

2. Schulze method. *Wikipedia.* 6 September 2024. URL: https://en.wikipedia.org/wiki/Schulze_method (дата звернення: 20.05.2024).

УДК 004.6

Поліщук О. С., *здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,*
Поремський Ю. В., *канд. техн. наук, старший викладач кафедри інформаційних технологій*

ТЕОРЕТИЧНІ ЗАСАДИ ХЕШУВАННЯ ДАНИХ

Донецький національний університет імені Василя Стуса, м. Вінниця

Хешуванням даних називається операція перетворення вхідного масиву даних довільної довжини у вихідний бітовий рядок фіксованої довжини за допомогою деякого визначеного алгоритму. Перетворення вхідного масиву даних відбувається за допомогою спеціальної функції, яку називають хеш-функцією, або функцією згортки [1].

Вхідні дані, на які діє хеш-функція, називаються ключем (інколи повідомленням). Результат дії хеш-функції на вхідні дані називається хеш-значенням, хеш-кодом або просто хешем. Хеш – це натуральне число, яке часто записують у шістнадцятковому вигляді [1].

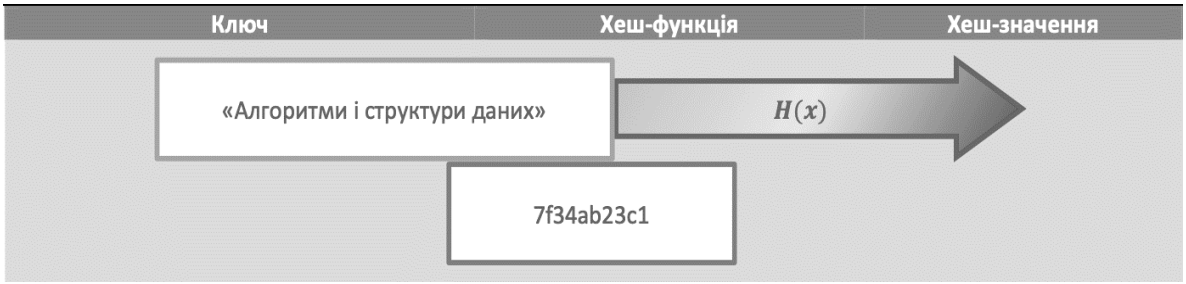


Рис. 1. Хешування рядка

Ідеальною хеш-функцією є така хеш-функція, яка для будь-яких двох неоднакових ключів дає неоднакові адреси. Підібрати таку функцію можна у випадку, якщо всі можливі значення ключів відомі наперед. Така організація даних має назву «досконале хешування».

Для реалізації хешування необхідні три речі:

- структура даних (хеш-таблиця) для зберігання даних;
- функція хешування, яка встановлює відповідність між значеннями ключа і розміщенням в таблиці;
- алгоритм вирішення конфліктів, який визначає послідовність дій, якщо декілька ключів відповідають одній комірці таблиці [2].

Простий приклад практичної реалізації хешування:

```
using System;
using System.Security.Cryptography;
using System.Text;

class Program
{
   static void Main(string[] args)
   {
      // Приклад хешування рядка "Hello, World!"
      string input = "Hello, World!";
      string hashedString = ComputeHash(input);
      Console.WriteLine($"Input: {input}");
      Console.WriteLine($"Hash: {hashedString}");
   }

   static string ComputeHash(string input)
   {
      // Створення екземпляру об’єкта MD5
      using (MD5 md5 = MD5.Create())
      {
         // Конвертування вхідного рядка у байтовий масив
         byte[] inputBytes = Encoding.UTF8.GetBytes(input);

         // Обчислення хеша для вхідних даних
         byte[] hashBytes = md5.ComputeHash(inputBytes);

         // Перетворення байтів хеша у рядок шістнадцяткового представлення
         StringBuilder sb = new StringBuilder();
         for (int i = 0; i < hashBytes.Length; i++)
         {
            sb.Append(hashBytes[i].ToString("x2"));
         }

         return sb.ToString();
      }
   }
}
```

```
Input: Hello, World!
Hash: 65a8e27d8879283831b664bd8b7f0ad4
```

Рис. 2. Результат програмного коду

Для розв’язання колізій використовуються різноманітні методи, які в основному зводяться до методів «ланцюжків» і «відкритої адресації».

Пошук у хеш-таблиці з ланцюжками переповнення здійснюється так. Спочатку обчислюється адреса за значенням ключа. Потім здійснюється послідовний пошук у списку, який зв’язаний з обчисленою адресою. Процедура вилучення з таблиці зводиться до пошуку елементу і його вилучення з ланцюжка переповнення.

Метод відкритої адресації полягає в тому, щоб, користуючись якимось алгоритмом, який забезпечує перебір елементів таблиці, переглядати їх у пошуках вільного місця для нового запису.

Очевидно, що в міру заповнення хеш-таблиці будуть відбуватися колізії, і внаслідок їх розв’язання методами відкритої адресації чергова адреса може вийти за межі адресного простору таблиці. Щоб це явище відбувалося рідше, можна піти на збільшення розмірів таблиці, порівняно з діапазоном адрес, які обчислюються хеш-функцією.

З одного боку, це приведе до скорочення кількості колізій і прискорення роботи з хеш-таблицею, а з іншого – до нераціональних витрат адресного простору. Навіть у разі збільшення таблиці удвічі, порівняно з областю значень хеш-функції, нема гарантій того, що внаслідок колізій адреса не перевищить розмір таблиці. Водночас у початковій частині таблиця може залишатися достатньо вільних елементів. Тому на практиці використовують циклічний перехід на початок таблиці.

Для того, щоб попередньо оцінити якість хеш-функції, можна провести імітаційне моделювання. Формується вектор цілих чисел, довжина якого співпадає з довжиною хеш-таблиці. Випадково генерується достатньо велика кількість ключів, для кожного ключа обчислюється хеш-функція. В елементах вектора підраховується кількість генерацій даної адреси. За результатами такого моделювання можна побудувати графік розподілу значень хеш-функції.

Якщо кількість елементів таблиці достатньо велика, то графік будується не для окремих адрес, а для груп адрес. Великі нерівномірності засвідчують високу імовірність колізій в окремих місцях таблиці. Зрозуміло, така оцінка є наближеною, але вона дає змогу попередньо оцінити якість хеш-функції і уникнути грубих помилок під час її побудови.

Оцінка буде більше точною, якщо генеровані ключі будуть більш близькими до реальних ключів, які використовуються під час заповнення хеш-таблиці. Для символьних ключів дуже важливо добитися відповідності генерованих кодів символів тим кодам символів, які є в реальному ключі. Для цього потрібно проаналізувати, які символи можуть бути використані в ключі.

Список використаних джерел

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

2. Воробйова О. Д., Глазунова Л. В. Алгоритми пошуку, стиснення даних, внутрішнього та зовнішнього сортування, алгоритми на графах. Одеса: ОНАЗ ім. О. С. Попова, 2017. 52 с.

УДК 004.9

Грибаніна А. О., *здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,*
Поремський Ю. В., *канд. техн. наук, старший викладач кафедри інформаційних технологій*

МЕТОДИ ВІДОКРЕМЛЕННЯ КОРЕНІВ ПІД ЧАС РОЗВ'ЯЗАННЯ НЕЛІНІЙНИХ РІВНЯНЬ

Донецький національний університет імені Василя Стуса, м. Вінниця

Математичними моделями багатьох об'єктів і процесів навколишнього світу є нелінійні рівняння і системи нелінійних рівнянь: алгебраїчні і трансцендентні – для сталих станів, диференціальні – для динамічних процесів.

Розв'язання нелінійних рівнянь виду $f(x) = 0$ означає знайдення такого $x \in R^1$. Водночас x називають коренем рівняння.

Класифікація нелінійних рівнянь наведена на рис. 1. Розглянемо означення та приклади різних типів нелінійних рівнянь.

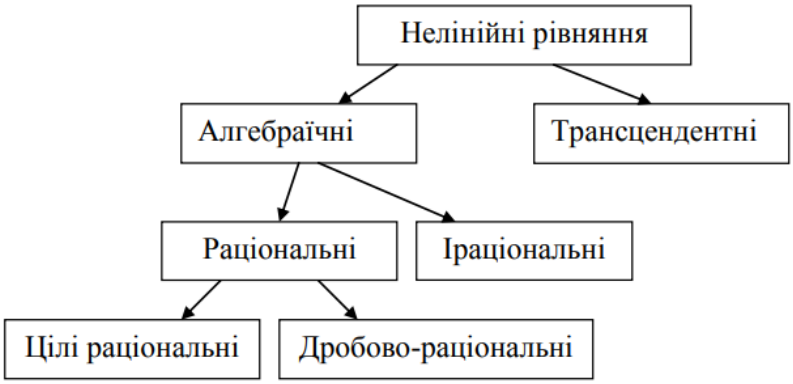


Рис. 1. Класифікація нелінійних рівнянь

Функція є алгебраїчною, якщо для отримання значення функції за даним значенням x треба виконати арифметичні операції і операції піднесення до степеня з раціональним показником.

Алгебраїчна функція є раціональною відносно змінної x, якщо над x не виконується ніяких інших дій, крім додавання, віднімання, множення, ділення і піднесення до цілого степеня.

Якщо в раціональній функції хоча б один раз зустрічається ділення на змінну x або змінна x не входить у вираз, який є дільником, то така функція є дробово-раціональною.

Функція є ірраціональною, якщо для отримання значення функції за даним значенням x необхідно виконати, крім чотирьох арифметичних дій (усіх або деяких), ще й обчислення кореня. Водночас функція буде ірраціональною лише тоді, коли аргумент x знаходиться під знаком радикала.

Другий великий клас функцій – трансцендентні функції. До них належать усі неалгебраїчні функції: показникова a^x; логарифмічна $\log_a x$; тригонометричні $\sin x$, $\cos x$, $\tan x$, $\cot x$.

Відокремити корінь рівняння – означає знайти такий інтервал, в середині якого є корінь даного рівняння, і цей корінь – єдиний на даному інтервалі. Алгоритм відокремлення коренів розбивається на такі кроки:

1. Розбивається деяка область на деяку кількість рівних відрізків $[x_i, x_{i+1}]$, де x_0 задано, $x_{i+1} = x_i + h$, h– крок розбиття, $i = 0, \dots, n$.

2. Визначається знак функції $f(x)$ на кінцях кожного i-го відрізка $[x_i, x_{i+1}]$.

3. Якщо добуток $f(x_i)f(x_{i+1})$ доданий і h достатньо мале, то можна сподіватися, що на відрізку $[x_i, x_{i+1}]$ коренів рівняння немає.

4. Якщо добуток $f(x_i)f(x_{i+1})$ від'ємний, то на відрізку $[x_i, x_{i+1}]$ існує розв'язок рівняння, хоча можливо, що він не один.

5. Перевіряється, чи змінює знак похідна $f'(x)$ на кінцях відрізка $[x_i, x_{i+1}]$. Якщо h достатньо мале і знак похідної $f'(x)$ на кінцях відрізка $[x_i, x_{i+1}]$ не змінюється, то можна сподіватися, що на відрізку корінь один. Отже, корінь рівняння відокремлений.

6. Якщо знак похідної $f'(x)$ змінюється, то впевненості, що корінь один, немає.

7. Відрізки, для яких немає упевненості в тому, що розв'язок рівняння лише один, продовжують розбивати вже з меншим кроком, тобто повторюється для них описана процедура відокремлення коренів.

Отже, відокремлення коренів є початковим кроком під час наближеного аналізу нелінійних рівнянь, розв'язок яких далі потребує використання методів, якот метод дихотомії, ітерації, хорд, дотичних.

Список використаних джерел

1. Комп'ютерне моделювання систем та процесів. Методи обчислень. Частина 1: навчальний посібник / Р. Н. Квєтний, І. В. Богач, О. Р. Бойко, О. Ю. Софина, О. М. Шушура. Вінниця: ВНТУ, 2012. 193 с. URL: http://kist.ntu.edu.ua/textPhD/kmsp.pdf

2. Чисельні методи: навчальний посібник / Л. О. Волонтир, О. В. Зелінська, Н. А. Потапова, І. А. Чіков. Вінниця: ВНАУ. 2020 322 с.

УДК 004.6

Богач Т. О., *здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,*
Поремський Ю. В., *канд. техн. наук, старший викладач кафедри інформаційних технологій*

ІТЕРАЦІЙНІ МЕТОДИ ТА ЇХ ЗАСТОСУВАННЯ У ВИРІШЕННІ СИСТЕМ ЛІНІЙНИХ АЛГЕБРАЇЧНИХ РІВНЯНЬ

Донецький національний університет імені Василя Стуса, м. Вінниця

У сучасному світі чисельні методи роблять значний внесок у розв'язання різноманітних задач, від фінансового моделювання до складних інженерних розрахунків та задач комп'ютерних наук. Серед цих задач розв'язання систем лінійних

алгебраїчних рівнянь (СЛАР) має особливе місце, адже воно є фундаментом для багатьох галузей науки та техніки. Ітераційний метод розв'язання систем лінійних алгебраїчних рівнянь (СЛАР) є потужним інструментом для пошуку наближених розв'язків. Він відрізняється від прямого методу тим, що починається з наближеного правильного розв'язку і, якщо обчислювальний процес збігається, після кожного кроку буде отримано послідовність наближених розв'язків, що стають усе ближче до справжнього після кожного кроку [1].

Під час розв'язування СЛАР виникає проблема знаходження балансу між обчислювальною складністю і точністю розв'язку. Існують прямі методи, які можуть забезпечити точний розв'язок СЛАР за обмежену кількість часу, але вони не ефективні для великих систем або складних структур. З іншого боку, ітераційні методи на основі повторних обчислень можуть бути ефективними для великих СЛАР, але збіжність цих методів вимагає більше часу і може бути менш стійкою. Отже, необхідно знайти оптимальний баланс між точністю, швидкістю та обчислювальною складністю для розв'язування СЛАР у чисельних обчисленнях. Це вимагає розробки та застосування різних методів залежно від конкретної задачі і вимог.

Ефективні методи розв'язання систем лінійних алгебраїчних рівнянь (СЛАР) мають велике значення у багатьох областях науки та техніки, включно з фінансовою математикою, інженерією та комп'ютерними науками.

- Фінансова математика виникла в середині 1980-х років, коли дослідники математики зацікавилися проблемами, пов'язаними зі стохастичним керуванням, які до того часу вивчалися переважно економістами. Ця область використовує квантитативні методи для вирішення завдань, як-от ціноутворення фінансових інструментів, включно деревовими методами, скінченними різницями для рівнянь з частинними похідними та методом Монте-Карло [2].
- У сучасному інженерному світі, де моделювання складних систем, обробка сигналів та аналіз даних відіграють ключову роль у проєктуванні та розробці, ефективні методи розв'язання систем лінійних алгебраїчних рівнянь (СЛАР) стають невід'ємною частиною успіху. Математичні концепції та інструменти, як-от інтеграл, алгебра та статистика, дають змогу інженерам робити точні розрахунки та вимірювання, зменшуючи ризик помилок у проєктуванні та аналізі [3].
- У комп'ютерних науках, де чисельні методи є розповсюдженими для розв'язання різних задач як оптимізації програм, так і обробки великих обсягів даних, ефективні методи розв'язання СЛАР є важливим елементом для розробки швидких та ефективних алгоритмів. Чисельні методи, включно з ітераційними методами, є важливим інструментом у комп'ютерних науках. Вони використовуються для розв'язання СЛАР, оптимізації, обробки зображень, машинного навчання та багатьох інших завдань.

Здійснимо огляд різних підходів до побудови ітераційних методів, як-от метод Якобі, метод Гауса–Зейделя, метод релаксації.

Метод ітерації – це чисельний і наближений метод розв'язання СЛАР. Основна ідея ітераційних методів полягає в тому, щоб знайти за наближеним значенням величини наступного наближення, яке є точнішим. Тобто зближаються до точного розв'язку з кожною ітерацією. Цей процес триває доти, поки розв'язок не збли-

зиться до потрібної точності або не буде досягнута максимальна кількість ітерацій. Метод дає змогу отримати значення коренів системи із заданою точністю у вигляді межі послідовності деяких векторів (ітераційний процес). Характер збіжності і сам факт збіжності методу залежить від вибору початкового наближення кореня x_0 [4].

Метод Якобі – один з ітераційних методів наближення розв'язку системи n лінійних рівнянь у n змінних. Ітераційний метод Якобі розглядається як ітераційний алгоритм, який використовується для визначення розв'язків системи лінійних рівнянь у чисельній лінійній алгебрі, яка є діагонально домінуючою. У цьому методі для кожного діагонального елементу підставляється наближене значення. Поки воно не збігається, процес повторюється. Цей алгоритм був вперше названий процесом перетворення Якобі під час діагоналізації матриці. Метод Якобі також відомий як метод одночасних зсувів [5]:

$$a_{11}x_1 + a_{12}x_2 + a_{1n}x_n = b_1;$$
$$a_{n1}x_1 + a_{n2}x_2 + a_{nn}x_n = b_n.$$

У методі Гауса–Зейделя кожна змінна оновлюється на основі найновіших значень інших змінних. Ця модифікація часто призводить до вищого ступеня точності за меншу кількість ітерацій. У методі Якобі значення змінних не змінюються до наступної ітерації, тоді як у методі Гаусса–Зейделя значення змінних змінюються, як тільки оцінюється нове значення. Наприклад, у методі Якобі значення $x_i(k)$ не змінюється до $(k+1)$ ітерації, а в методі Гаусса–Зейделя значення $x_i(k)$ змінюється лише в k-й ітерації [6].

Ітераційні методи широко використовуються в різних галузях, включно з фінансовою математикою, обробкою сигналів, машинним навчанням та ін.

- Фінансова математика. У фінансовій математиці ітераційні методи використовуються для оцінки ціни фінансових інструментів, як-от опціони та деривативи. До прикладу, методи релаксації можна використовувати для розрахунку цін опціонів або моделювання ризику у фінансових портфелях.
- Обробка сигналів. В обробці сигналів ітераційні методи використовуються для вирішення проблем фільтрації та відновлення сигналів. До прикладу, методи Гаусса–Зейделя можна використовувати для реконструкції зображень або фільтрації сигналів у медичній діагностиці.
- Машинне навчання. У машинному навчанні ітераційні методи використовуються для навчання моделей та оптимізації параметрів. До прикладу, метод стохастичного градієнтного спуску – це ітераційний метод, який використовується для оптимізації ваг моделей у задачах класифікації та регресії.

Ітераційні методи розв'язання лінійних алгебраїчних систем рівнянь мають переваги перед прямими методами. Вони є ефективними для великих систем, оскільки вимагають менше обчислювальних ресурсів і працюють із матрицями у вигляді векторно-матричних операцій. Багато ітераційних методів можна розпаралелити, що дає змогу використовувати паралельні обчислення для прискорення процесу розв'язання системи лінійних рівнянь (СЛАР) на багатоядерних і розподілених системах. Ітераційні методи допускають різні модифікації та оптимізації для збільшення швидкості збіжності або врахування специфічних характеристик

завдання. Вони також можуть бути ефективними для розв’язування невизначених або погано обумовлених систем. До того ж ітераційні методи мають застосовність до складних проблем, де прямі методи можуть бути непрактичними або неефективними. Загалом ітераційні методи забезпечують більшу гнучкість, адаптивність і потенціал для розв’язання різних типів лінійних алгебраїчних систем рівнянь.

Ітераційні методи мають свої переваги, проте вони також мають обмеження, які треба враховувати під час їх використання. Деякі ітераційні методи можуть бути нестійкими за певних умов, що призводить до повільної або відсутньої збіжності, якщо початкове наближення вибрано неправильно або система має особливості. До того ж методи можуть вимагати додаткових обчислень або збереження додаткових даних, що призводить до збільшення обчислювальних витрат та вимог до пам’яті. Ефективність ітераційних методів також може залежати від вибору початкового наближення, і неправильний вибір може призвести до повільної збіжності чи навіть до розбіжності методу. Деякі ітераційні методи можуть проявляти нестабільність під час обчислення, особливо якщо матриця системи має аномалії або втрату точності.

Отже, ітераційні методи є потужним інструментом для розв’язання систем лінійних алгебраїчних рівнянь. Їх використання поширене у різних галузях, як-от фінансова математика, обробка сигналів та машинне навчання. Основні переваги ітераційних методів полягають у здатності працювати з великими і розрідженими матрицями, економії пам’яті та можливості паралельного обчислення. Проте важливо враховувати обмеження цих методів, як-от нестійкість у певних умовах, потреба в додаткових даних та обмежена працездатність для деяких типів систем. У майбутньому дослідження в цій галузі можуть спрямовуватися на поліпшення ефективності, стійкості та швидкості збіжності ітераційних методів. Це може бути досягнуто через розробку нових методів або оптимізацію наявних, використання специфічних властивостей систем та покращення алгоритмів паралельних обчислень. Також можливе дослідження нових застосувань ітераційних методів для розв’язання конкретних задач у різних галузях, що сприятиме подальшому розвитку цієї важливої галузі чисельних обчислень.

Список використаних джерел

1. Елементи комп’ютерного моделювання. Лекція № 9. Ітераційні та прямі методи. *Кафедра обчислювальної математики факультету кібернетики Київського національного університету імені Тараса Шевченка.* URL: http://om.univ.kiev.ua/users_upload/15/upload/file/cm_lecture_09.pdf

2. The Crucial Role of Mathematics in Engineering / Prof. D. Saranya. September 12, 2023. URL: https://www.bitsathy.ac.in/the-crucial-role-of-mathematics-in-engineering/ (дата звернення: 20.05.2024).

3. Чисельні методи: навчальний посібник / Л. О. Волонтир, О. В. Зелінська, Н. А. Потапова, І. А. Чіков. Вінниця: ВНАУ. 2020 322 с.

4. Jacobian method. *Byjus.* URL: https://byjus.com/maths/jacobian-method/#:~:text=The%20Jacobi%20iterative%20method%20is,in%20for%20each%20diagonal%20element (дата звернення: 20.05.2024).

5. Iterative methods Jacobi and Gauss-Seidel. *Byjus.* URL: https://byjus.com/maths/iterative-methods-gauss-seidel-and-jacobi/ (дата звернення: 20.05.2024).

УДК 004.01

Зимич А. П., *здобувач 2 курсу спеціальності 122 Комп'ютерні науки,*
науковий керівник:
Хмелівський Ю. С., *асистент кафедри інформаційних технологій*

РОЗРОБКА ГРИ ARKANOID НА МОВІ ПРОГРАММУВАННЯ JAVA З ВИКОРИСТАННЯМ БІБЛІОТЕКИ LIBGDX

Донецький національний університет імені Василя Стуса, м. Вінниця

Розробка комп'ютерних ігор – це область, яка динамічно розвивається та поєднує різні аспекти науки і мистецтва. Вона включає безліч дисциплін, як-от програмування, дизайн, музика, сценарна майстерність і управління проєктами. Ігрова індустрія має величезний потенціал для зростання і вже довела свою популярність серед людей різного віку та культур. Одним із перших кроків в історії комп'ютерних ігор була розробка гри Spacewar! 1962 року. Ця гра була створена здобувачами Массачусетського технологічного інституту та спочатку не планувалася як комерційний проєкт, проте згодом відіграла ключову роль у розвитку індустрії розваг, заклавши основу для майбутніх комп'ютерних ігор. У наш час комп'ютерні ігри є не лише формою інтерактивної розваги, а й засобом самовираження та комунікації, поєднуючи людей по всьому світу. Комп'ютерні ігри надають унікальні можливості для розвитку творчих та технічних навичок, а також вивчення складних процесів у розробці ігор.

Актуальність дослідження полягає у швидкому розвитку індустрії розробки комп'ютерних ігор, яка завдяки новітнім технологіям створює ігри з реалістичною графікою та складними механіками. Метою цієї роботи є розробка копії гри Arkanoid 1986 року за допомогою Java та бібліотеки libGDX, з акцентом на освоєння libGDX та об'єктно-орієнтоване програмування. Об'єктом дослідження в роботі є процес створення комп'ютерних ігор, а предметом – робота з бібліотекою libGDX, а також вивчення підходів до розробки ігор на основі об'єктно-орієнтованої парадигми.

Способи реалізації гри Arkanoid включають декілька основних напрямів розробки. Кожен із цих аспектів відіграє важливу роль у створенні захопливої та якісної гри. Ось деякі ключові елементи реалізації гри:

- Розробка механіки гри, що включає управління кулею та платформою, руйнування цеглинок та відстеження рахунку. Просте та зрозуміле управління дасть змогу гравцям швидко освоїтися та почати грати.
- Використання libGDX для створення та відображення графіки та відтворення звукових ефектів. Ця бібліотека дасть змогу створити привабливий візуальний стиль для гри та додати якісні звукові ефекти, створюючи атмосферу та настрій для гравців.
- Створення рівнів різної складності, які відрізнятимуться за кількістю цегли, швидкість руху кулі та інших факторів, що впливають на складність гри. Це дасть змогу гравцям вибрати рівень, що відповідає їх навичкам та досвіду.

- Інтеграція елементів взаємодії з користувачем, включно з розробкою простого та інтуїтивного керування грою, щоб гравці могли легко переміщати платформу та спрямовувати кулю у потрібному напрямку.

Для реалізації гри були використані такі програми та технології:

- Java SE Development Kit 17.
- Бібліотека libGDX версії 1.12.1.
- IntelliJ IDEA Community.
- Asprite.
- Audacity.

Вибір мови програмування для будь-якого розробника є особистим вибором, який відображає його стиль та цілі проєкту. Наш вибір Java обумовлений її універсальністю, потужністю та широким прийняттям у світі програмного забезпечення. Для цього проєкту обрано фреймворк libGDX через його спеціалізацію на 2D-іграх, популярність та розширену функціональність. Цей фреймворк базується на бібліотеці LWJGL, що дає змогу взаємодіяти з низькорівневими функціями графіки та аудіо на Java.

IntelliJ IDEA Community Edition вибрано через його розширений набір інструментів та інтуїтивний інтерфейс, який полегшує процес розробки, забезпечуючи високу продуктивність і стабільність. Графічні елементи для проєкту створені та зібрані в текстурний атлас за допомогою Asprite, графічного редактора з відкритим вихідним кодом, який відомий своєю простотою використання та потужним функціоналом.

Графічні елементи для проєкту створені та зібрані в текстурний атлас за допомогою Asprite, графічного редактора з відкритим вихідним кодом, який відомий своєю простотою використання та потужним функціоналом.

Для редагування музики використано Audacity – безкоштовний та багатоплатформовий аудіоредактор, який надає широкий спектр функцій для створення якісних звукових ефектів та музичних треків.

Реалізація ігрового прототипу є ключовим етапом у розробці програмного продукту, оскільки на цьому етапі ідеї перетворюються на робочу версію гри. Для ефективного управління ресурсами гри та забезпечення зручної структури проєкту було організовано файли так, щоб полегшити доступ до них під час розробки і майбутнього супроводу гри.

Каталог assets містить усі ресурси, необхідні для роботи гри, зокрема текстурні атласи, музичні файли і звукові ефекти, що сприяє їх централізованому зберіганню та полегшує пошук і заміну. Використання текстурних атласів сприяє оптимізації рендерингу та зменшенню завантаження файлів під час гри, що підвищує продуктивність.

Каталог core містить загальний код, включно з класами для ігрових об’єктів, обробки подій та управління станом гри. Така організація дає змогу зберігати основний код окремо від специфічних реалізацій для різних платформ, що підвищує його повторне використання та зменшує дублювання. Підкаталог screens містить класи для управління різними екранами гри, а підкаталог menus включає класи та ресурси для створення меню гри. Підкаталог levels містить класи, пов’язані з рівнями гри та їх конфігураціями.

Каталог desktop містить код, специфічний для запуску гри на настільних комп’ютерах, включно з налаштуваннями та класом DesktopLauncher, який відповідає за запуск гри в середовищі десктопу.

Клас Arkanoid.java є основним класом гри, який успадковується від базового класу Game фреймворка libGDX. Він відповідає за ініціалізацію та управління всіма аспектами ігрового процесу.

Клас Arkanoid.java відповідає за головні аспекти гри та успадковується від базового класу Game фреймворка libGDX. Він керує завантаженням ресурсів, налаштуванням екрана та переходами між екранами гри. Завантаження текстур відбувається через клас Assets, а звукові ефекти та музика управляються класом AudioManager.

Основна логіка гри реалізована через взаємодію класів Ball, Paddle та Brick, які керують поведінкою об’єктів гри. Клас GameScreen координує взаємодію цих об’єктів та відображення гри на екрані. Рівні гри відповідають за складність і структуру кожного етапу, додаючи унікальні виклики для гравців.

Музика та звукові ефекти доповнюють атмосферу гри, забезпечуючи більш реалістичне і захоплююче враження. Проєкт розроблений з використанням піксельного шрифту та збережений на GitHub для зручності зберігання версій та спільної роботи.

Розробка комп’ютерних ігор являє собою захопливий синтез науки та мистецтва, що відіграє ключову роль у сучасному світі розваг. Процес розробки гри Arkanoid за допомогою Java та бібліотеки libGDX є важливим кроком у дослідженні цього поля. Завдяки інтеграції новітніх технологій, вивченню об’єктно-орієнтованого програмування та застосуванню передових методів розробки ігор цей проєкт створює основу для подальших досягнень у цій захоплюючій галузі.

Список використаних джерел

1. Документація libGDX. URL: https://libgdx.com/wiki (дата звернення: 20.05.2024).
2. Obregon A. IntelliJ IDEA vs. Other Java IDEs – A Comparison. Apr. 28, 2023. URL: https://medium.com/@AlexanderObregon/intellij-idea-vs-other-java-ides-a-comprehensive-comparison-8866c172257e (дата звернення: 20.05.2024).
3. Asprite – Animated sprite editor & pixel art tool: вебсайт. URL: https://www.aseprite.org/ (дата звернення: 20.05.2024).
4. libGDX demo – Cuboc. URL: https://github.com/libgdx/libgdx-demo-cuboc (дата звернення: 20.05.2024).

УДК 004.2

Скороход О. М., *здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,*
науковий керівник:
Якубич К. О., *асистент кафедри інформаційних технологій*

АЛГОРИТМИ НАЙКОРОТШОГО ШЛЯХУ

Донецький національний університет імені Василя Стуса, м. Вінниця

Проблема пошуку найкоротшого шляху в графах є важливим складником теорії графів та має велике застосування в різних галузях. Це завдання полягає у пошуку найбільш оптимального маршруту або шляху з однієї вершини графу до іншої, такого, який має мінімальну суму ваг ребер. Відмінності у вагах ребер та типах графів визначають різні підходи до розв'язання цієї проблеми.

Граф – це математична структура, що складається з вершин (вузлів) та ребер (зв'язків), які їх з'єднують. Граф може бути орієнтованим, коли кожне ребро має напрямок, або неорієнтованим, коли напрямок не враховується.

Найкоротший шлях – це мінімальний за вагою шлях між двома вершинами у графі. Це поняття може мати різне значення залежно від конкретної мети: від мінімальної кількості ребер, яку необхідно пройти, до мінімальної суми ваг ребер на шляху.

Класифікація алгоритмів може бути проведена за ознаками:

1. За кількістю вершин:

– одноточковий: шукає найкоротші шляхи від однієї конкретної вершини до всіх інших у графі;

– багатоточковий: визначає найкоротші шляхи між усіма парами вершин.

2. За типом графу:

– неорієнтований: граф, де ребра не мають напрямку;

– орієнтований: граф, де ребра мають напрямок.

3. За характером ваги ребер:

– з додатними вагами: всі ваги ребер є додатними числами або нулем;

– з від'ємними вагами: деякі ребра можуть мати від'ємні ваги.

Розглянемо основні алгоритми пошуку найкоротшого шляху:

1. Алгоритм Дейкстри. Це один з найефективніших алгоритмів для пошуку найкоротших шляхів у графі з невід'ємними вагами. Він працює в часовому складі $O(V2)$O(V2), де VV – кількість вершин у графі.

2. Алгоритм Беллмана–Форда. Цей алгоритм може використовуватися в графах з від'ємними вагами. Він також здатний виявити від'ємні цикли. Однак його часова складність складає $O(VE)$O(VE), що робить його менш ефективним за алгоритм Дейкстри у багатьох випадках.

3. Алгоритм Флойда–Воршелла. Цей алгоритм використовується для знаходження найкоротших шляхів між усіма парами вершин у графі. Його часова складність становить $O(V3)$O(V3).

Розлянемо реалізацію алгоритму Дейкстри на мові програмування C#. Він використовує структуру даних – список суміжності – для представлення графу та пріоритетну чергу для ефективного вибору наступної вершини для дослідження.

```
using System;
using System.Collections.Generic;

class Graph
{
   private int Vertices;
   private List<Tuple<int, int>>[] adjacencyList;

   public Graph(int vertices)
   {
      Vertices = vertices;
      adjacencyList = new List<Tuple<int, int>>[vertices];
      for (int i = 0; i < vertices; i++)
      {
         adjacencyList[i] = new List<Tuple<int, int>>();
      }
   }

   public void AddEdge(int u, int v, int weight)
   {
      adjacencyList[u].Add(Tuple.Create(v, weight));
   }

   public void Dijkstra(int source)
   {
      var distance = new int[Vertices];
      var shortestPathTreeSet = new bool[Vertices];
      var priorityQueue = new SortedSet<Tuple<int, int>>(Comparer<Tuple<int, int>>.Create((x, y) =>
      x.Item1 == y.Item1 ? x.Item2.CompareTo(y.Item2) : x.Item1.CompareTo(y.Item1)));

      for (int i = 0; i < Vertices; i++)
      {
         distance[i] = int.MaxValue;
         shortestPathTreeSet[i] = false;
      }

      distance[source] = 0;
      priorityQueue.Add(Tuple.Create(0, source));

      while (priorityQueue.Count > 0)
      {
         var minNode = priorityQueue.Min;
         priorityQueue.Remove(minNode);

         int u = minNode.Item2;

         if (shortestPathTreeSet[u]) continue;

         shortestPathTreeSet[u] = true;
```

```
            foreach (var adjacent in adjacencyList[u])
            {
                int v = adjacent.Item1;
                int weight = adjacent.Item2;

                if (!shortestPathTreeSet[v] && distance[u] != int.MaxValue && distance[u] + weight < distance[v])
                {
                    distance[v] = distance[u] + weight;
                    priorityQueue.Add(Tuple.Create(distance[v], v));
                }
            }
        }

        Print(distance);
    }

    private void Print(int[] distance)
    {
        Console.WriteLine("Вершина \t Відстань від джерела");
        for (int i = 0; i < Vertices; i++)
        {
            Console.WriteLine("{0} \t\t {1}", i, distance[i]);
        }
    }
}

class Program
{
    static void Main()
    {
        Graph g = new Graph(9);

        g.AddEdge(0, 1, 4);
        g.AddEdge(0, 7, 8);
        g.AddEdge(1, 2, 8);
        g.AddEdge(1, 7, 11);
        g.AddEdge(2, 3, 7);
        g.AddEdge(2, 8, 2);
        g.AddEdge(2, 5, 4);
        g.AddEdge(3, 4, 9);
        g.AddEdge(3, 5, 14);
        g.AddEdge(4, 5, 10);
        g.AddEdge(5, 6, 2);
        g.AddEdge(6, 7, 1);
        g.AddEdge(6, 8, 6);
        g.AddEdge(7, 8, 7);

        g.Dijkstra(0);
    }
}
```

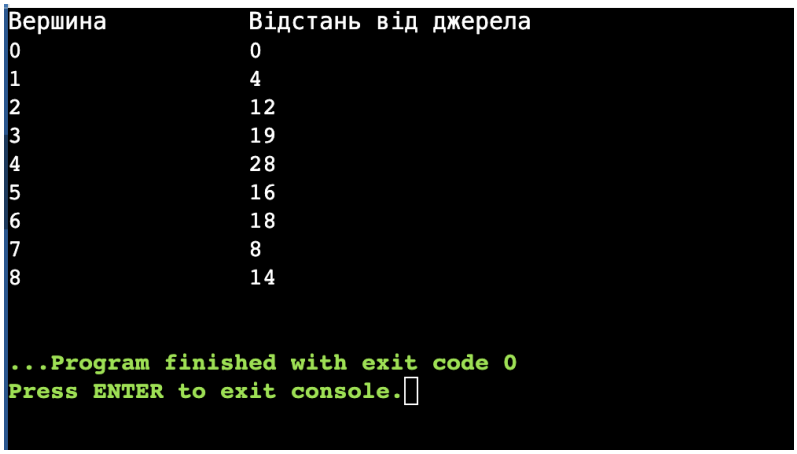


Рис. 1. Результат реалізації алгоритму Дейкстри

Отже, алгоритми найкоротшого шляху є потужними інструментами для вирішення широкого спектра задач оптимізації. Вибір конкретного алгоритму залежить від типу графу, ваг ребер та необхідної інформації (найкоротший шлях до однієї вершини чи до всіх).

Список використаних джерел

1. Ільман В. М., Іванов О. П., Панік Л. О. Алгоритми та структури даних: навчальний посібник. Дніпро. 2019. URL: https://crust.ust.edu.ua/server/api/core/bitstreams/16eada7c-c082-46f7-af81-1d6e97ac9320/content

2. Вікіпедія. Алгоритм Дейкстри. Відредаговано 4 січня 2024. URL: https://uk.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%94%D0%B5%D0%B9%D0%BA%D1%81%D1%82%D1%80%D0%B8 (дата звернення: 20.05.2024).

3. Креневич А. Алгоритми і структури даних. Підручник. Київ: ВПЦ «Київський Університет», 2021. 200 с.

УДК 004.9

Тронь Д. В., *здобувач 2 курсу спеціальності 122 Комп'ютерні науки,*
науковий керівник:
Волонтир Л. О., *старший викладач кафедри інформаційних технологій*

ЧИСЕЛЬНІ МЕТОДИ РОЗВ'ЯЗАННЯ НЕЛІНІЙНИХ РІВНЯНЬ НА C#

Донецький національний університет імені Василя Стуса, м. Вінниця

Нелінійні рівняння зустрічаються у багатьох наукових та інженерних задачах. Проблема розв'язання нелінійних рівнянь полягає в тому, що багато з них не мають аналітичних розв'язків або їх знаходження є дуже складним. Це вимагає застосування чисельних методів для знаходження наближених розв'язків. Різні методи обчислень дають змогу отримувати точні та швидкі результати, роблячи їх незамінними в сучасних обчисленнях. Серед найбільш популярних методів є метод Ньютона та метод хорд, які широко застосовуються в різних галузях. Важ-

ливим аспектом використання чисельних методів є програмування, яке дає змогу автоматизувати обчислення та обробляти великі обсяги даних. Мова програмування C# є потужним інструментом для реалізації таких методів завдяки своїй ефективності та зручності.

Головна мета цієї роботи – показати, як можна реалізувати метод Ньютона та метод хорд для розв'язання нелінійних рівнянь за допомогою мови програмування C#.

Аналітичні методи для розв'язання нелінійних рівнянь часто не можуть бути застосовані через складність або неможливість отримання точного розв'язку. У таких випадках використовуються чисельні методи, які дають змогу знаходити наближені розв'язки з високою точністю.

Метод Ньютона є ефективним способом знаходження коренів нелінійних рівнянь. Ось приклад його **реалізації на мові програмування C#**:

```
using System;
public class NewtonMethod
{
   // Функція, корінь якої шукаємо
   static double Function(double x)
   {
      return x * x - 2;
   }
   // Похідна функції
   static double Derivative(double x)
   {
      return 2 * x;
   }
   public static void Main()
   {
      double x0 = 1.0; // Початкове наближення
      double epsilon = 1e-6; // Точність
      double x1;
      do
      {
         x1 = x0 - Function(x0) / Derivative(x0);
         x0 = x1;
      }
      while (Math.Abs(Function(x1)) > epsilon);
      Console.WriteLine("Корінь: " + x1);
   }
}
```

У цьому коді функція 'Function' визначає нелінійну функцію $f(x) = x^2 – 2$, а 'Derivative' – її похідну $f'(x) = 2x$. Метод ітераційно оновлює значення 'x0', поки функція не стане достатньо близькою до нуля.

Реалізація методу хорд на C#. Метод хорд не вимагає обчислення похідної функції, тому він часто використовується у випадках, коли похідну важко знайти. Ось приклад реалізації методу хорд на C#:

```
using System;
public class SecantMethod
{
   // Функція, корінь якої шукаємо
   static double Function(double x)
   {
      return x * x - 2;
   }
   public static void Main()
   {
      double x0 = 1.0; // Перше початкове наближення
      double x1 = 2.0; // Друге початкове наближення
      double epsilon = 1e-6; // Точність
      double x2;
      do
      {
         x2 = x1 - Function(x1) * (x1 - x0) / (Function(x1) - Function(x0));
         x0 = x1;
         x1 = x2;
      }
      while (Math.Abs(Function(x2)) > epsilon);
      Console.WriteLine("Корінь: " + x2);
   }
}
```

У цьому коді функція 'Function' визначає нелінійну функцію f(x) = x^2 – 2 . Метод використовує два початкові наближення 'x0' та 'x1' і ітераційно оновлює їх, поки функція не стане достатньо близькою до нуля.

Обидва методи дають змогу ефективно знаходити корені нелінійних рівнянь. Метод Ньютона зазвичай швидше збігається, але потребує обчислення похідної, тоді як метод хорд не потребує похідної, але може вимагати більше ітерацій для досягнення заданої точності.

Розглянемо розв’язання простого нелінійного рівняння методом Ньютона: x^2 – 2 = 0. Результати обчислення такі: після декількох ітерацій корінь наближається до $\sqrt{2}$ ≈1.414213.

Аналіз результату: метод Ньютона швидко збігається до точного розв’язку за умови, що початкове наближення x0 близьке до кореня. У цьому випадку алгоритм досягає необхідної точності всього за кілька ітерацій.

Розглянемо розв’язання простого нелінійного рівняння методом хорд: x^2 – 2 = 0. Результати обчислення: після декількох ітерацій корінь наближається до $\sqrt{2}$ ≈1.414213.

Аналіз результату: метод хорд, хоча і потребує більше ітерацій для досягнення необхідної точності, порівняно з методом Ньютона, є корисним у випадках, коли похідну функції обчислити складно або неможливо. Він не потребує знання похідної, що робить його більш універсальним.

Отже, програмна реалізація методів показала, що мова програмування C# виявилася зручним і потужним інструментом для реалізації методів обчислень. Її синтаксис та можливості дають змогу ефективно створювати алгоритми й обробляти дані. Дослідження в галузі чисельних методів на C# може бути продовжене

у багатьох напрямах. Наприклад, можна досліджувати ефективність та збіжність різних модифікацій методів Ньютона та хорд, розробляти адаптивні методи зміни кроку ітерацій для підвищення швидкості збіжності, або використовувати ці методи для розв'язання складних систем рівнянь.

Список використаних джерел

1. Чисельні методи: навчальний посібник / Л. О. Волонтир, О. В. Зелінська, Н. А. Потапова, І. А. Чіков. Вінниця: ВНАУ. 2020. 322 с.
2. Задачин В. М., Конюшенко І. Г. Чисельні методи: навчальний посібник. Харків: Вид. ХНЕУ ім. С. Кузнеця, 2014. 180 с.
3. Гончаров О. А., Васильєва Л. В., Юнда А. М. Чисельні методи розв'язання прикладних задач: навч. посіб. Суми: Сумський державний університет, 2020. 142 с.

УДК 004.6

Юрчук Д. М., *здобувач 2 курсу спеціальності 122 Комп'ютерні науки,*
науковий керівник:
Волонтир Л. О., *старший викладач кафедри інформаційних технологій*

МЕТОД МОНТЕ-КАРЛО В МОДЕЛЮВАННІ ОЦІНКИ ФІНАНСОВИХ АКТИВІВ

Донецький національний університет імені Василя Стуса, м. Вінниця

Метод Монте-Карло – це клас обчислювальних алгоритмів, які використовують випадкові числа для розв'язання математичних і фізичних задач, що можуть бути надто складними або неможливими для аналітичного розв'язання. Основна ідея методу полягає в моделюванні випадкових процесів і статистичному аналізі їх результатів для отримання наближених розв'язків.

Метод Монте-Карло був винайдений Джоном фон Нейманом і Станіславом Уламом під час Другої світової війни для покращення прийняття рішень у невизначених умовах.

Метод Монте-Карло використовується для чисельного інтегрування, розв'язання диференціальних рівнянь, оптимізації, а також для моделювання систем із великою кількістю змінних, де традиційні детерміністичні методи неефективні. В основі методу лежить генерація випадкових або псевдовипадкових чисел, які використовуються для створення статистичної вибірки. Ця вибірка дає змогу оцінювати характеристики системи або процесу, що моделюється.

Підходи моделювання на основі методу Монте-Карло використовують у своїй основі єдиний шаблон:

1. Визначити область можливих вхідних даних.
2. Випадковим способом згенерувати вхідні дані із визначеної вище області за допомогою деякого заданого розподілу ймовірностей.
3. Виконати детерміновані обчислення над вхідними даними.
4. Проміжні результати окремих розрахунків звести у кінцевий результат.

У випадках невизначеності прогнозної оцінки за методом Монте-Карло середня очікувана величина є результатом усереднення значень кількох проведених експериментів. Моделювання за методом Монте-Карло має широкий спектр застосувань у галузях, які оцінюються випадковими змінними, зокрема в бізнесі та інвестиціях. Вони використовуються для оцінки ймовірності перевищення витрат у великих проєктах і ймовірності того, що ціна активу змінюватиметься за випадковим законом.

Телекомунікаційні компанії використовують метод для оцінки продуктивності мережі в різних сценаріях, що допомагає їм оптимізувати власні мережі. Фінансові аналітики використовують моделювання за методом Монте-Карло, щоб оцінити ризик дефолту підприємства та проаналізувати похідні інструменти, як-от опціони. Страховики та бурильники нафтових свердловин також використовують їх для вимірювання ризику.

На відміну від звичайної моделі прогнозування, моделювання Монте-Карло передбачає набір результатів на основі оціненого діапазону значень у порівнянні з набором фіксованих вхідних значень. Тобто симуляція Монте-Карло будує модель можливих результатів за допомогою розподілу ймовірностей, як-от рівномірний або нормальний розподіл, для будь-якої змінної, що має невизначеність. Потім відбувається перерахування результатів, щоразу з використанням іншого набору випадкових чисел між мінімальним і максимальним значеннями. У типовому експерименті Монте-Карло цю вправу можна повторювати тисячі разів, щоб отримати велику кількість ймовірних результатів.

Завдяки своїй точності моделювання Монте-Карло також використовується для довгострокових прогнозів. Зі збільшенням кількості вхідних даних зростає й кількість прогнозів, що дає змогу точніше прогнозувати результати в більш віддаленому часі. Коли моделювання Монте-Карло завершено, воно дає ряд можливих результатів з імовірністю кожного результату.

Розглянемо моделювання за методом Монте-Карло в фінансовій оцінці активів. Процес складається з таких етапів:

Етап 1. Щоб спроєктувати одну можливу цінову траєкторію, використовують історичні дані про ціну активу, щоб створити серію періодичних щоденних прибутків за допомогою натурального логарифма:

$$\text{Періодичне щоденне повернення} = l_n(\frac{\text{Ціна дня}}{\text{ціна попереднього дня}}).$$

Етап 2. Далі використовують функції *AVERAGE*, *STDEV.P* і *VAR.P* для всього результуючого ряду, щоб отримати вхідні дані середнього добового доходу, стандартного відхилення та дисперсії відповідно. Дрейф даних дорівнює:

$$\text{Дрейф} = \text{Середній добовий дохід} - \frac{\text{Дисперсія}}{2},$$

де середній добовий дохід = Виготовляється з Excel.

Функція *AVERAGE* із ряду періодичних щоденних прибутків.
Дисперсія = Виготовляється з Excel.

Функція *VAR.P* із ряду періодичних щоденних прибутків.

До того ж дрейф може бути встановлений на 0. Цей вибір відображає певну теоретичну орієнтацію, але різниця не буде великою, принаймні для коротших часових меж.

Етап 3. Далі отримують випадковий вхід:

$$\text{Випадкове значення} = \sigma \times NORMSINV(RAND()),$$

де σ = Стандартне відхилення, отримане з Excel.

Функція *STDEV.P* із періодичних щоденних показників.
NORMSINV і *RAND* = Функції Excel.

Рівняння для ціни наступного дня таке:

$$\text{Ціна наступного дня} = \text{Сьогоднішня ціна} \times \sigma^{(\text{Дрейф}+\text{Випадкове значення})}.$$

Етап 4. Щоб привести e до заданого степеня x у Excel, ми можемо скористатись функцією EXP: EXP(x). Повторити цей розрахунок потрібно необхідну кількість разів (кожне повторення означає один день).

Результатом є імітація майбутнього руху ціни активу.

Моделювання Монте-Карло показує спектр ймовірних результатів для невизначеного сценарію. Ця техніка присвоює кілька значень невизначеним змінним, отримує кілька результатів, а потім бере середнє значення цих результатів, щоб отримати оцінку.

Від інвестування до проєктування метод Монте-Карло використовується в багатьох програмах для вимірювання ризику, зокрема для оцінки ймовірності прибутку чи збитку від інвестицій або ймовірності того, що проєкт перевищить бюджет.

Список використаних джерел

1. Сайт IBM. URL: https://www.ibm.com/topics/monte-carlo-simulation (дата звернення: 16.05.2024).

2. Сайт вервісу TECHTARGET. URL: https://www.techtarget.com/searchcloudcomputing/definition/Monte-Carlo-simulation (дата звернення: 16.05.2024).

3. Monte carlo simulation: what it is, history, how it works, and 4 key steps / by W. Kenton. 27.06.2024. URL: https://www.investopedia.com/terms/m/montecarlosimulation.asp (дата звернення: 16.05.2024).

СЕКЦІЯ 3
ТЕХНОЛОГІЇ ЗБОРУ, ПРЕДСТАВЛЕННЯ ОБРОБКИ, ЗБЕРІГАННЯ ІНФОРМАЦІЇ В СУЧАСНИХ ІНФОРМАЦІЙНИХ ТА КОМП'ЮТЕРНИХ СИСТЕМАХ

УДК 004.738.1

Підруцький Д. А., *здобувач 3 курсу спеціальності 122 Комп'ютерні науки,*
Хмелівський Ю. С., *асистент кафедри інформаційних технологій*

РОЗРОБКА ПОГОДНОГО ВЕБДОДАТКА З ВИКОРИСТАННЯМ VISUAL CROSSING WEATHER API

Донецький національний університет імені Василя Стуса, м. Вінниця

Сьогодні інформація відіграє ключову роль у різних аспектах нашого життя, тому надати зручний доступ до даних, які змінюються, може бути проблематично. Розробка погодного вебдодатка може бути надзвичайно корисною для сучасного розробника. Використання API для отримання погодних даних, зокрема Visual Crossing Weather API, надає не тільки можливість створювати додатки, які забезпечують точну та актуальну інформацію, а й досвід у застосуванні API у своїх проєктах.

Погодна тематика є доволі актуальною, оскільки сама погода постійно змінюється, а актуальна інформація про її зміни є важливою як для індивідуальних користувачів, так і для різних галузей економіки та науки, і вимагає постійного вдосконалення методів її прогнозування та аналізу.

Останні дослідження в галузі розробки погодних додатків підкреслюють важливість надання точної та швидкої інформації, особливо з огляду на погоду, яка постійно змінюється. Тому проєкти, які адаптуються до цих змін через використання API, є доволі затребуваними.

Метою є опис створення інтуїтивно зрозумілого, компактного, а отже, – простого і функціонального проєкту, який надає користувачам актуальну та точну інформацію про погоду з використанням API.

Відповідно до мети, ми отримаємо таку постановку задач:

1. Використання відповідних баз даних для зберігання, обробки та використання інформації про погоду.
2. Створення панелі погоди для сьогодення з інформацією про обране місто, опади, температуру, швидкість вітру, вологість та показник того, як відчувається температура.
3. Розробка панелі прогнозування на наступні дні з окремими блоками для кожного дня тижня, що містять інформацію про опади та температуру.
4. Реалізація функції пошуку для знаходження інформації про погоду в обраних містах із можливістю автоматичного переходу до обраного місця.
5. Виведення спливаючого повідомлення, якщо обране місце не знайдене в результаті пошуку.
6. Створення декоративної функції для задавання фонів залежно від погоди, що відображається, для кожного міста.
7. Врахування значень опадів у декоративній функції для підбору відповідного зображення.

8. Розробка інтуїтивного та ефективного інтерфейсу для забезпечення зручної взаємодії користувачів з інформацією про погоду та налаштуванням уподобань.

Під час розробки застосовано такі інструменти:

1. JavaScript – це мова програмування, яка широко використовується для створення динамічних, інтерактивних вебсайтів та вебдодатків. Це мова високого рівня, а отже, що вона розроблена так, щоб її було легко читати і писати, і часто використовується разом з HTML та CSS для додавання інтерактивності та функціональності вебсторінкам [1].

2. React – це JavaScript-бібліотека для створення інтерфейсів користувача. Це популярний інструмент для створення вебдодатків з високою динамічністю, що дає змогу швидко та ефективно створювати складні вебдодатки [2].

3. Vite – сучасний інструмент для розробки вебдодатків. Він дає змогу розробникам використовувати JavaScript для швидкого створення, запуску та оновлення вебдодатків у реальному часі. Vite використовує модульний підхід до імпортування файлів, що забезпечує неблокуючу та ефективну обробку запитів [3].

4. Npm – інструмент для управління пакетами у середовищі Node.js. Він дає змогу розробникам швидко встановлювати, оновлювати та видаляти пакети з їхніх проєктів, а також виконувати різноманітні інші дії, як-от запуск скриптів, тестування та збірка проєктів [4].

Також буде застосована інформація про погоду з Visual Crossing Weather API для того, щоб відображалась достовірна інформація, яка зі зміною погоди також буде змінюватись у різних містах.

Результатом використання інструментів та виконання задач буде такий вигляд вебдодатка (рис. 1):

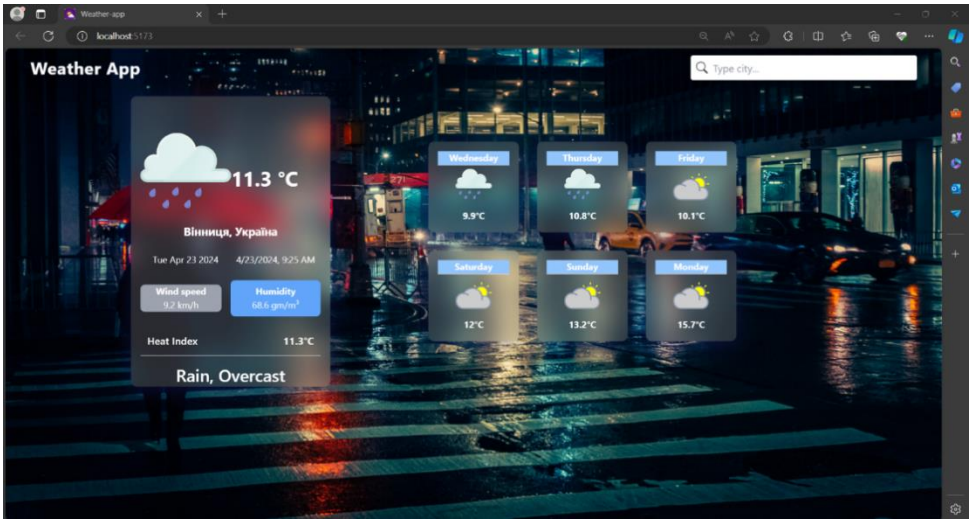


Рис. 1. Головна сторінка вебдодатка

Згідно з поставленими задачами, вдалося створити простий та зручний інтерфейс, додати пошукову функцію, панелі з інформацією та фон, який змінюється відповідно до погоди.

У підсумку, вдалося виконати поставлені задачі, а також описати розробку погодного вебдодатка, показати те, що використання API дає змогу легко і швидко отримувати доступ до погодних даних та інтегрувати їх у додаток зручним для користувачів способом.

Список використаних джерел

1. JavaScript Підручник. Основи вебпрограмування. URL: https://w3schoolsua.github.io/js/index.html#gsc.tab=0 (дата звернення: 11.05.2024).
2. Meta Platforms. Посібник: знайомство з React. 2024. URL: https://uk.legacy.reactjs.org/tutorial/tutorial.html (дата звернення: 11.05.2024).
3. Використання Vite для швидкого розгортання веб-додатків на основі Vue та React. *IT Рейтинг України.* 20.04.2024. URL: https://cutt.ly/AeeT3lDb (дата звернення: 11.05.2024).
1. Що таке npm (Node Package Manager). Як встановити та розмістити пакети / А. Денисенко (автор, розробник і перекладач). *mc.today.* 18.04.2023. URL: https://highload.today/uk/npm/ (дата звернення: 11.05.2024).

УДК 004.422.63

Козачок А. О., *здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,*
Вєтров О. С., *старший викладач кафедри інформаційних технологій*

ВИКОРИСТАННЯ ХЕШ-ТАБЛИЦЬ У СТРУКТУРАХ ДАНИХ

Донецький національний університет імені Василя Стуса, м. Вінниця

Хеш-таблиці відіграють важливу роль у сучасних структурах даних, забезпечуючи швидкий і ефективний доступ до інформації. Це потужний інструмент, який використовується в багатьох областях інформатики, включно з базами даних, комп'ютерними мережами, компіляторами, а також системами управління контентом і іншими програмами, де потрібен швидкий доступ до даних. Завдяки своїй здатності виконувати операції пошуку, вставки та видалення за постійний час хеш-таблиці часто є ключовим складником сучасних алгоритмів та систем [1].

Основа хеш-таблиці – це хеш-функція, яка перетворює ключ у індекс, за яким зберігається відповідне значення. Хеш-функція має бути достатньо ефективною, щоб уникнути колізій, а також швидкою, щоб не уповільнювати операції. Хеш-таблиця зазвичай представляється у вигляді масиву чи списку, де індекси визначаються хеш-функцією.

Однією з основних проблем, пов'язаних із хеш-таблицями, є колізії –випадки, коли два різні ключі мають однаковий хеш-індекс. Для вирішення цієї проблеми використовуються різні методи. Метод ланцюжків передбачає, що в кожному індексі може зберігатися список (ланцюжок) елементів, які мають однаковий індекс. Інший підхід – відкрита адресація, де в разі колізії хеш-таблиця шукає інше місце для зберігання, застосовуючи різні стратегії, як-от лінійне або квадратичне пробування. Подвійне хешування використовує другу хеш-функцію, щоб зменшити ймовірність колізій [2].

Переваги хеш-таблиць у швидкості та ефективності. Операції пошуку, вставки та видалення відбуваються за дуже короткий час, оскільки не вимагають лінійного пошуку. До того ж хеш-таблиці є гнучкими і можуть використовуватися для зберігання різноманітних типів даних, від простих чисел до складних об'єктів [3].

Однак хеш-таблиці також мають свої недоліки. Колізії можуть уповільнити операції і потребують додаткових механізмів вирішення. Хеш-таблиці можуть споживати багато пам’яті, особливо коли прагнуть мінімізувати колізії. Також у хеш-таблицях неможливо зберігати порядок елементів, що може бути критичним у деяких застосуваннях.

Хеш-таблиці широко використовуються в різних галузях. У базах даних вони слугують для індексації та швидкого пошуку. У комп’ютерних мережах вони використовуються для кешування та маршрутизації. У мовах програмування вони є основою для реалізації словників, карт і асоціативних масивів.

Отже, хеш-таблиці є важливим інструментом у структурі даних. Їх використання дає змогу забезпечити високу ефективність, особливо під час роботи з великими обсягами інформації, де швидкість доступу є критичною. Водночас необхідно враховувати можливі недоліки і застосовувати відповідні стратегії для вирішення проблем, пов’язаних із колізіями та використанням пам’яті.

Список використаних джерел

1. Михалько В. Г. Адаптивна індексна структура бази даних для точкового пошуку за допомогою підходів машинного навчання: магістерська дисертація. *Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського».* 2018. URL: https://ela.kpi.ua/server/api/core/bitstreams/af870de7-5956-4265-9b4b-0b91d588e666/content

2. Hash Table Data Structure. *geeksforgeeks.org.* URL: https://www.geeksforgeeks.org/hash-table-data-structure/ (дата звернення: 20.05.2024).

3. Розподілені хеш-таблиці. *Medium.* Oct. 30, 2020. URL: http://surl.li/suowl (дата звернення: 20.05.24).

УДК 004.06

Яценко В. В., *здобувач 2 курсу спеціальності 122 Комп’ютерні науки,*
Вєтров О. С., *старший викладач кафедри інформаційних технологій*

ОГЛЯД АЛГОРИТМІВ ЧИТАННЯ ТА ЗАПИСУ В РЕЛЯЦІЙНИХ БАЗАХ ДАНИХ ТА СПОСОБИ ЇХ ОПТИМІЗАЦІЇ

Донецький національний університет імені Василя Стуса, м. Вінниця

Методи аналізу і обробки певних даних, упорядкованих і неупорядкованих, для добування знань називається наукою про дані [1]. Упорядкована інформація, об’єднана між собою за певними критеріями, називається базою даних [2]. Правила представлення, організації даних і їх зв’язків називаються моделлю даних [3]. Існують реляційна, об’єктно-орієнтована, графова, мережева, ієрархічна моделі даних. Найпопулярніша модель – реляційна – представляється у вигляді двовимірних таблиць. Для роботи з реляційними базами даних використовується SQL [4]. SQL – це мова структурованих запитів, що дає змогу користувачу взаємодіяти з даними у реляційних базах даних.

Ресурси knowledgehut, gigaspaces досліджують метод аналізу даних у реальному часу [5–6]. Згідно з їх дослідженням, сучасні компанії все частіше викорис-

товують метод аналізу даних у реальному часі, оскільки зі звичайними методами збору і аналізу інформації організації не зможуть швидко реагувати на зміни і відповідно миттєво приймати рішення, залежно від ситуації. Швидка обробка даних критично важлива, оскільки дає змогу виявити загрози, проблеми та недоліки у різних галузях на початковому етапі і швидко зреагувати на подібні події. Відтак швидкість читання і запису даних критично важливі для швидкого опрацювання інформації та реагування на неї.

Метою дослідження стане аналіз алгоритмів читання і запису та методики їх оптимізації. Необхідно дослідити, як у реляційних базах даних відбувається читання і запис даних, дослідити роботу запитів, надати можливі методи підвищення продуктивності цих алгоритмів і підсумувати інформацію.

Розглянемо, як виконуються запити у базах даних. Для дії над даними необхідно здійснити запит мовою SQL. Процес здійснення запиту можна поділити на декілька етапів. На першому етапі здійснюється розбір тексту запиту. Відбувається синтаксичний аналіз і перевірка запиту на помилки. Цей процес називається парсингом запиту. Якщо на цьому етапі буде знайдена синтаксична помилка – обробка запиту припиниться і виведеться помилка. Якщо етап синтаксичного аналізу буде пройдено вдало, він буде розділений на частини і буде створено дерево послідовності виконання запиту. Дерево являє собою кроки, які необхідно здійснити для виконання запиту. На наступному етапі дерево парсингу передається до алгебризатора. Алгебризатор перевіряє назви таблиць, атрибутів, полів, вказаних у запиті. Перевіряються і розпізнаються всі посилання, вказані у запиті. Якщо хоча б одне з посилань – некоректне, робота над запитом буде зупинена і виведеться відповідна помилка. Процес, що проводиться алгебризатором, називається прив’язкою запиту. Згодом, якщо всі посилання пройшли перевірку, відбувається перевірка типів даних посилань і визначаються агрегатори – відбувається прив’язка агрегаторів. Фактично на цьому етапі відбувається перевірка, чи дійсно імена, вказані у запиті, існують у вказаних таблицях, чи правильно вказані назви полів. Якщо всі перевірки були успішно пройдені, створюється дерево процесора запитів. Це двійковий файл, що містить дані про запит у вигляді кодованого значення – хеш-значення. За допомогою хеш-значення визначається, чи має цей запит актуальний план виконання. План вважається доцільним, якщо не було здійснено змін у таблицю. Якщо у запиту існує дійсний план виконання, процес обробки запиту припиняється і використовується план виконання запиту для оптимального виконання операцій, заданих користувачем. Якщо ж такого плану не існує, виконання запиту переходить на наступний етап. Відбувається планування виконання запиту – пошуку найоптимальнішого виконання цього запиту. Оптимізатор запитів здійснює пошук і оцінку всіх шляхів виконання запиту. Оцінюється швидкість виконання і ресурсозатратність кожного методу виконання запиту. Залежно від складності запит може пройти повну оптимізацію і отримати план на основі повної переоцінки всіх можливих шляхів виконання та може отримати тривіальний план. Тривіальні плани отримують запити, настільки легкі у виконанні, що запуск повної оптимізації не є раціональним рішенням.

Зчитування даних із бази даних здійснюється за допомогою запиту SELECT FROM. За допомогою цього запиту можна вибирати певні поля. Для вибірки пев-

ного поля з певної таблиці необхідно вказати назву поля і назву таблиці, якій це поле належить, наприклад, SELECT name FROM employee, де name – це назва поля, дані з якого потрібно зчитати, employee – назва таблиці, де знаходиться це поле. Також можна обрати всі поля з певної таблиці, використовуючи синтаксис SELECT*. Дані вибірки можуть бути відсортовані за допомогою ORDER BY. Також можна обмежити вибірку за допомогою LIMIT.

Для запису даних у базу даних використовується запит INSERT INTO VALUES. За допомогою цього запиту можна записувати значення в певні або в усі стовпці. За допомогою запиту UPDATE SET WHERE можна оновити дані у певних таблицях.

Першим методом оптимізації стане нормалізація бази даних. Нормалізація бази даних дасть змогу зменшити надмірність даних, ймовірність появи аномалій, пришвидшити запис даних у базу даних. До того ж нормалізовані бази даних мають більш ефективні індекси, що також позитивно вплине на продуктивність читання і запису. Але з іншого боку, це збільшить кількість з'єднань, через які запиту потрібно пройти для пошуку необхідної інформації, що відповідно сповільнить процес читання даних таблиць.

Другий метод оптимізації читання і запису – оптимізація запитів. По-перше, використання оператора OR не є раціональним. SQL не може обробити оператор в одній операції. У разі використання оператора UNION і задання подвійної умови в якості двох умов SQL зможе використати індекси, і запит відбудеться швидше й ефективніше. По-друге, часте використання JOIN і підзапитів у запиті сповільнює процес аналізу запиту. SQL не зможе якісно побудувати план виконання і оптимізувати запит. До того ж не гарантується, що побудований план буде максимально ефективний. Розділення запиту на декілька менших запитів зменшить кількість JOIN і підзапитів, і так прискорить аналіз та виконання. По-третє, варто уникати SELECT DISTINCT. Такі запити відбуваються значно повільніше. До того ж використання SELECT* у всіх запитах не є раціональним рішенням. Потрібно вказувати лише потрібні поля, і якщо це можливо, виставляти ліміт вибірки. Це значно прискорить швидкість роботи запиту і значно зменшить використання ресурсів ним.

Третім методом оптимізації стане індексація даних. Індексування значно прискорює пошук інформації у базі даних. Існує декілька видів індексів: кластеризовані і некластеризовані індекси. Кластеризовані індекси – це індекси, які сортують дані в таблиці за ключовим значенням. Кластеризований індекс визначає порядок розміщення даних у таблиці. На кожну таблицю може бути створений лише один кластеризований індекс. Кластеризовані індекси значно пришвидшують послідовне читання інформації. Такі індекси також значно прискорять вибірки великої кількості полів. До того ж більша продуктивність буде помітна і під час роботи з діапазонами. Однак кластеризований індекс ускладнює запис даних у таблиці, особливо якщо відбувається багато непослідовних вставок. У такому разі продуктивність запису даних буде знижена. Некластеризований індекс – індекс, який зберігає посилання на певні дані, і, на відміну від кластеризованого, не впорядковує їх фізично. Некластеризований індекс дає змогу швидко отримати доступ до певного поля, але якщо буде вибірка з багатьох полів, то цей метод

індексації не буде раціональним у використанні. Перевагою некластеризованого індексу також є пришвидшення роботи INSERT- і UPDATE-запитів, оскільки для здійснення таких операцій не потрібне сортування даних. Використання індексів дійсно пришвидшує роботу з даними, але занадто велика кількість індексів може сповільнити швидкість запису даних і зайняти багато дискового простору.

Четвертий метод оптимізації – кешування. Це процес зберігання даних, що часто використовуються в оперативній пам'яті або на жорсткому диску користувача. Існує декілька методик кешування: Cache-Aside, Read-through Cache, Write-back Cache, Write-through Cache. Cache-aside працює так, що якщо даних немає в пам'яті, відбувається звернення до бази даних. Кеш оновлюється новими даними, і тоді повертається результат запиту. Перевагою цієї методики є вирішення проблеми частих зчитувань. До того ж дані зберігаються в кеші, лише якщо вони потрібні, що заощаджує пам'ять системи. Read-through Cache працює аналогічно, за винятком того, що користувач завжди звертається до кешу. Переваги і недоліки аналогічні першому методу. Write-back Cache – дані постійно записуються в кеш, але не зберігаються в базі даних доти, доки не відбудеться запит на збереження внесених змін. Отже, в пам'яті завжди буде актуальна інформація, а запис даних у базу даних буде значно ефективніший. Недоліком способу є можлива втрата даних у разі зупинки системи. Write-through Cache записує дані в кеш і одразу переносить їх у базу даних. Перевагою методу є те, що у пам'яті завжди буде актуальна інформація, але необхідно робити два записи одночасно – в кеш і в базу даних, що може бути ресурсозатратно. Отже, методики кешування пришвидшують доступ до даних, що регулярно використовуються, шляхом зменшення кількості звернень до бази даних. Це відповідно приводить до зменшення навантаження на базу даних і значного скорочення часу обробки запиту. Вибір методики кешування залежить від потреб користувача.

П'ятий метод оптимізації – використання масових операцій, або об'єднання малих запитів у один. Методика масових операцій – виконання багатьох однотипних операцій одночасно. Це дає змогу значно прискорити обробку даних. Також можна власноруч об'єднати запити в один, що значно пришвидшить запис і оновлення даних у таблиці.

Підсумовуючи дослідження, зазначимо, що швидка обробка інформації критично важлива. Велика швидкість аналізу інформації дає змогу ефективно виявляти проблеми на перших етапах і відповідно негайно реагувати та приймати миттєві рішення. Особливо важливою стає швидкість обробки запитів читання і запису в базу даних, оскільки від цього буде залежати вчасність прийняття рішення. SQL володіє вбудованими методами оптимізації запитів і виконує їх за найбільш ефективним планом, але методи оптимізації все ж відіграють важливу роль у продуктивності системи. Ефективна структура бази даних, оптимізовані запити, правильно налаштовані індекси, а також використання різних методик кешування і масових операцій можуть значно збільшити швидкість і ефективність системи.

Список використаних джерел

1. Avijeet B. What is data science. *SimpliLearn.* 2024. URL: https://www.simplilearn.com/tutorials/data-science-tutorial/what-is-data-science (дата звернення: 15.05.2024).

2. What is a database. *amazon.com.* URL: https://aws.amazon.com/what-is/database/ (дата звернення: 15.05.2024).
3. What is data model. *Center for Data, Analytics and Reporting (CeDAR).* URL: https://cedar.princeton.edu/understanding-data/what-data-model (дата звернення: 15.05.2024).
4. What is SQL. *geeksforgeeks.org.* 2024. URL: https://www.geeksforgeeks.org/what-is-sql/ (дата звернення: 15.05.2024).
5. Gulati A. Real-time data analytics. *upGrad. Knowledgehut.* 24 Apr, 2024. URL: https://www.knowledgehut.com/blog/data-science/real-time-data-analytics (дата звернення: 15.05.2024).
6. Real-Time Data Processing. *Gigaspaces.* URL: https://www.gigaspaces.com/data-terms/real-time-data-processing (дата звернення: 15.05.2024).

УДК 004.9

Грибаніна А. О., *здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,*
Поремський Ю. В., *канд. техн. наук, старший викладач кафедри інформаційних технологій*

ОСОБЛИВОСТІ ОРГАНІЗАЦІЇ ЧЕРГИ

Донецький національний університет імені Василя Стуса, м. Вінниця

Черга – це змінюваний упорядкований (чи ні) набір елементів. Додавання елементу в чергу проводиться з одного кінця (хвоста черги, REAR), а вибірка – з іншого кінця (голови черги, FRONT) відповідно до правила «Першим прийшов – першим пішов» (FIFO: First Input – First Output). Така черга є простою чергою без пріоритетів (рис. 1). Часто використовуються черги з пріоритетами, в них більш пріоритетні елементи включаються ближче до голови черги, вибірка здійснюється зазвичай з голови черги.

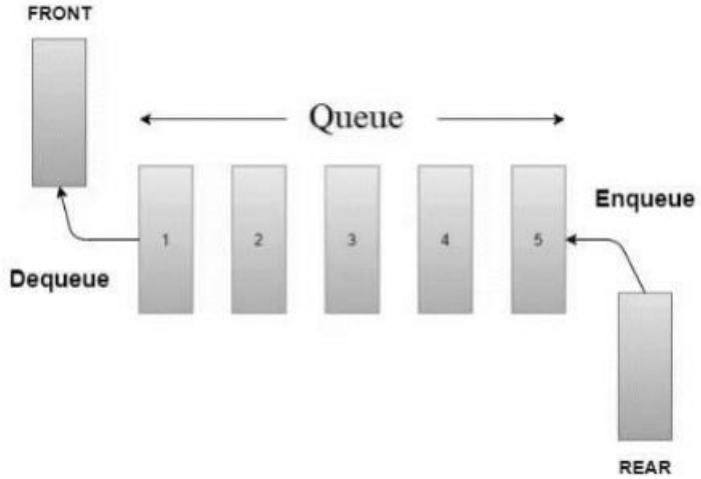


Рис. 1. Схема простої черги

Основні операції над чергою – ті ж, що і над стеком – включення, виключення, визначення розміру. Класична реалізація черги вимагає реалізувати структуру даних із такими операціями:

1. Створення порожньої черги.

2. Операція empty() – визначення, чи є черга порожньою. Повертає булеве значення.

3. Операція enqueue(item) – додати елемент item до кінця черги.

4. Операція dequeue() – взяти елемент з початку черги.

Як і у випадку зі стеком, крім зазначених вище операцій, опціонально визначають інші операції: перегляд елементу в голові та хвості черги без їх видалення з черги, розмір черги, тобто кількість елементів у ній, тощо.

Реалізація черги у вигляді масиву ефективна тоді, коли максимальний розмір черги маленький або відомий заздалегідь. Але якщо розмір масиву під чергу не може бути визначено заздалегідь (а таких випадків більшість), іншої альтернативи, ніж використання представлення черги у вигляді зв'язного списку, не існує.

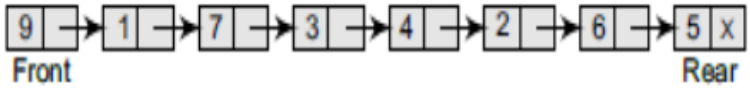


Рис. 2. Реалізація простої зв'язної черги

Основними операціями над чергою є:

1) створення і видалення черги;

2) включення в чергу нового елементу;

3) вибірка елементу з черги.

Під час їх виконання використовуються допоміжні операції, як-от перевірка наявності елементів у черзі і перевірка переповнення черги.

Як і у випадку зі стеком, створення черги зводиться до побудови тільки дескриптора черги. У ньому наявні покажчик на голову черги front, покажчик на хвіст черги rear і поточний розмір черги size. Покажчики front і rear встановлюється в NULL, поле size встановлюється в нуль, позначаючи, що черга порожня.

Зводиться до звільнення динамічної пам'яті, отриманої в процесі створення і обробки черги. Перед видаленням дескриптора черги треба видалити всі елементи черги по черзі, з голови черги до її хвоста. В іншому випадку будуть виникати витоки пам'яті.

У разі черги спискової структури переповнення зазвичай не буває, однак перевірка необхідна. Включення в чергу зводиться до отримання динамічної пам'яті під новий елемент, занесення в нього даних і додавання елементу у хвіст черги. Очевидно, що в покажчик наступного елементу черги в старому хвості заноситься значення покажчика на новостворений елемент (новий хвіст черги).

Операція можлива тільки тоді, коли черга не порожня. Вона означає, що вибирається значення з елементу, який знаходиться в голові черги, а сам елемент виключається з черги. Покажчик на голову черги в дескрипторі замінюється на адресу наступного елементу, а пам'ять під колишню голову черги звільняється.

Чергу з пріоритетом можна розглядати як модифіковану чергу, в якій для обслуговування вибирається перший елемент із найвищим пріоритетом із головної частини черги. Сам пріоритет елементу може бути встановлений на основі різних факторів. Черги пріоритетів широко використовуються в операційних системах для виконання процесів із найвищим пріоритетом насамперед. Пріоритет процесу може бути встановлений на основі необхідного процесорного часу для повного завершення процесу. Наприклад, якщо є три процеси, причому першому

процесу необхідно 5 мкс, щоб завершитися, другому процесу необхідно 4 мкс, а третій процес потребує 7 мкс, то другий процес матиме найвищий пріоритет, а отже, буде взятий із черги завдань і виконаний першим.

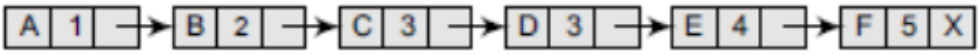


Рис. 3. Приклад реалізації черги з пріоритетом

Отже, черги з пріоритетом можуть застосовуватись для реалізації алгоритмів паралельної обробки з метою найбільш ефективного використання машинного часу.

Список використаних джерел

1. Грудзинський Ю. Є. Алгоритми та структури даних: навчальний посібник. 2022. URL: https://ela.kpi.ua/server/api/core/bitstreams/0db974f9-16fa-459c-9f19-fab0021222ed/content

2. Креневич А. П. Алгоритми і структури даних. Підручник. Київ: ВПЦ «Київський Університет», 2021. 200 с. URL: chttps://www.mechmat.univ.kiev.ua/wp-content/uploads/2021/09/pidruchnyk-alhorytmy-i-struktury-danykh.pdf

3. Програмна реалізація та дослідження алгоритмів паралельного швидкого сортування / В. О. Денисюк , Н. А. Потапова, О. В. Зелінська, М. Б. Тарасюк. *Вісник Хмельницького національного університету. Технічні науки.* 2023. № 4. С. 95–105. URL: http://journals.khnu.km.ua/vestnik/?page_id=41

УДК 004.1

Вишневський А. В., *здобувач*
1 курсу ОС «Магістр»
спеціальності 122 Комп'ютерні науки,
Горяшин А. С., *асистент*
кафедри інформаційних технологій

ІНФОРМАЦІЙНА СИСТЕМА ДЛЯ ПРОГНОЗУВАННЯ ЗАБРУДНЕННЯ ПОВІТРЯ ЗА ДОПОМОГОЮ АНАЛІЗУ МЕТЕОРОЛОГІЧНИХ ТА ЕКОЛОГІЧНИХ ДАНИХ

Донецький національний університет імені Василя Стуса, м. Вінниця

У сучасному світі проблема забруднення повітря стає дедалі актуальнішою. Викиди шкідливих речовин від промислових підприємств, транспорту та інших джерел негативно впливають на здоров'я людей і стан навколишнього середовища. Впровадження інформаційних систем для моніторингу та прогнозування рівня забруднення повітря є необхідним кроком для вирішення цієї проблеми.

Інформаційні системи для прогнозування забруднення повітря дають змогу вчасно виявляти та реагувати на критичні ситуації, що може значно знизити негативний вплив на здоров'я населення та довкілля. Використання сучасних інформаційних технологій та аналізу великих обсягів даних забезпечує підвищення точності та ефективності прогнозів.

Останні дослідження показують, що аналіз метеорологічних даних у поєднанні з даними про забруднення повітря значно покращує точність моделей прог-

нозування. Використання методів машинного навчання дає змогу ефективно обробляти великі обсяги даних та виявляти закономірності, що сприяють точнішому передбаченню рівня забруднення.

Інформаційна система для прогнозування забруднення повітря вирішує проблему моніторингу поточного стану забруднення в режимі реального часу та прогнозування можливих змін на основі аналізу метеорологічних та екологічних даних. Це допомагає своєчасно надавати рекомендації щодо зниження рівня забруднення та попередження населення про можливі ризики.

Впровадження таких систем має кілька ключових переваг.

По-перше, вони забезпечують оперативний моніторинг якості повітря, що дає змогу швидко виявляти перевищення допустимих норм та вживати необхідних заходів.

По-друге, прогнозування на основі аналізу метеорологічних умов допомагає передбачити потенційні епізоди підвищення рівня забруднення та заздалегідь вжити профілактичних заходів.

По-третє, інформаційні системи надають можливість накопичувати й аналізувати історичні дані, що сприяє глибшому розумінню динаміки забруднення та його причин.

Отже, впровадження інформаційної системи для прогнозування забруднення повітря дає змогу значно підвищити ефективність екологічного моніторингу та знизити негативний вплив забруднення на здоров’я населення. Подальші дослідження мають бути спрямовані на вдосконалення моделей прогнозування та інтеграцію нових джерел даних.

Список використаних джерел

1. Ковальчук І. В., Марченко О. В. Використання метеорологічних даних для прогнозування забруднення повітря. *Екологічний вісник.* 2020. URL: https://www.ecoleague.net/diialnist/vydannia-vel/ekolohichnyi-visnyk/2020-rik.

2. Моніторинг атмосферного повітря. *Офіційний сайт Міністерства захисту довкілля та природних ресурсів України.* URL: https://mepr.gov.ua/

3. Монастирський Л., Гура В. Прогнозування якості повітря за допомогою машинного навчання. *Електроніка та інформаційні технології*. 2023. Вип. 22. С. 57–67 URL: https://www.researchgate.net/publication/374748191_AIR_QUALITY_FORECASTING_WITH_THE_SUPPORT_OF_MACHINE_LEARNING

УДК 004.4

Сивак Д. І., *здобувач 2 курсу спеціальності 122 Комп'ютерні науки,*
Вєтров О. С., *старший викладач кафедри інформаційних технологій*

РЕФЛЕКСІЯ ТА ІНТРОСПЕКЦІЯ В PYTHON

Донецький національний університет імені Василя Стуса, м. Вінниця

Рефлексія в програмуванні – це здатність програми аналізувати та змінювати свою структуру і поведінку під час виконання. За допомогою рефлексії програмісти можуть отримувати доступ до інформації про класи, методи та поля об'єктів, а також змінювати їх. Завдяки рефлексії програмування може стати гнучкішим і масштабованішим, що особливо важливо для великих проєктів [1].

Інтроспекція в програмуванні – це здатність програми до аналізу власного коду та структури під час виконання. Це означає, що програма може досліджувати власну структуру, отримувати інформацію про класи, функції, модуль та інші складники коду.

Основна відмінність між рефлексією та інтроспекцією полягає у напрямах аналізу. Рефлексія спрямована на аналіз та модифікацію структури й поведінки програми під час її виконання, тоді як інтроспекція спрямована на аналіз власного коду та структури програми під час її виконання. Отже, рефлексія дає змогу програмі змінювати свою поведінку на льоту, водночас інтроспекція надає програмі можливість аналізувати власну структуру та інформацію про код без зміни його виконання.

Важливість рефлексії та інтроспекції в мові програмування Python:

1. Гнучкість та динамічність. Python – це мова з високим рівнем рефлексії, яка надає програмістам можливість динамічно маніпулювати об'єктами та їх властивостями, що сприяє створенню гнучких і масштабованих програм.

2. Відладка та тестування. Рефлексія та інтроспекція значно полегшують процеси відладки та тестування, що дає змогу виявляти помилки та аналізувати поведінку програми.

3. Метапрограмування. Python дає змогу програмістам створювати код, який може змінювати власну структуру або генерувати новий код під час виконання програми, що відкриває широкі можливості для метапрограмування.

4. Фреймворки та бібліотеки. Багато фреймворків та бібліотек Python, як-от Django, Flask або NumPy, використовують рефлексію та інтроспекцію для виконання різних завдань, як-от маршрутизація URL, серіалізація об'єктів або маніпулювання даними.

Об'єктна модель Python:

Python має потужну об'єктну модель, яка дає змогу використовувати рефлексію для маніполювання об'єктами та їх властивостями. Всі об'єкти в Python є екземплярами класів, а класи можуть мати методи та атрибути, які можуть бути змінені або отримані в часі виконання.

Функції type() та isinstance():

Функція type() дає змогу отримати тип об’єкта в часі виконання. Вона допомагає динамічно визначати типи об’єктів та виконувати дії залежно від цього типу.

Функція isinstance() використовується для перевірки того, чи є об’єкт екземпляром певного класу або його підкласу. Це дає змогу програмі динамічно адаптуватися до типів об’єктів та виконувати різні дії залежно від їх класу.

Використання атрибутів об’єктів та модулів для рефлексії:

Python надає зручний спосіб отримання доступу до атрибутів об’єктів та модулів у часі виконання. Це може бути зроблено за допомогою вбудованих функцій, як-от getattr(), setattr та hasattr(), які дають змогу отримувати, встановлювати та перевіряти наявність атрибутів об’єктів. Також можна використовувати метод dir(), щоб отримати список усіх доступних атрибутів об’єктів або модуля.

Використання модуля inspect:

Модуль inspect в Python надає зручні інструменти для інтроспекції, тобто динамічного аналізу коду в часі виконання. Цей модуль дає змогу отримувати інформацію про об’єкти, функції, класи та модулі, а також атрибути цих об’єктів.

Методи модуля inspect, як-от getmembers(), getsource(), getmodule(), допомагають отримувати різноманітну інформацію про об’єкти в часі виконання. Наприклад, ви можете отримати список атрибутів об’єкта, джерело функції, модуль, до якого вона належить, тощо [2].

Динамічне отримання атрибутів та їх значень:

За допомогою модуля inspect можна динамічно отримувати атрибути об’єктів та їх значення в часі виконання. Наприклад, ви можете використовувати методи getattr() та setattr() для отримання та встановлення значень атрибутів об’єктів.

Загалом модуль inspect допомагає програмістам глибоко досліджувати та аналізувати свій код у часі виконання, що дає змогу створювати більш гнучкі та динамічні програми.

Практичне використання рефлексії та інтроспекції у реальних проєктах.

Автоматичне завантаження плагінів. У деяких програмних системах, як-от редактори тексту чи вебсервери, рефлексія використовується для автоматичного виявлення та завантаження плагінів під час запуску програми. Завдяки рефлексії програма може аналізувати папки або конфігураційні файли, знаходити класи, які реалізують певний інтерфейс, і динамічно завантажувати їх.

Конфігураційні файли. У деяких вебфреймворках рефлексія використовується для читання конфігураційних файлів та автоматичного налаштування програми згідно з цими налаштуваннями. Наприклад, програма може динамічно завантажувати параметри підключення до бази даних або налаштування маршрутів маршрутизації HTTP.

Документація коду. Великі проєкти часто використовують інтроспекцію для автоматичної генерації документації коду. За допомогою інтроспекції програма може аналізувати структуру класів, методів та атрибутів і генерувати документаційні файли у форматі, зрозумілому для розробників.

Відладка та аналіз коду. Інтроспекція допомагає розробникам відлажувати програмний код та виявляти проблеми шляхом аналізу структури об’єктів та їх атрибутів в часі виконання. Наприклад, програма може динамічно виводити ін-

формацію про типи та значення атрибутів об’єктів, що допомагає зрозуміти, що відбувається в певному місці коду під час його виконання.

Перевагами рефлексії є гнучкість та динамічність, наявна можливість метапрограмування, полегшення відладки та тестування. Натомість очевидними перевагами інтроспекції є можливість аналізу та зміни структури програми і гнучкість у взаємодії з об’єктами під час виконання. Варто зазначити також відповідні недоліки рефлексії: пониження продуктивності; складність розуміння коду; потенційні проблеми безпеки. Для інтроспекції недоліками будуть потенційна складність у реалізації або розумінні коду.

Отже, за результатами аналізу переваг і недоліків рефлексії та інтроспекції варто зазначити, що обидві концепції мають свої вагомі переваги й обмеження. Рефлексія надає можливість гнучкої та динамічної роботи з кодом, сприяючи метапрограмуванню і полегшуючи відладку та тестування. Проте вона може призвести до пониження продуктивності та складнощів у розумінні коду. З іншого боку, інтроспекція дає змогу аналізувати та змінювати структуру програми у часі виконання, що полегшує взаємодію з об’єктами та сприяє відладці. Проте вона також може призвести до складнощів у розумінні коду та потенційних проблем безпеки.

Отже, вибір між рефлексією та інтроспекцією повинен ґрунтуватися на конкретних потребах та вимогах проєкту, а також з обов’язковим урахуванням переваг і недоліків кожної з цих концепцій.

Список використаних джерел

1. Foxmind: вебсайт. https://foxminded.ua/refleksiia-v-prohramuvanni/ (дата звернення: 15.05.2024).

2. Devopedia: вебсайт. https://devopedia.org/introspection-in-python (дата звернення: 15.05.2024).

СЕКЦІЯ 4
ТЕХНОЛОГІЇ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ ТА ПРИЙНЯТТЯ РІШЕНЬ

УДК 004.89

Бурківський О. С., *здобувач 2 курсу спеціальності 122 Комп'ютерні науки,*
науковий керівник:
Фриз І. В., *канд. фіз.-мат. наук, старший викладач кафедри інформаційних технологій*

ЙМОВІРНІСНІ МЕТОДИ В МАШИННОМУ НАВЧАННІ

Донецький національний університет імені Василя Стуса, м. Вінниця

Вступ. Одним із ключових аспектів машинного навчання є застосування ймовірнісних методів, які дають змогу ефективно моделювати та управляти невизначеністю в даних.

Теорія ймовірностей, в основі якої лежить математична обробка ймовірностей подій та випадкових величин, відіграє одну з ключових ролей у розвитку та застосуванні різноманітних методів машинного навчання. Вона допомагає створювати моделі, які не лише адаптуються до наявних даних, але й враховують ступінь невизначеності та ймовірнісні закономірності.

Розглянемо один з аспектів застосування теорії ймовірності у машинному навчанні, а саме, як ймовірнісні методи використовуються для кластеризації і допомагають вирішувати складні завдання, пов'язані з аналізом даних та прийняттям рішень.

Виклад основного матеріалу. Кластеризація, або кластерний аналіз – це статистична процедура, завдання якої полягає в розбитті вибірки об'єктів на підмножини (кластери), які не перетинаються, так, що об'єкти всередині одного кластера мають високу подібність між собою, а об'єкти різних кластерів відрізняються. Задача кластеризації є однією з найпоширеніших задач без нагляду в машинному навчанні та аналізі даних. Кластеризація використовується для виявлення структури в наборах даних, групуванні подібних об'єктів і для виявлення схожих підгруп серед даних.

Як описано у [1], алгоритми кластеризації розподіляються на два типи ієрархічних методів: агломеративні та дивизимні. Перші ґрунтуються на тому, що всі об'єкти спочатку розташовуються у власні класи, а потім, використанням метрики подібності, ці об'єкти поступово об'єднуються, зменшуючи кількість кластерів до моменту отримання одного. Другі – навпаки, починають з віднесення об'єктів до одного кластера, а зі збільшенням відстані розподіляють їх за окремими кластерами.

Розглянемо типовий алгоритм кластерного аналізу, наведений у [1], на основі аналізу робіт [2, 3, 4], який застосовується у різних галузях, зокрема і в бізнесі. Нехай $A = \{a_1, a_2, \dots, a_n\}$ – множина об'єктів, а B – множина номерів кластерів. Обирається метрика $d_{ij}(x_{ik}, x_{jk})$, водночас є формулою відстані. Мета полягає у розбитті множини A на підмножини (кластери), які не перетинаються, і кожен кластер містить об'єкти, що близькі за метрикою $d_{ij}(x_{ik}, x_{jk})$, водночас об'єкти

різних класів значно відрізняються. Кожному об'єкту a_i призначається номер кластера B_i.

Множина A може містити об'єкти з різними одиницями вимірювання або різним діапазоном значень, тому необхідно здійснити нормалізацію вхідних даних. Для цього можна скористатися MinMax-нормалізацією або Z-нормалізацією.

Перетворення вхідних даних за допомогою MinMax-нормалізації виконується так:

$$x' = \frac{x - \min[X]}{\max[X] - \min[X]} \quad \text{або} \quad x' = a + \frac{x - \min[X]}{\max[X] - \min[X]}(b - a)$$

для даних, які знаходяться в діапазоні [0, 1] або ж для довільного діапазону $[a, b]$ відповідно.

У процесі Z-нормалізації кожен вхідний параметр перетворюється таким чином, щоб його середнє значення було рівним 0, а стандартне відхилення – 1. Перетворення вхідних даних за допомогою Z-нормалізації здійснюється за такою формулою

$$x' = \frac{x - M[X]}{\sigma[X]},$$

де $M[X]$ – математичне сподівання, $\sigma[X]$ – середнє квадратичне відхилення.

У кластерному аналізі можуть використовуватися різні міри подібності (зазвичай описуються невід'ємною функцією), як-от коефіцієнт кореляції, міри відстані, зокрема евклідова відстань між двома точками на площині: $d_{AB} = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$, коефіцієнти асоціативності та ймовірнісні коефіцієнти подібності.

У кластеризації широко використовується функція Гауса, особливо для методів, що базуються на моделі гаусової суміші (GMM). Ці методи моделюють дані як суміш декількох розподілів Гауса, що дає змогу ефективно виявляти кластери з різними формами та розмірами. Кожен кластер описується власним розподілом Гауса, параметри якого потрібно визначити, і з урахуванням цього розподілу знайти ймовірність, і з якою кожна точка належить до певного кластера.

Нагадаємо, що для d-вимірного випадку функція щільності розподілу Гауса має такий вигляд [5]:

$$f(x|\mu, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)},$$

де x – d-вимірний вектор спостереження, μ – вектор середніх значень, Σ – коваріаційна матриця, $|\Sigma|$ – детермінант коваріаційної матриці, Σ^{-1} – обернена коваріаційна матриця.

На заключному кроці обирається та впроваджується відповідний алгоритм кластерного аналізу, а результати перевіряються на достовірність.

Висновки. Застосування ймовірнісних методів у машинному навчанні дає змогу ефективно моделювати та управляти невизначеністю в даних, що є важливим для створення адаптивних та точних моделей. Методи кластеризації на основі моделей, як-от GMM, дають змогу виявляти кластери з різними формами та розмірами, враховуючи ступінь невизначеності. Для покращення якості класте-

ризації важливо здійснювати нормалізацію даних за допомогою MinMax або Z-нормалізації, що забезпечує порівнянність значень об'єктів. Використання функції густини ймовірності Гауса та алгоритму GMM сприяє точному визначенню параметрів кластерів та їх ймовірностей, підвищуючи гнучкість і точність аналізу, що розширює можливості прийняття рішень у різних галузях.

Список використаних джерел

1. Шевченко С. М., Жданова Ю. Д., Шевцова Т. І. Застосування кластерного аналізу для просування бізнесу у соціальних мережах. *Вісник ХНТУ*. 2023. № 4(87). С. 271–281. DOI: 10.35546/kntu2078-4481.2023.4.32 (дата звернення 08.05.2024).
2. Koirala J. Understanding the Use of Cluster Analysis in Business (March 27, 2023). DOI: 10.2139/ssrn.4400674 (дата звернення 08.05.2024).
3. Безпарточний М. Г. Використання кластерного аналізу при оцінці ефективності діяльності торговельних підприємств. *Торгівля, комерція, підприємництво:* збірник наукових праць. Львів: Львівська комерційна академія. 2014. Вип. 17. С. 24–27.
4. Модель експертної системи для медичного скринінгу на основі методів кластерного аналізу / С. М. Шевченко, Ю. Д. Жданова, О. В. Негоденко, В. А. Куцук. *Moderní aspekty vědy:* XXVII: díl mezinárodní kolektivní monografie. Mezinárodní Ekonomický Institut s. r. o. Česká republika. Mezinárodní Ekonomický Institut s. r. o., 2023. С. 478–494. URL: http://perspectives.pp.ua/public/site/mono/mono-27.pdf (дата звернення 08.05.2024).
5. Reynolds D. Gaussian Mixture Models / S. Z. Li, A. Jain (eds). *Encyclopedia of Biometrics*. Springer, Boston, MA, 2009. DOI: 10.1007/978-0-387-73003-5_196 (дата звернення 08.05.2024).

УДК 004.8:519.2:658.6

Яценко В. В., *здобувач 2 курсу спеціальності 122 Комп'ютерні науки,*
Комаров В. Ф., *канд. техн. наук, старший викладач кафедри інформаційних технологій*

ПОРІВНЯННЯ АЛГОРИТМІВ КЛАСИФІКАЦІЇ НА ПРИКЛАДІ ЗАДАЧІ ВИЗНАЧЕННЯ КАТЕГОРІЇ ТОВАРУ МАГАЗИНУ

Донецький національний університет імені Василя Стуса, м. Вінниця

Вступ. Швидкий аналіз і обробка вхідних даних – основний виклик сучасних автоматизованих систем. Від швидкості прийняття рішень залежить ефективність вирішення багатьох прикладних задач кібернетики у різних галузях застосування. Обсяги даних, які необхідно аналізувати, зростають експоненційно, здатність оперативно обробляти великі потоки даних та виявляти в них критично важливі закономірності стає конкурентною перевагою під час розробки нових програмних продуктів, а в окремих задачах є вирішальною. Затримка з прийняттям рішень на основі отримуваної інформації може мати вкрай негативні наслідки – від збитків у бізнесі до небезпечних помилок у критично більш важливих системах. В умовах активного екстенсивного розвитку й проникнення смарт-технологій у всі сфери життєдіяльності людини ручне програмування рішень під кожен набір даних не є швидким і оптимальним вибором.

Одним зі способів вирішення проблеми є методи машинного навчання [1]. Машинне навчання – це галузь штучного інтелекту, що спеціалізується на розробці і дослідженні моделей та алгоритмів, що здатні навчатися з даних, аналізувати їх і на основі аналізу приймати рішення без явних інструкцій.

Метою дослідження є аналіз та порівняння різних методів класифікації для визначення доречного для поставленої задачі. У межах роботи необхідно пояснити, як відбувається машинне навчання, представити кожен алгоритм, визначити переваги і недоліки та підсумувати відповідну інформацію.

Виклад основного матеріалу. Машинне навчання моделі складається з декількох етапів, а саме: збір та підготовка даних, вибір моделі машинного навчання, безпосередньо навчання моделі, оцінка та налаштування моделі.

На першому етапі відбувається початковий збір інформації. Відбір інформації може відбуватися з вебсайтів, сенсорів обладнання, баз даних, даних з платформ соціальних мереж. Так, наприклад, для задачі визначення категорії товару в магазині необхідно відібрати найменування і ознаки товарів. Далі дані піддаються обробці. Відбувається очищення початкових даних, видалення пропусків або їх заповнення методами інтерполяції чи спеціальними значеннями. До того ж дані перевіряються на предмет помилок. Вибірка перевіряється на наявність аномальних значень – даних, що виходять за задані діапазони, або нереалістичної інформації. Обов'язково видаляються дублікати, оскільки наявність повторюваних даних може призвести до погіршення здатності розпізнавати невідомі дані і навіть до перенавчання моделі. Згодом початкові дані нормалізуються і стандартизуються [2]. Ідея нормалізації і стандартизації полягає в тому, щоб подати дані для моделі в однаковому масштабі в однакових мірах вимірювання, тобто так, щоб модель правильно порівнювала дані з певними ознаками і не надавала перевагу ознаці через неправильний акцент у даних.

На другому етапі відбувається вибір моделі машинного навчання, яка оптимально підходить для вирішення задачі з урахуванням складності, характеристик даних і витрат ресурсів. До основних типів моделей машинного навчання належать: регресійні моделі – моделі, які займаються прогнозуванням певної інформації; класифікаційні – такі, що відносять певне явище до класу згідно з його ознаками; кластеризаційні – моделі, що групують подібні явища. У випадку категоризації товарів класифікаційна модель є оптимальною для розв'язування задачі.

Основні методи класифікації інформації – це логістична регресія, дерево рішень, випадковий ліс і метод опорних векторів. У роботах [3, 4] подаються нові методи прискореного і більш ефективного машинного навчання моделей. За допомогою паралелізованої обробки інформації і методів ансамблювання, тобто поєднання прогнозувань декількох моделей, можна значно пришвидшити процес машинного навчання, зробити його більш ефективним, знизити потреби у ресурсах і спростити подальше масштабування моделі.

Логістична регресія застосовує модель, яка описує зв'язок між незалежними змінними і залежними змінними. Зазвичай це лінійне рівняння, де кожна змінна має свій коефіцієнт. Після формування моделі використовується логістична функція, яка перетворює вихідні значення рівняння у ймовірність, у діапазоні значень від 0 до 1 (подія не відбудеться і подія відбудеться). Далі класифікація

здійснюється за допомогою обчислення ймовірності для точки даних. Логістична регресія дуже проста у використанні і доволі ефективна. Недоліки методу – регресія не підходить для задач з більше ніж двома категоріями і припускає, що зв'язок між незалежними змінними і залежними лінійний, що може призвести до неточних прогнозів. Цей метод не є ефективним у використанні через відсутність можливості роботи з множинною класифікацією [5].

Дерево рішень працює так: починаючи з кореневого вузла вибудовується дерево, за допомогою якого згодом будуть прийматись рішення. У кожному вузлі вибирається ознака, яка найкраще описує дані за змінною, яку користувач хоче прогнозувати. Ця ознака використовується для розмежування даних на групи, представлені дочірніми вузлами. Процес повторюється, поки не будуть досягнуті кінцеві вузли. Під час аналізу змінної відбувається проходження дерева від кореневого вузла до листа. Дерево рішень – доволі простий у використанні метод, але водночас дуже ефективний, особливо для великих наборів даних. До того ж дерево рішень можна візуалізувати, що полегшує аналіз рішень моделі. Однак цей метод потребує особливо ретельної обробки даних, оскільки дерево чутливе до якості даних. Дерево рішень схильне до перенавчання, що ускладнює навчання з різними випадками класифікації.

Випадковий ліс – ансамбль дерев рішень. Цей метод є доволі оптимальним для вирішення даної задачі. Фактично алгоритм лісу аналогічний побудові дерева, з відмінністю у тому, що вибудовується множина дерев рішень, і під час прийняття рішень відбувається голосування кожного дерева. Прогнози дерев узагальнюються, і на основі остаточного підрахунку голосів відбувається прийняття рішення. Випадковий ліс поєднує в собі високу точність і надійність. Цей алгоритм дуже стійкий до шуму. Основним недоліком є великі затрати в ресурсах для побудови цієї моделі і низька швидкість прийняття рішень. До того ж рішення таких моделей важко інтерпретувати через поєднання прогнозів багатьох дерев. Випадковий ліс буде доречним у задачі з категоріями товарів.

Метод опорних векторів шукає гіперплощину, тобто багатовимірну пряму, яка розділяє простір між двома точками даних різних класів, причому відстань від найближчої точки площини має бути максимальна. Точки даних, які розташовані до гіперплощини найближче, називаються опорними векторами. На основі того, в яку площину потрапить нова точка, буде визначено, до якого класу належить ця точка. Алгоритм дуже точний і стійкий до шуму, доволі легко інтерпретується, але побудова такої моделі може бути ресурсозатратним процесом, а процес прийняття рішень повільним.

Далі відбувається ***навчання моделі і оцінка її продуктивності.*** Залежно від результатів тестування можуть бути модифіковані параметри моделі. Якщо результати будуть невдалі, навчання може початися спочатку.

Висновки. У підсумку машинне навчання значно спрощує процес обробки і прийняття рішень. Кожен з алгоритмів має свої переваги та недоліки і може бути корисним у певній ситуації. Логістична регресія – дуже швидкий метод, але має незадовільну точність у комплексних задачах і неактуальний у задачах із множинною класифікацією. Дерево рішень надає оптимальний за часом і точністю результат, але цей метод нестійкий до шуму і його успішність сильно залежить

від початкових даних. Випадковий ліс стійкий до шуму, але ресурсозатратний у генерації і роботі. Метод опорних векторів дає максимально точний результат, але порівняно повільний у роботі. Вибір алгоритму повинен відповідати потребам – обсягу вхідних даних та вимогам користувача.

Список використаних джерел

1. Machine learning explained. 2024. URL: https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained (дата звернення: 16.05.2024).

2. Optimization vs standartization. 2024. URL: https://www.geeksforgeeks.org/normalization-vs-standardization/ (дата звернення: 16.05.2024).

3. Harmon M., Klabjan D. Activation Ensembles for Deep Neural Networks. arXiv, 2017. DOI: 10.48550/arXiv.1702.07790 (дата звернення: 16.05.2024).

4. Luo H., Haffner P., Paiement J.-F. Accelerated Parallel Optimization Methods for Large Scale Machine Learning. arXiv, 2014. DOI: 10.48550/arXiv.1411.6725 (дата звернення: 16.05.2024).

5. What is logistic regression. 2024. URL: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-logistic-regression/ (дата звернення: 16.05.2024).

УДК 519.2

Мороз Д. В., *здобувачка 1 курсу спеціальності 111 Математика,*
Луценко А. В., *д-р філос. з математики, старший викладач кафедри прикладної математики та кібербезпеки*

ЗАСТОСУВАННЯ ЛІНІЙНОЇ АЛГЕБРИ В ІНТЕЛЕКТУАЛЬНОМУ АНАЛІЗІ ДАНИХ

Донецький національний університет імені Василя Стуса, м. Вінниця

Актуальність. У світі, що переповнений даними, важливість їх аналізу та інтерпретації зростає з кожним днем. Інтелектуальний аналіз даних стає ключовим інструментом для виявлення закономірностей, отримання цінних інсайтів та прийняття стратегічних рішень у різних сферах, від бізнесу до медицини. Одним із фундаментальних математичних інструментів, який забезпечує основу для інтелектуального аналізу даних, є лінійна алгебра. Розуміння та ефективне використання принципів лінійної алгебри в цьому контексті стає важливим завданням для дослідників, аналітиків та фахівців з обробки даних.

Метою роботи є розгляд застосування лінійної алгебри в інтелектуальному аналізі даних. У роботі будуть розглянуті основні поняття лінійної алгебри, як-от вектори, матриці, лінійні простори та операції над ними, а також конкретні сфери застосування цих принципів у сучасному інтелектуальному аналізі даних, як-от обробка та аналіз даних, методи машинного навчання, обробка зображень та сигналів, а також рекомендаційні системи. Результатом роботи буде розуміння важливості лінійної алгебри для ефективного використання інтелектуальних методів аналізу даних та виявлення перспектив подальшого використання цих знань у практиці.

Основними поняттями лінійної алгебри є вектори, матриці, лінійні простори, операції додавання та множення [1].

Застосування лінійної алгебри в інтелектуальному аналізі даних. Представлення даних у вигляді матриць: дані можуть бути представлені у вигляді матриць, де кожен рядок відповідає окремому спостереженню, а кожен стовпець – означенню або функції. Цей підхід дає змогу ефективно зберігати та обробляти великі обсяги даних, а також виконувати різноманітні операції за допомогою матричних операцій [1].

Зведення до матричних операцій: багато методів обробки даних, як-от факторний аналіз, аналіз головних компонент та кластерний аналіз, можуть бути сформульовані у вигляді матричних операцій. Наприклад, у методі головних компонент використовуються операції знаходження власних значень та власних векторів матриці коваріації даних.

Розв'язання систем лінійних рівнянь: у деяких випадках обробка даних включає розв'язання систем лінійних рівнянь, що можуть виникнути під час застосування певних методів, наприклад, у задачах оптимізації або моделюванні. Лінійна алгебра надає ефективні методи для розв'язання таких систем, що дає змогу отримати числові рішення задач.

Декомпозиція та розкладання матриць: у деяких методах обробки даних використовуються операції декомпозиції та розкладання матриць, як-от SVD (Singular Value Decomposition) або QR-розклад. Ці операції дають змогу розкрити основні структурні властивості даних та зменшити їх розмірність, що може полегшити подальший аналіз.

Визначення лінійних залежностей та кореляцій: лінійна алгебра дає змогу визначити лінійні залежності та кореляції між різними змінними в наборі даних. Це допомагає виявляти структуру та взаємозв'язки між змінними, що може бути важливим під час розуміння даних та прийняття рішень на їх основі.

Методи машинного навчання. *Лінійна регресія* – це метод, який використовує лінійну функцію для прогнозування значення залежної змінної на основі набору незалежних змінних. У лінійній регресії кожен прихований шар моделі може бути представлений у вигляді вектора параметрів, а сама модель – у вигляді матриці ваг. Застосування лінійної алгебри дає змогу ефективно оцінювати параметри моделі та робити прогнози на основі нових даних [2].

Метод головних компонент – це метод зменшення розмірності даних, який знаходить лінійну комбінацію змінних, що максимізує дисперсію даних. У PCA використовуються матричні операції для знаходження власних значень та власних векторів коваріаційної матриці даних.

За допомогою лінійної алгебри виконуються операції зменшення розмірності та відновлення даних після зменшення розмірності.

Метод опорних векторів – це метод класифікації, який шукає оптимальну гіперплощину, що розділяє класи даних у просторі вищих вимірів. У SVM лінійна алгебра використовується для знаходження оптимальних ваг та зсуву, які визначають гіперплощину. Операції лінійної алгебри дають змогу ефективно знаходити оптимальні параметри моделі та робити прогнози для нових даних.

Логістична регресія – це метод класифікації, який використовує лінійну комбінацію змінних та логістичну функцію для передбачення ймовірності належності до певного класу. У логістичній регресії параметри моделі можуть бути оцінені за допомогою методів лінійної алгебри, наприклад, методом градієнтного спуску [3].

У багатьох архітектурах нейронних мереж, як-от багатошарові персептрони, лінійна алгебра використовується для обчислення ваг та активаційних функцій. Операції лінійної алгебри виконуються для передачі сигналів між шарами та оновлення параметрів моделі під час навчання.

Обробка зображень та сигналів з використанням лінійної алгебри. Зображення та сигнали можуть бути представлені у вигляді матриць чисел, де кожен елемент відповідає інтенсивності пікселя або значенню сигналу. Це представлення дає змогу виконувати різноманітні операції обробки та аналізу за допомогою матричних операцій [4].

Лінійна алгебра використовується для застосування різних фільтрів до зображень, як-от розмивання, розкриття різкості, видалення шуму тощо. Фільтри часто представлені у вигляді матриць, які використовуються для зміни значень пікселів зображення. Лінійна алгебра використовується для зменшення обсягу зображень та сигналів шляхом видалення зайвої інформації або скорочення кількості бітів, необхідних для представлення даних. Методи компресії, як-от Singular Value Decomposition (SVD) або Wavelet Transform, використовують матричні операції для зменшення розміру даних. Після обробки або компресії зображень та сигналів лінійна алгебра може бути використана для відновлення початкових даних [5]. Матричні операції використовуються для виявлення взаємозв’язків та структурних особливостей у зображеннях та сигналах.

Висновки. Лінійна алгебра відіграє ключову роль в інтелектуальному аналізі даних, що є важливою галуззю у сучасному світі. Її застосування можливе від обробки та аналізу даних до розробки та вдосконалення методів машинного навчання. Завдяки лінійній алгебрі можливо ефективно представляти, обробляти та аналізувати складні набори даних, здійснювати класифікацію, прогнозування та рекомендації. Її розуміння є необхідним для фахівців у сфері аналізу даних та машинного навчання.

Використання лінійної алгебри в інтелектуальному аналізі даних є дуже перспективним.

Список використаних джерел

1. Strang G. Introduction to Linear Algebra, 2023.
2. Lang S. Linear Algebra. Springer, 2002.
3. Lay D., Lay S., McDonald J. Linear Algebra and Its Applications. Pearson, 2018.
4. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, 2009.
5. Golub H., van Loan C. Matrix Computations. Johns Hopkins University Press, 2012.

УДК 004.01

Парасочкін В. В., *здобувач*
1 курсу ОС «Магістр»
спеціальності 122 Комп'ютерні науки

ОГЛЯД ТА КЛАСИФІКАЦІЯ ЕКСПЕРТНИХ СИСТЕМ

Донецький національний університет імені Василя Стуса, м. Вінниця

Експертна система – це один з різновидів штучного інтелекту [1], який може імітувати знання та приймати певною мірою успішні рішення на основі цих знань в одній з визначених галузей. Цю систему для простоти і зрозумілості можна уявити як певний тип комп'ютерної програми, яка може містити в собі глибокі знання або певний досвід у сфері спеціалізації, як-от фінанси, медицина тощо [2]. Тобто можна сказати, що це система, яка містить у собі знання і здібності одного або, як буває в більшості випадків, декількох експертів у певній галузі застосування, і ця система може здійснювати логічні висновки на основі тих знань, які має, що допомагає кінцевому користувачу вирішувати певний і специфічний тип проблем або завдання. Також такі системи можуть проводити консультування, тестування, проєктування, навчання, діагностування тощо. Також її можна охарактеризувати як систему, яка включає в себе базу знань, на основі яких вже може вирішувати завдання в тій чи іншій галузі. Також її ще можна охарактеризувати як методологію адаптації алгоритму пошуку успішних рішень. Простими словами, експертна система – це система, що імітує здатність людини-експерта в вузько направленій галузі приймати рішення.

Початком застосування експертних систем можна вважати 1960–1970 рр., коли вони виникли на хвилі розвитку штучного інтелекту. Вони використовувались для вирішення конкретних задач у вузькоспеціалізованих областях. Вперше було реалізовано програму ELIZA [3], яка імітувала діалог із психотерапевтом, яка використовувала базу знань для взаємодії з користувачем.

У 1970–1980 рр. вони отримали ширше коло застосування, взявши фокус на медицину, фінанси, інженерію та управління. З'явилась система MYCIN для діагностики бактеріальних інфекцій та призначення антибіотиків.

1980–1990 рр. – це період дуже активного росту, коли системи ставали більш потужними і використовувались у тих самих сферах, але для вирішення більш складних проблем. У цей період з'явилась XCON – система для конфігурації комп'ютерних систем, що значно скорочувала час та витрати на конфігурацію.

У проміжок часу 1990–2000 рр. експертні системи отримали розвиток інтерактивності та мультимедійності. Один із прикладів CYC – проєкт зі створення бази загальних знань для штучного інтелекту.

У 2000–2010 рр. – експертні системи отримали інтеграцію з іншими різноманітними технологіями та дали змогу автоматизувати процеси. Wolfram Alpha – система, що використовувала методи штучного інтелекту для відповіді на запити користувачів.

Наприкінці 2010–2020 рр. експертні системи отримали прорив завдяки загальному розвитку штучного інтелекту та новим методикам розробки. Відбулася інтеграція експертних систем у технології, як-от Amazon Alexa.

Щодо сфер застосування, експертні системи завжди використовувались в медицині, фінансах, інженерії та військовій сфері для вирішення різного типу і складності завдань.

Основні функції, які може виконувати експертна система:

- моделює механізм мислення людини під час вирішення проблем у заданій предметній області;
- не тільки виконує обчислення, але й робить висновки на основі своїх знань.

Ці знання, які зазвичай описані спеціальною мовою, називають базою знань [4]. Вона потрібна для зберігання даних, що описують розглянуту область і також можуть містити правила які описують перетворення, власне кажучи, даних цієї області;

- завдяки своїй практичній орієнтованості стає незамінним інструментом у наукових дослідженнях та комерційній діяльності, даючи змогу вирішувати складні задачі, які раніше були доступні лише для досвідчених експертів;
- важливою характеристикою є продуктивність, яка визначається часом, необхідним для отримання результату, та його достовірністю;
- може не лише генерувати рішення, але й уміти чітко та аргументовано пояснювати, чому саме це рішення було запропоновано. Користувач повинен мати доступ до всієї інформації, що підтверджує обґрунтованість прийнятого рішення, адже це показує наочний звіт роботи системи та сприяє кращому її розумінню.

Експертні системи зазвичай складаються з чотирьох ключових компонентів:

1. База знань [4] – це сховище інформації, що містить правила, факти та приклади, які експертна система використовує для прийняття рішень. Прийнято описувати базу знань на фізичному та логічному рівнях. Фізичний рівень – рівень фізичного представлення даних, описує адресування та організацію даних у зовнішній пам’яті комп’ютера. Логічний рівень даних являє собою абстрактну модель предметної області, що описує її структуру, зв’язки між елементами та кількісні характеристики у вигляді даних.

2. Машина виведення [5] – цей компонент використовує правила бази знань, щоб вивести нові знання з інформації, наданої користувачем.

3. Інтерфейс користувача [6] – завдяки цьому інтерфейсу користувач може взаємодіяти з системою, надаючи їй інформацію та отримуючи рекомендації.

4. Підсистема пояснень [7] – ця підсистема пояснює користувачеві, як експертна система дійшла до того чи іншого рішення. Розрізняють два види пояснень:

4.1 Пояснення, які надаються користувачу за вимогою, тобто користувач у будь-який момент може зажадати від експертної системи пояснення своїх дій.

4.2. Пояснення отриманого рішення проблеми. Користувач може захотіти отримати пояснення після одержання рішення, як воно було отримано.

На рис. 1 можна побачити приблизну схему взаємодії користувача з експертною системою і змоделювати, який шлях проходить відповідь після того, як користувач почав взаємодіяти з системою.

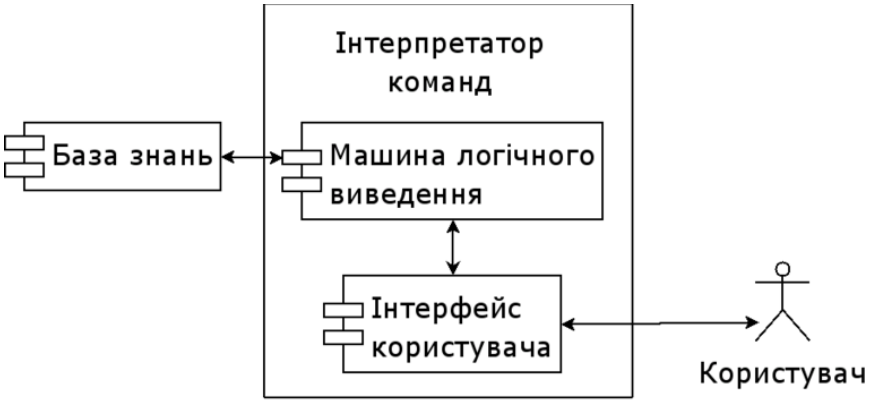


Рис. 1. Склад експертної системи

Список використаних джерел

1. Що таке штучний інтелект: історія, види та складові. *GigaCloud.* 2023. URL: https://gigacloud.ua/blog/navchannja/scho-take-shtuchnij-intelekt-istorija-vidi-ta-skladovi

2. Експертна система. *Geeksforgeeks.* 2024. URL: https://www.geeksforgeeks.org/expert-systems/

3. Ладанюк А. П. Машина виведення. *Основи систематичного аналізу.* 2014. С. 123–124.

4. Що таке інтерфейс. Різновиди інтерфейсів. *Cases.* URL: https://cases.media/article/sho-take-interfeis-riznovidi-interfeisiv

5. Кастілло Е., Гутьєррес Х. М., Хаді А. С. Підсистема пояснень. *Експертні системи та ймовірнісні мережеві моделі.* 2016. С 27–28.

УДК 004.9

Лавренюк Б. В., *здобувач*
1 курсу ОС «Магістр»
спеціальності 122 Комп'ютерні науки,
Потапова Н. А., *канд. екон. наук,*
доцент, доцент кафедри
інформаційних технологій

ТЕНДЕНЦІЇ РОЗВИТКУ СИСТЕМ РЕКОМЕНДАЦІЙ У СФЕРІ КІНЕМАТОГРАФІЇ: ВІД СТАНДАРТНИХ ПІДХОДІВ ДО ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ

Донецький національний університет імені Василя Стуса, м. Вінниця

У сучасному світі стрімкого розвитку цифрових технологій системи рекомендацій набули значної популярності, особливо у сфері кінематографії. Вони допомагають користувачам знайти фільми та серіали, які відповідають їхнім уподобанням, що підвищує задоволеність від перегляду та зменшує час на пошук контенту.

Зі зростанням кількості доступного контенту на платформах потокового відео, як-от Netflix, Amazon Prime та Disney+, проблема вибору стає все більш гострою. Використання інтелектуальних систем рекомендацій, які застосовують алгоритми машинного навчання та штучного інтелекту, дає змогу вирішити цю проблему ефективніше, ніж традиційні методи.

Тому **метою статті** є дослідження та аналіз тенденцій розвитку систем рекомендацій у сфері кінематографії, з акцентом на перехід від стандартних підходів до інтелектуального аналізу даних. Це включає оцінку ефективності сучасних алгоритмів та їх впливу на користувацький досвід.

Початково системи рекомендацій базувалися на простих алгоритмах, як-от колаборативна фільтрація та контент-орієнтовані методи. Колаборативна фільтрація використовує дані про вподобання багатьох користувачів для рекомендацій конкретному користувачеві. Вона поділяється на дві основні категорії: методи на основі користувачів і методи на основі об'єктів. У першому випадку система шукає користувачів зі схожими вподобаннями, а у другому – об'єкти (фільми або серіали), які мають схожі характеристики з тими, що вже переглянув користувач.

Контент-орієнтовані методи аналізують атрибути фільмів або серіалів (жанр, актори, режисери тощо) та рекомендують контент на основі подібності з переглянутими користувачем об'єктами. Незважаючи на свою простоту, ці методи мають обмеження, як-от недостатня персоналізація та нездатність враховувати змінні вподобання користувачів з часом.

Із розвитком технологій машинного навчання та штучного інтелекту стали доступними більш просунуті методи, що використовують нейронні мережі та глибоке навчання. Серед них виділяються рекурентні нейронні мережі (RNN) та мережі глибокого навчання (DNN).

Рекурентні нейронні мережі (RNN) використовуються для обробки послідовних даних, як-от історія переглядів користувача. Вони враховують час між переглядами та інші послідовні залежності, що дає змогу робити більш точні прогнози щодо майбутніх уподобань користувача. Дослідження показують, що RNN можуть значно підвищити точність рекомендацій завдяки здатності враховувати часові аспекти споживання контенту.

Окрім RNN, активно використовуються конволюційні нейронні мережі (CNN), які здатні обробляти мультимедійні дані (зображення та відео). У контексті рекомендаційних систем CNN застосовуються для аналізу візуальних характеристик фільмів та серіалів. Наприклад, система може враховувати стиль операторської роботи або кольорову палітру, що робить рекомендації ще більш персоналізованими.

Застосування гібридних моделей, які поєднують кілька методів, також є ефективним підходом. Наприклад, система може одночасно використовувати RNN для аналізу послідовності переглядів та контент-орієнтовані методи для врахування атрибутів фільмів. Це дає змогу забезпечити більш комплексний підхід до генерації рекомендацій, враховуючи як історичні дані, так і поточні вподобання користувача.

Дослідження також показують важливість інтеграції контекстуальних факторів у системи рекомендацій. Наприклад, час доби, день тижня, поточний настрій користувача та навіть погодні умови можуть впливати на його вподобання. Сучасні алгоритми використовують ці фактори для адаптації рекомендацій у реальному часі, що робить їх ще більш релевантними.

Отже, сучасні тенденції розвитку систем рекомендацій у сфері кінематографії демонструють перехід до більш складних і інтелектуальних моделей, які здатні

враховувати численні фактори та адаптуватися до змінних вподобань користувачів. Це підвищує задоволеність користувачів та сприяє більш ефективному використанню платформи.

Результати дослідження підтверджують, що розвиток систем рекомендацій у сфері кінематографії рухається в бік використання інтелектуального аналізу даних, що дає змогу підвищити їх ефективність і точність. Перспективи подальших досліджень включають розробку нових алгоритмів, які зможуть ще краще враховувати індивідуальні потреби та контекстуальні фактори, а також дослідження етичних аспектів використання таких систем.

Список використаних джерел

1. Ankit J. Role of a Movie Recommender System in the Streaming Industry. 2022. URL: https://www.muvi.com/blogs/movie-recommender-system/

2. Щербань В. С. Інформаційна технологія відбору відеоматеріалів. Програмні засоби на основі колаборативної фільтрації. URL: https://inmad.vntu.edu.ua/portal/static/D2FE0A0F-A676-4969-A811-F7F6946D54A4.pdf

3. Prabhat. Creating a Hybrid Neural Network for Movie Recommendations using TensorFlow Recommenders. 2023. URL: https://prabhatm27.medium.com/creating-a-hybrid-neural-network-for-movie-recommendations-using-tensorflow-recommenders-fdad13aa4979

4. Як працює система рекомендацій Netflix: *Netflix.* URL: https://help.netflix.com/uk/node/100639

УДК 004.852

Семенюк А. М., *здобувач 3 курсу спеціальності 122 Комп'ютерні науки,*
Хмелівський Ю. С., *асистент кафедри інформаційних технологій*

КЛАСТЕРНИЙ АНАЛІЗ СТАТИСТИЧНИХ МЕДИЧНИХ ДАНИХ НА МОВІ R

Донецький національний університет імені Василя Стуса, м. Вінниця

Вступ. Статистичний аналіз даних засобами мови R дає змогу розв'язувати основні статистичні задачі, візуалізувати дані, виконувати аналіз даних та прогнозування результату.

Засоби для візуалізації результатів обчислень дають змогу створювати різного виду графіки, що сприяють легкому сприйняттю інформації.

Значні можливості мови R для здійснення статистичних аналізів пов'язані з наявністю засобів лінійної і нелінійної регресії, класичних статистичних тестів, часових рядів (серій), кластерних обчислень і багато іншого [1].

Для успішного проведення статистичних досліджень важливо мати методологію, яка дає змогу проводити збір даних, що містять інформацію про вибірку об'єктів і потім упорядковує їх у відносно однорідні групи. Мова R має такий інструмент – це кластерний аналіз.

Перелік напрямів та галузей, де може бути використаний кластерний аналіз, дуже великий. Насамперед це пов'язано з головною метою такого дослідження – знаходження груп схожих об'єктів у вибірці.

Статистичний аналіз даних на мові R. У цій роботі розглядаються можливості цієї методології для обробки статистичних даних у медицині, зокрема – аналіз діяльності медико-соціальних експертних комісій України з питань медико-соціальної реабілітації осіб з інвалідністю [2].

Для забезпечення планової роботи медико-соціальної експертизи та реабілітації осіб з інвалідністю статистичні методи дають змогу виробити обґрунтовані, правильні організаційні та методологічні рішення, що пов'язане з ретельним аналізом отриманої інформації.

Статистичне дослідження як процес можна розбити на п'ять основних етапів:

1. Визначення мети та завдань дослідження.
2. Збір статистичного матеріалу.
3. Попередня обробка даних.
4. Розрахунок та інтерпретація узагальнюючих статистичних показників.
5. Моделювання та прогнозування.

Перші три етапи не вимагають використання математичного апарату. П'ятий ґрунтується на даних обробки, отриманих на попередньому етапі. Використання функцій мови R дає змогу виконувати розрахунки показників та їх різносторонню інтерпретацію з формуванням вихідних форм [3, 4]. Як приклад кластерного аналізу наведено результуючу таблицю для формування рангів показників реабілітації (рис. 1) в розрізі областей.

Як уже зазначалось, наявність великих об'ємів інформації потребує особливого підходу до її обробки. По-перше, необхідно з'ясувати, як це виконати. Найефективніше – виконати сортування на групи для можливості попереднього аналізу. Групування за методом кластеризації – це метод, коли в самому алгоритмі відбору закладено пошук ознак, які об'єднують дані за визначеним критерієм. Тобто алгоритм із первинного потоку інформації виявляє параметри з подібними характеристиками.

Внаслідок виконання цієї процедури ми отримаємо масив даних розбитий на окремі кластери (підмножини). Елементи у кластерах мають схожі ознаки, водночас мають суттєву відмінність від елементів іншого кластера – підмножини між собою не перетинаються.

Кластери можна утворювати, ґрунтуючись на відстані між ними, на щільності ділянок у просторі даних, інтервалах або на конкретних статистичних розподілах [5]. Формування їх критеріїв – це ітераційний процес, він залежить від мети використання результатів і від самого вхідного набору інформації. Цей потік може мати будь-який розподіл даних:

- ✓ порядковий;
- ✓ інтервальний;
- ✓ категорійний.

Допускається також об'єднання типів, але в такому випадку ускладнюється аналіз (кластеризація).

Рангові місця служби МСЕ областей України
за показниками повної реабілітації та реабілітації інвалідів ІІІ групи
за 2022 рік (на 100 переоглянутих) [1]

Рангові місця	Адміністративні території	Показник повної реабілітації	Рангові місця	Адміністративні території	Не визнані інвалідами серед переоглянутих інвалідів ІІІ групи
1	Дніпропетровська	3,27	1	м. Київ	5,38
2	м. Київ	2,77	2	Дніпропетровська	4,43
3	Запорізька	2,70	3	Запорізька	3,78
4	Вінницька	2,47	4	Вінницька	3,03
4	Черкаська	2,47	4	Черкаська	3,03
6	Миколаївська	2,21	6	Миколаївська	2,89
7	Кіровоградська	1,94	7	Київська	2,71
8	Київська	1,71	8	Кіровоградська	2,69
9	Херсонська	1,41	9	Херсонська	2,12
10	Сумська	1,34	10	Сумська	1,87
11	Тернопільська	1,26	11	Одеська	1,70
12	Луганська	1,23	12	Тернопільська	1,65
13	Полтавська	1,06	13	Полтавська	1,55
14	Волинська	0,90	14	Луганська	1,45
15	Одеська	0,88	15	Волинська	1,29
16	Закарпатська	0,87	16	Донецька	1,17
17	Донецька	0,85	17	Закарпатська	1,16
18	Хмельницька	0,83	18	Хмельницька	1,08
19	Чернівецька	0,76	19	Чернівецька	1,01
20	Львівська	0,69	20	Львівська	0,84
21	Харківська	0,58	21	Харківська	0,83
22	Чернігівська	0,55	22	Чернігівська	0,76
23	Житомирська	0,51	23	Житомирська	0,74
24	Рівненська	0,49	24	Рівненська	0,61
25	Івано-Франківська	0,14	25	Івано-Франківська	0,19
	В Україні, 2022 р.	1,62		В Україні, 2022 р.	2,27

Рис. 1. Зразок таблиці вихідних даних

Математично кластеризація описується виразом:

$$S(c) = \sum_{i=1}^{N} s(i, c_i) + \sum_{k=1}^{N} \delta(c_k), \quad (1)$$

де c_k – множина вхідних елементів; $s(i, c_i)$ – множина подібності; $\delta(c_k)$ – символ обмежувач; k – критерій відбору (середина діапазону ранжування).

У подібному алгоритмі розподіл на кластери можна представити як завдання дискретної максимізації з деякими обмеженнями.

Вибір мови R пов'язаний з її орієнтацією на розв'язок і аналіз статистичних задач. Під час її роботи можливо виконувати одразу багато інструкцій, що записані в окремому файлі (скрипті) і збираються у пакети (packages) користувача.

Середовище розробки та наявні базові пакети дають змогу провести кластерний аналіз за допомогою алгоритму *k*-відбору, а можливості графіки покращують сприйняття результатів. Результатом задачі кластеризації з використанням цього алгоритму є векторні набори даних. Робота з програмами на мові R інтуїтивно проста та не потребує спеціальних навичок, а пакет доступний на безкоштовній основі, що сприяє його використанню.

Висновки. Медико-соціальну експертизу в Україні у 25 областях здійснювали 362 МСЕК. Об'єм звітних даних, які вони підготовляють, значний. Тому одним з основних завдань МСЕК, особливо на сучасному етапі реформування охорони здоров'я в Україні, є регулярне проведення організаційно-методичної роботи, яка неможлива без аналітичної інформації. Підготовка таких даних, їх інтерпретація та групування за потрібними ознаками може бути виконана з використанням програми кластерного аналізу на мові R навіть особами з базовою комп'ютерною підготовкою.

Список використаних джерел

1. Семенюк А. М., Хмелівський Ю. С. Статистичний аналіз медичних даних на мові R. *Прикладні аспекти сучасних міждисциплінарних досліджень*: матеріали II Міжнародної науково-практичної конференції (м. Вінниця, 24 листопада 2023 р.). Вінниця: ДонНУ імені Василя Стуса. 2023. 282 с.

2. Основні показники медико-соціальної реабілітації осіб з інвалідністю в Україні за 2022 рік. Аналітико-інформаційний довідник / В. І. Шевчук, Р. Я. Перепелична, Л. О. Сторожук, І. В. Куриленко, Л. Г. Семененко, М. В. Семенюк, А. М. Семенюк. Вінниця: ФОП Данилюк В. Г. 2023. 119 с.

3. Методи програмування в R: вебсайт. URL: https://tvimc.jimdofree.com (дата звернення: 18.05.2024).

4. Селезньов О. Мова R для користувачів Excel. 2022. URL: https://selesnow.github.io/r4excel_users/index.html (дата звернення: 18.05.2024).

5. Junkui L., Yuanzhen W., Xinping L. LB HUST: A symmetrical boundary distance for clustering time series. *9th International Conference on Information Technology (ICIT'06)*. 2006. С. 203–208.

УДК 006.32:004.41:745.4:004.9

Коновалюк І. Л., *здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,*
Зелінська О. В., *канд. техн. наук, доцент, доцент кафедри інформаційних технологій*

ШТУЧНИЙ ІНТЕЛЕКТ ДИЗАЙНУ

Донецький національний університет імені Василя Стуса, м. Вінниця

Галузь комп'ютерної лінгвістики та інформатики, яка швидко розвивається, передбачає створення інтелектуальних машин, здатних виконувати завдання, які зазвичай вимагають людського таланту. Ці обов'язки можуть включати в себе розпізнавання мови чи зображень та більш складні обов'язки, як-от ігри чи водіння автомобіля [1].

Дизайн – це творчий підхід, який передбачає художньо-технічне проєктування виробів, комплексів і систем, спрямованих на максимальну сумісність створених об'єктів і навколишнього середовища з потребами людини, як практичними, так і естетичними [2].

Поєднання дизайну та штучного інтелекту сьогодні має вирішальне значення як для розробників, так і для споживачів з кількох причин:

1) допомагає оптимізувати процес розробки і автоматизувати автоматичні та рутинні завдання;

2) штучний інтелект може аналізувати величезні обсяги даних про користувачів та їх поведінку для покращення інтерфейсів користувача, рекомендацій та персоналізації продуктів, що в результаті призводить до поліпшення UX;

3) застосування штучного інтелекту дає змогу створювати продукти зі значними функціональними перевагами, як-от системи розпізнавання образів, голосові асистенти, автоматизовані процеси та багато іншого;

4) дизайнери та розробники повинні постійно адаптуватися до змін у вимогах користувачів та технологічних можливостей. Штучний інтелект допомагає реагувати на ці зміни швидше та ефективніше;

5) використання штучного інтелекту у дизайні може привести до зменшення помилок, оптимізації процесів та зниження витрат на розробку і підтримку продуктів;

6) штучний інтелект може допомогти виявляти потенційні загрози безпеці та забезпечувати захист конфіденційності даних користувачів.

Штучний інтелект і дизайн – це дві нероздільні сфери, розвиток яких відбувається паралельно з подальшим розвитком технологій. Штучний інтелект почав розвиватися в середині XX століття з появою перших концепцій і моделей машинного навчання та експертних систем. Протягом наступних десятиліть у цій галузі було досягнуто значного прогресу, зокрема появи глибоких нейронних мереж, які відкрили нові можливості для застосування штучного інтелекту в різних сферах, як-от комп'ютерний зір, природна мова та розпізнавання образів.

Водночас проєктування як процес створення та планування об’єктів чи систем також зазнало значного розвитку. Історично дизайн почав розвиватися ще в епоху промислової революції, коли його почали активно використовувати у промисловому виробництві. З часом він зазнав численних трансформацій та еволюції, відображаючи сучасні технологічні та культурні тенденції. З появою комп’ютерів і цифрових технологій у XX столітті дизайн отримав нові інструменти та можливості, що сприяли подальшому розвитку.

Сьогодні штучний інтелект і дизайн все частіше поєднуються для створення інноваційних і ефективних продуктів. Використання штучного інтелекту в дизайні дає змогу розробникам автоматизувати процеси, покращувати взаємодію з користувачем, впроваджувати інноваційні концепції та робити виробничі процеси більш ефективними. Це створює нові можливості для розвитку та інновацій у багатьох сферах, від програмного забезпечення до архітектури, від медичних технологій до мистецтва.

Штучний інтелект відіграє ключову роль у вдосконаленні процесу проєктування на всіх етапах. Наприклад, системи штучного інтелекту можуть аналізувати дані про користувачів та їх поведінку, даючи змогу дизайнерам розробляти користувацькі інтерфейси, оптимізовані для конкретних потреб і переваг. Технологія машинного навчання може автоматично генерувати концепції дизайну або прогнозувати продуктивність дизайну на основі аналізу історичних даних.

Як приклад можна назвати конструктор сайтів Wix, який працює на базі штучного інтелекту, що дає змогу за допомогою лише опису створити унікальний дизайн власного сайта.

Рис. 1. Конструктор сайтів Wix. Логотип

Приклади програм, які працюють з використанням штучного інтелекту:

- Canva – це популярний онлайн-сервіс для створення графічного дизайну. Він містить велику кількість шаблонів та елементів дизайну, які можна використовувати для створення різноманітних проєктів. Canva також використовує AI для автоматичного налаштування кольорів та компонування елементів дизайну.
- Artisto – це додаток для iOS та Android, який використовує AI для перетворення зображень та відео в мистецькі шедеври. Він дає змогу додавати різноманітні ефекти та фільтри до зображень, створюючи унікальний дизайн.
- Adobe Sensei – це набір інструментів від Adobe, який використовує штучний інтелект для створення та редагування зображень, відео та графічного дизайну. Він містить інструменти, як-от Content-Aware Fill, Auto Tone та Scene Stitching, що допомагають дизайнерам створювати складніші та креативніші проєкти.
- The Grid – це бот, який автоматично створює вебсайти на основі введених дизайнером параметрів та зображень. Він використовує AI для аналізу даних та створення унікальних макетів для кожного відвідувача. The Grid дає змогу дизайнерам зосередитись на створенні якісного контенту, а не витрачати час на технічну роботу зі створення вебсайта.

Хоча штучний інтелект допомагає виконувати багато щоденних завдань, він не може повністю замінити людський фактор у дизайні.

Дизайнери все ще відіграють важливу роль у створенні оригінального та привабливого дизайну. Ідеї та творчі методи дизайнерів є невід'ємною частиною графічного дизайну і не можуть бути повністю замінені штучним інтелектом.

Однак його використання може істотно полегшити і прискорити роботу дизайнерів, які можуть зосередитися на творчому процесі і створювати унікальні рішення [3].

Загалом штучний інтелект постійно розвивається і завойовує ринок. Останні дослідження та наукові відкриття у цій сфері показали його безкінечний потенціал. Тому можна зробити цілком логічний висновок у популярності та актуальності цього напряму.

Список використаних джерел

1. Глибовець М. М., Олецький О. В. Штучний інтелект. Київ: «Києво-Могилянська академія», 2002. 364 с.

2. Design Research: Synergies from Interdisciplinary Perspectives / J. Simonsen, J. O. Bærenholdt, M. Büscher, J. D. Scheuer. Taylor & Francis, 2010. 240 p.

3. Шаховська Н. Б., Камінський Р. М., Вовк О. Б. Системи штучного інтелекту: навч. посіб. Львів: Вид-во Львівської політехніки, 2018. 392 с.

УДК 004.45

Бєжин Є. В., *здобувач 3 курсу спеціальності 122 Комп'ютерні науки,*
Сеник І. О., *асистент кафедри інформаційних технологій*

АГЕНТНО-ОРІЄНТОВАНЕ МОДЕЛЮВАННЯ В ЕКОНОМІЧНИХ СИСТЕМАХ: ТЕОРІЯ ТА ПРАКТИКА

Донецький національний університет імені Василя Стуса, м. Вінниця

Агентно-орієнтоване моделювання (АОМ) в економічних системах є важливим інструментом для аналізу та прогнозування ринкових процесів і динаміки. Цей підхід дає змогу враховувати неоднорідність і адаптивність учасників ринку, що робить його відмінним інструментом для дослідження складних економічних систем.

Агентно-орієнтоване моделювання базується на ідеї моделювання окремих агентів, кожен з яких має власні характеристики, стратегії та взаємодіє з іншими агентами [1]. В контексті економічних систем це означає моделювання різних учасників ринку, як-от підприємства, споживачі, фінансові установи тощо. Теоретичний аналіз АОМ базується на визначенні стратегій та правил взаємодії агентів, що дає змогу прогнозувати різні сценарії розвитку економічних систем. До того ж у теоретичному аналізі вивчаються математичні моделі, які допомагають описати взаємодію агентів та встановити залежності між їх поведінкою та впливом на ринкові процеси. Це включає в себе використання моделі здійснення рі-

шень, теорії ігор, теорія оптимізації та інших методів аналізу економічної поведінки агентів.

У практичному аспекті агентно-орієнтоване моделювання включає в себе розробку комп'ютерних моделей, які відтворюють взаємодію агентів у економічній системі. Ці моделі можуть бути використані для проведення різних експериментів, симуляцій та аналізу впливу різних факторів на поведінку системи загалом. Практичний аналіз АОМ допомагає виявити ефективні стратегії управління економічними процесами та прогнозувати їх можливі наслідки [2]. До того ж у практичному аналізі використовуються реальні або симульовані дані, щоб перевірити адекватність моделей та їх здатність передбачати реальні ринкові умови. Це дає змогу аналізувати різноманітні сценарії та приймати обґрунтовані рішення на основі отриманих результатів. Також у межах практичного аналізу можуть бути використані різноманітні інструменти та підходи, як-от імітаційне моделювання, аналіз великих даних, машинне навчання та інші методи аналізу даних для отримання більш точних та надійних результатів [3].

Агентно-орієнтоване моделювання знаходить застосування в різних галузях економіки, включно з фінансами, торгівлею, макроекономікою та ін. Від аналізу ринкових тенденцій до моделювання ефектів регулюючої політики – АОМ може бути важливим інструментом для прийняття управлінських рішень та розробки стратегій у різних галузях економіки. Застосування агентно-орієнтованого моделювання у реальних умовах дає змогу аналізувати ринкові динаміки, прогнозувати ризики та приймати управлінські рішення на основі отриманих результатів [4]. Також відомості, отримані з аналізу, можуть бути використані для розробки стратегій розвитку бізнесу, оптимізації фінансових процесів та підвищення конкурентоспроможності на ринку.

Агентно-орієнтоване моделювання в економічних системах відкриває нові можливості для розуміння та управління складними ринковими процесами. Цей підхід дає змогу враховувати неоднорідність та адаптивність ринкових учасників, що допомагає розробляти більш ефективні стратегії управління економікою та прогнозувати її подальший розвиток. Розуміння та застосування принципів агентно-орієнтованого моделювання може сприяти створенню інноваційних рішень у сфері економіки та підвищити ефективність управлінських процесів у різних галузях.

Список використаних джерел

1. Тесфаціон, Л., Джадд, К. Л. (2006). Агентно-орієнтована обчислювальна економіка [Agent-Based Computational Economics]. Кембридж, Велика Британія: MIT Press.

2. Бонабо, Е. (2002). Агентне моделювання: методи та техніки моделювання людських систем [Agent-Based Modeling: Methods and Techniques for Simulating Human Systems]. Спрінгер, США.

3. Гандольфо, Дж. (2001). Економічна динаміка: методи та моделі [Economic Dynamics: Methods and Models]. Оксфорд, Великобританія: Oxford University Press.

4. Гільберт, Н. (2007). Агентні моделі [Agent-Based Models]. Лондон, Велика Британія: SAGE Publications.

УДК 519.876.5:517.55]:574.3(043.2)

Чемес В. С., *здобувач 2 курсу спеціальності 122 Комп'ютерні науки,*
Сеник І. О., *асистент кафедри інформаційних технологій*

ВИКОРИСТАННЯ ПРОСТИХ МЕТОДІВ ТА МОДЕЛЕЙ АПРОКСИМАЦІЇ ДЛЯ АНАЛІЗУ ДИНАМІКИ ПОПУЛЯЦІЙ

Донецький національний університет імені Василя Стуса, м. Вінниця

Вступ. Аналіз динаміки популяцій є важливим інструментом для розуміння та передбачення змін у розмірі та структурі популяцій у природних та робочих середовищах. Відомості про тенденції в рості або спаді популяцій можуть мати вирішальне значення для прийняття рішень у сферах екології, біології консервації, сільського господарства та інших галузях.

Апроксимація – це процес заміщення складного об'єкта або явища більш простим модельним представленням, яке дає змогу зменшити складність або обсяг обчислень, не втрачаючи водночас суттєвих характеристик оригіналу. У математиці та науці загалом апроксимація часто використовується в тих випадках, коли точне розв'язання задачі є недосяжним або дуже складним з обчислювального погляду [1, 2].

Виклад основного матеріалу. Для аналізу динаміки популяцій, крім простих методів апроксимації, використовуються й прості моделі. Дві з найпоширеніших моделей апроксимації – це модель Мальтуса та модель логістичного зростання.

Модель Мальтуса, названа на честь англійського економіста Томаса Мальтуса, є математичною моделлю, яка використовується для пояснення експоненційного росту популяції. Ця модель передбачає, що рівень зростання популяції пропорційний поточному розміру популяції. Інакше кажучи, чим більший розмір популяції, тим більше нових особин народжується (або приєднується до популяції) за одиницю часу.

Формула, що використовується для опису цієї моделі, має вигляд:

$$N_t = N_0 \cdot e^{rt},$$

де N_t – розмір популяції в момент часу;

N_0 – початковий розмір популяції в момент часу $t = 0$;

r – коефіцієнт росту, який визначає швидкість зміни популяції. Коли $r > 0$, популяція зростає; коли $r < 0$, популяція зменшується;

t – час, протягом якого відбувається зростання популяції;

e – число Ейлера, приблизно рівне 2.71828;

Модель Мальтуса є відносно простою та легкою у розумінні, що робить її важливим початковим інструментом для вивчення динаміки популяцій. Модель Мальтуса добре описує ситуації, де рівень зростання популяції пропорційний її поточному розміру. Це особливо корисно для пояснення ранніх стадій росту популяції, коли обмеження середовища ще не виявляються домінантними факторами.

Одним з основних недоліків моделі Мальтуса є відсутність урахування факторів, що обмежують ріст популяції, як-от доступність ресурсів, конкуренція за прос-

тір та їжу, вплив хижаків, хвороб тощо. Зважаючи на формулу моделі, можна помітити, що коефіцієнт росту r вважається постійним, що не завжди відповідає реальній ситуації, де цей коефіцієнт може змінюватися з часом через різні умови [3].

Модель логістичного зростання є математичною моделлю, яка враховує обмеження середовища та інші фактори, що впливають на зростання популяції. Порівняно з моделлю Мальтуса, яка передбачає необмежений експоненційний ріст, модель логістичного зростання враховує, що популяція не може безмежно зростати через обмежену доступність ресурсів та інші обмеження середовища.

У моделі логістичного зростання розмір популяції (N) залежно від часу (t) описується такою формулою:

$$P(t) = \frac{KP_0e^{rt}}{K + P_0(e^{rt} - 1)},$$

де $P(t)$ – розмір популяції в момент часу t;

K – максимальна місткість середовища (тобто максимальний розмір популяції, який середовище може підтримувати);

P_0 – початковий розмір популяції;

r – межа, до якої може зростати популяція, відома як ємність середовища або максимальна популяція, за якою стабілізується ріст;

t – час, протягом якого відбувається зростання популяції;

e – число Ейлера, приблизно рівне 2.71828 [4];

Розглянемо приклад. Уявімо, що ми досліджуємо популяцію кролів на острові. Дано початкову кількість кролів $(P0) = 50$. Через 6 місяців $(t = 6)$ кількість кролів $(P(6)) = 150$, а через 12 місяців $(t = 12)$ кількість кролів $(P(12)) = 300$.

Необхідно знайти максимальну ємність середовища (K), швидкість зростання популяції (r) та кількість популяції через два роки $(P(24))$.

Для розв'язання використовуємо метод найменших квадратів для знаходження параметрів K та r, які мінімізують суму квадратів різниць між фактичними значеннями популяції та значеннями, отриманими за допомогою логістичної функції.

Виконувавши розрахунки, отримаємо такі результати: $K \approx 450, r \approx 0.231$.

Використовуючи формулу логістичного зростання, утворюємо нову таблицю з даними, які вказують на кількість кролів у кожен місяць, та графік, який показує зростання (табл. 1, рис. 1).

Таблиця 1 – Кількість кролів у кожен місяць

t	$N(t)$
0	50
1	61,22519
2	74,50111
3	89,9894
4	107,7735
5	127,8242
6	149,9706
7	173,8831
8	199,0784

t	$N(t)$
9	224,9503
10	250,8235
11	276,0227
12	299,9411
13	322,095
14	342,1541
15	359,947
16	375,444
17	388,7281

t	$N(t)$
18	399,9607
19	409,3495
20	417,1215
21	423,5037
22	428,7103
23	432,935
24	436,3481

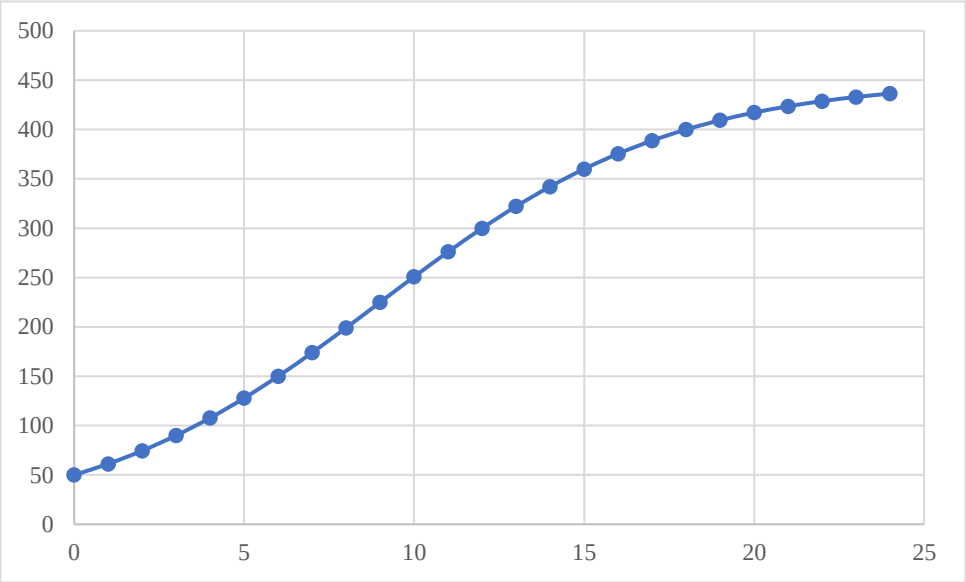


Рис. 1. Модель логістичного зростання

Висновки. Аналіз динаміки популяцій є важливим для розуміння та передбачення змін у розмірі та структурі популяцій у природних та робочих середовищах. Апроксимація, процес заміщення складних моделей простішими, як – от модель Мальтуса та модель логістичного зростання, допомагає спростити аналіз, зберігаючи водночас суттєві характеристики оригіналу. Ці моделі дають змогу передбачати тенденції в рості або спаді популяцій, враховуючи обмеження середовища та інші фактори.

Список використаних джерел

1. Approximation and Estimation. *StudySmarter*. URL: https://www.studysmarter.co.uk/explanations/math/pure-maths/approximation-and-estimation/ (дата звернення: 15.05.2024).

2. Linear and Quadratic Approximation. *CoCalc*. URL: https://cocalc.com/share/public_paths/a2c4095c79cfd0ff50a71047808a7a565dae2933 (дата звернення: 15.05.2024).

3. Wenner J. M. Teaching Exponential Growth and Decay. Geology Department, University of Wisconsin-Oshkosh. *Serc*. 09.01.2006. URL: https://serc.carleton.edu/quantskills/methods/quantlit/expGandD.html (дата звернення: 16.05.2024).

4. Exponential & logistic growth. *KhanAcademy*. URL: https://www.khanacademy.org/science/ap-biology/ecology-ap/population-ecology-ap/a/exponential-logistic-growth (дата звернення: 17.05.2024).

УДК: 004.8:519.2

Бевзюк А. Ю., *здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,*
науковий керівник:
Якубич К. О., *асистент кафедри інформаційних технологій*

АПРОКСИМАЦІЯ ФУНКЦІЙ У МАШИННОМУ НАВЧАННІ: ПОРІВНЯННЯ ЛІНІЙНОЇ ТА НЕЛІНІЙНОЇ РЕГРЕСІЇ ДЛЯ КЛАСИФІКАЦІЇ ДАНИХ

Донецький національний університет імені Василя Стуса, м. Вінниця

У сучасному світі обробка даних та машинне навчання стали ключовими в технологічних застосуваннях. Машинне навчання – галузь штучного інтелекту, дає змогу комп'ютерам навчатися на основі даних та виконувати завдання без явного програмування. Останні десятиліття воно стало ключовим складником сфери інформаційних технологій, впливаючи на різноманітні галузі, як-от фінанси, медицина, наука про дані та ін. [1]. Фундаментальним аспектом машинного навчання є апроксимація функцій, що полягає у створенні моделей, які наближено описують поведінку вхідних даних, зокрема зв'язок між незалежними змінними та цільовою змінною. У цій статті порівнюються два основні методи апроксимації функцій: лінійна та нелінійна регресія.

Лінійна регресія – це метод у машинному навчанні, який використовує лінійну функцію для апроксимації залежностей між вхідними та вихідними даними. Вона широко використовувався протягом багатьох років завдяки своїй простоті, інтерпретації та ефективності. Це цінний інструмент для розуміння зв'язків між змінними та прогнозування в різноманітних програмах. Математично це можна представити так:$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \varepsilon,$

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \varepsilon,$$

де y – цільова змінна, $x_1,\ x_2, \ldots, x_n$ – вхідні ознаки, $\beta_1, \beta_2, \ldots, \beta_n$ – коефіцієнти регресії, ε – похибка [2].

Методи нелінійної регресії дають змогу більш гнучкий підхід, оскільки вони враховують нелінійні взаємозв'язки між змінними та цільовою змінною [3]. Порівняно з лінійною регресією, нелінійні моделі можуть краще апроксимувати складніші залежності в даних, роблячи їх більш ефективними у деяких випадках. Однак ця перевага може призвести до перенавчання, особливо якщо кількість параметрів у моделі велика. Один із поширених підходів – це використання поліноміальної регресії, де відносини між змінними та цільовою змінною моделюються у вигляді поліноміальної функції. Математично це можна виразити так:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_n x^n + \varepsilon,$$

де y – цільова змінна, $x, x^2, x^3, \ldots, x^n$ – це квадрат, куб та інші степені вхідних ознак, $\beta_1, \beta_2, \ldots, \beta_n$ – коефіцієнти регресії, ε – похибка.

Для порівняння лінійної та нелінійної регресії розглянемо завдання класифікації електронних листів на спам та не спам. Для цього ми використаємо рандом-

ний набір даних, що містить різні характеристики електронних листів та відповідні мітки класу.

Відповідно маємо такі дані:

- Набір даних складається з n прикладів, де кожен приклад характеризується певними властивостями електронного листа.
- Кожному прикладу поставлена у відповідність мітка класу, де 0 відповідає не спаму, а 1 – спаму.

За допомогою інтерактивного середовища для числових обчислень та програмування MATLAB, ми спробували реалізувати код для розв'язання цієї задачі (рис. 1).

```
% Генерація випадкових даних для електронних листів
n = 300; % кількість прикладів
x = linspace(0,10,n)'; % характеристики електронних листів
y = 0.5*x + 1 + randn(n,1); % мітки класу (0 - не спам, 1 - спам)

% Лінійна регресія
X_linear = [ones(n,1), x];
beta_linear = X_linear\y;
y_linear = X_linear*beta_linear;

% Нелінійна регресія (метод опорних векторів з ядром RBF)
svm_model = fitrsvm(x,y,'KernelFunction','rbf'); % навчання моделі
y_svm = predict(svm_model, x); % прогнозовані значення

% Розрахунок середньоквадратичної помилки для лінійної регресії
mse_linear = mean((y - y_linear).^2);
% Розрахунок середньоквадратичної помилки для нелінійної регресії
mse_svm = mean((y - y_svm).^2);

% Вивід результатів
fprintf('Середньоквадратична помилка для лінійної регресії: %.4f\n', mse_linear);
fprintf('Середньоквадратична помилка для нелінійної регресії: %.4f\n', mse_svm);

% Графіки
figure;
scatter(x,y, 'filled');
hold on;
plot(x,y_linear,'r','LineWidth',2); % лінійна регресія
plot(x,y_svm,'g','LineWidth',2); % нелінійна регресія
xlabel('Характеристика');
ylabel('Мітка класу');
legend('Дані','Лінійна регресія','Нелінійна регресія');
title('Класифікація електронних листів: Лінійна vs Нелінійна регресія');
```

Рис. 1. Реалізація задачі

Даний код створює різні моделі для класифікації електронних листів на спам та не спам, а також оцінює їх ефективність. Спочатку генерується набір випадкових даних, які відображають характеристики електронних листів. Під час цього враховується, що деякі електронні листи можуть бути класифіковані як спам, тоді як інші – як не спам. Далі використовується два різні підходи до класифікації, спочатку лінійна регресія, яка апроксимує залежність між характеристиками електронного листа та його класом. Далі втілений нелінійний підхід, використовуючи

метод опорних векторів з ядром RBF, щоб змоделювати більш складні залежності між характеристиками та класами [2].

Після навчання обох моделей ми оцінюємо їх ефективність, обчислюючи середньоквадратичну помилку для кожної. Це дає змогу нам порівняти, наскільки добре кожна модель передбачає класи електронних листів.

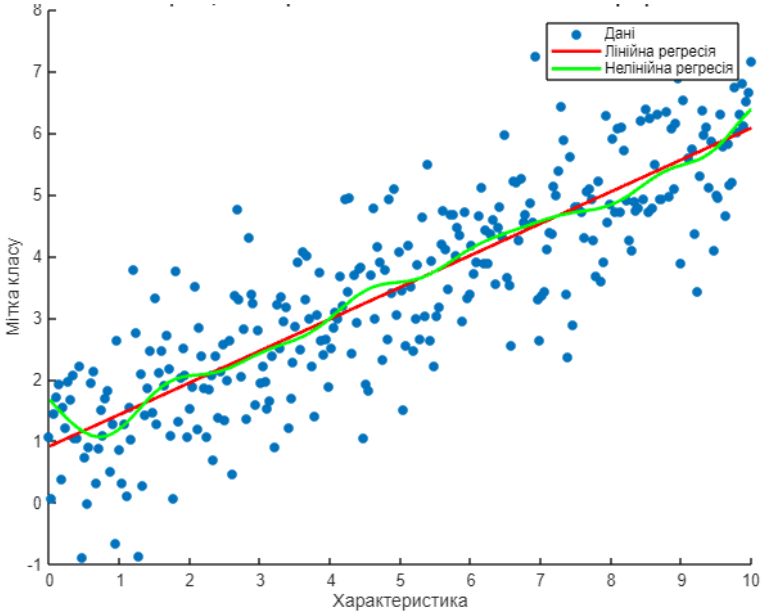


Рис. 2. Графік, який відображає лінійну та нелінійну регресію

На рис. 1 ми можемо бачити результат роботи програми, а саме побудований графік, який візуалізує як вихідні дані, так і прогнозовані значення для кожної моделі. Це дає нам змогу краще зрозуміти, як кожен підхід впорався з класифікацією електронних листів.

```
Середньоквадратична помилка для лінійної регресії: 0.8487
Середньоквадратична помилка для нелінійної регресії: 0.8411
```

Рис. 3. Результати, виведені в консоль

Також у консоль виводяться результати, які описують середньоквадратичну помилку для лінійної та нелінійної регресії. Це допомагає нам порівняти їх точність та ефективність у вирішенні задачі класифікації електронних листів на спам та не спам. Загалом отримані нами результати свідчать про те, що обидві моделі можуть бути використані для класифікації електронних листів на спам та не спам, проте нелінійна регресія може бути трохи ефективнішою в цьому випадку.

Отже, у машинному навчанні вибір між лінійною та нелінійною регресією залежить від складності та природи даних, а також від конкретної задачі, що вирішується. Хоча лінійна регресія проста і легко інтерпретується, вона може бути неефективною для моделювання нелінійних зв’язків. Нелінійні методи регресії, навпаки, забезпечують більш гнучкий підхід і можуть точніше відображати складні зв’язки в даних.

Список використаних джерел

1. Smola A., Vishwanathan S. V. N. Introduction to Machine Learning: Cambridge University Press, 2008. 234 p. URL: https://alex.smola.org/drafts/thebook.pdf (дата звернення: 17.04.2024).

2. Кононова К. Ю. Машинне навчання: методи та моделі: підручник для бакалаврів, магістрів та докторів філософії спеціальності 051 «Економіка». Харків: ХНУ імені В. Н. Каразіна, 2020. 301 с. URL: https://comsys.kpi.ua/upload/Rainforcement%20Learning%20.pdf (дата звернення: 17.04.2024).

3. What Is Nonlinear Regression? *Nonlinear Regression.* URL: https://ch.mathworks.com/discovery/nonlinear-regression.html (дата звернення: 17.04.2024).

4. Метод інтерполяції для прогнозування метрик використання хмарних обчислень в статистичному навчанні / Н. А. Потапова, Л. О. Волонтир, І. П. Частоколенко, М. С. Григоренко. *Наука і техніка сьогодні.* 2024. № 4(32). С. 1192–1205.

УДК 519.6:004.42:004.043

Козачок А. О., *здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,*
науковий керівник:
Хмелівський Ю. С., *асистент кафедри інформаційних технологій*

РОЛЬ МЕТОДІВ ОБЧИСЛЕНЬ У ВИРІШЕННІ ЗАДАЧ АНАЛІЗУ ДАНИХ

Донецький національний університет імені Василя Стуса, м. Вінниця

У сучасному світі обробка та аналіз даних є важливим кроком у прийнятті рішень у різних сферах діяльності, зокрема в бізнесі, науці, медицині та соціальних науках. Масштаби зібраної інформації стрімко зростають і потребують відповідних методів обробки та аналізу. Роль обчислювальних методів у вирішенні завдань аналізу даних полягає в їх здатності ефективно обробляти великі обсяги інформації та забезпечувати точність, швидкість і достовірність отриманих результатів [1].

Важливим аспектом обчислювальних методів у вирішенні задач аналізу даних є їх внесок у створення та розвиток алгоритмів обробки даних. Обчислювальні методи, як-от машинне навчання, статистичний аналіз і обробка сигналів, можуть допомогти створити алгоритми, здатні розпізнавати образи і виявлення кореляцій у великих наборах даних. Наприклад, алгоритми класифікації даних, як-от опорні векторні машини та нейронні мережі, можуть відрізнити дані від одного класу до іншого [2].

Другий аспект методів обчислень полягає в їх здатності працювати з великими обсягами даних. Традиційні методи аналізу даних, як-от ручне кодування чи статистичні методи, можуть бути неефективними під час обробки великих обсягів даних через обмежену швидкість та ресурси. Методи обчислень, які базуються на паралельних обчисленнях або використанні великих обчислювальних кластерів, можуть значно прискорити аналіз даних, забезпечуючи швидкий доступ до інформації та високу продуктивність обробки.

На прикладі (рис. 1) показано, як простий метод аналізу даних може значно полегшити життя в сучасному світі, допомагаючи вирішувати навіть складні завдання. Приміром, шляхом аналізу великого обсягу медичних даних можна розробити прості моделі для прогнозу можливого ризику серцево-судинного захворювання у пацієнтів. Це дає змогу вчасно виявляти особливості, що можуть вка-

зувати на загрозу здоров'ю, і вживати відповідних заходів для попередження ускладнень. Такий підхід сприяє покращенню якості медичного обслуговування та збереженню людських життів.

Лістинг програми оцінки ризику серцево-судинних захворювань наведено на рис. 1–2.

```
import urllib.request
# URL для завантаження набору даних про серцево-судинні захворювання
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.cleveland.data"
# Запит на сервер для отримання даних
response = urllib.request.urlopen(url)
# Перевірка статусу відповіді
if response.getcode() == 200:

    data = response.read().decode('utf-8')
    data_lines = data.splitlines()

    # Проаналізуємо дані та прогнозуємо ризик для кожного пацієнта
    for line in data_lines:
        patient_data = line.split(",")
        age = float(patient_data[0])
        sex = float(patient_data[1])
        blood_pressure = int(float(patient_data[3]))

        # Прогнозуємо ризик захворювання на основі віку та артеріального тиску
        if age > 50 and blood_pressure > 140:
            risk = "High"
        else:
            risk = "Low"
```

Рис. 1. Лістинг програми оцінки ризику серцево-судинних захворювань. Частина 1

```
        # Виводимо результати аналізу для кожного пацієнта
        print(f"Пацієнт: Вік={age}, Стать={sex}, Артеріальний тиск={blood_pressure}")
        print(f"Ризик серцево-судинного захворювання: {risk}")
        print("="*50)  # Для розділення результатів між пацієнтами
else:
    print("Помилка отримання даних:", response.getcode())
```

Рис. 2. Лістинг програми оцінки ризику серцево-судинних захворювань. Частина 2

Результат роботи цієї програми наведено на рис. 3.

```
Пацієнт: Вік=63.0, Стать=1.0, Артеріальний тиск=145
Ризик серцево-судинного захворювання: High
==================================================
Пацієнт: Вік=67.0, Стать=1.0, Артеріальний тиск=160
Ризик серцево-судинного захворювання: High
==================================================
Пацієнт: Вік=67.0, Стать=1.0, Артеріальний тиск=120
Ризик серцево-судинного захворювання: Low
==================================================
Пацієнт: Вік=37.0, Стать=1.0, Артеріальний тиск=130
Ризик серцево-судинного захворювання: Low
==================================================
Пацієнт: Вік=41.0, Стать=0.0, Артеріальний тиск=130
Ризик серцево-судинного захворювання: Low
```

Рис. 3. Результати роботи програми оцінки ризику серцево-судинних захворювань

Обчислювальні методи дають змогу покращити валідацію та тестування моделей аналізу даних. Широкий спектр обчислювальних методів допомагає проводити різні експерименти з даними, перевіряти різні гіпотези та визначати найбільш ефективний підхід до аналізу набору даних. Це підвищує достовірність результатів аналізу та робить процес прийняття рішень більш обґрунтованим [3].

Можна зазначити, що роль методів обчислень у вирішенні задач аналізу даних є критичною та незамінною. Сучасні підходи до обробки даних ґрунтуються на їх використанні, і вони відіграють ключову роль у всіх аспектах аналізу даних. Методи обчислень допомагають виявляти закономірності, які важливі для прийняття обґрунтованих рішень у різних сферах життя, від медицини до економіки. До того ж вони стимулюють розвиток нових напрямів досліджень у сфері аналізу даних, що відкриває нові можливості для вирішення складних проблем та досягнення нових знань. Тому важливо продовжувати розвивати та вдосконалювати методи обчислень, щоб забезпечити подальший прогрес в аналізі даних та в їх практичному застосуванні.

Список використаних джерел

1. Опрацювання даних як інформаційний процес. *Основні поняття з інформатики. UA5.org.* URL: https://ua5.org/osnovi/2167-opraczyuvannya-danyh-yak-informaczijnyj-proczes.html

2. Сем Фокс. OpenAI – нові можливості для бізнесу – розумовий аналіз даних та виявлення патернів. *Mediacom: світ новин.* 30.03.2024. URL: https://mediacom.com.ua/openai-rozumovij-analiz-danix-ta-viyavlennya-paterniv-novi-mozhlivosti-dlya-biznesu/

3. Верес О. М. Класифікація методів аналізу великих даних. *Вісник Національного університету «Львівська політехніка». Серія: Інформаційні системи та мережі.* Львів: Видавництво Львівської політехніки, 2017. № 872. С. 84–92.

УДК 519.2

Сапожнікова В. Є., *здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,*
Сеник І. О., *асистент кафедри інформаційних технологій*

МЕТОД НАЙМЕНШИХ КВАДРАТІВ

Донецький національний університет імені Василя Стуса, м. Вінниця

У галузі статистики часто виникає потреба знайти зв'язок між змінними, і метод найменших квадратів є широко використовуваною технікою для підгонки математичних моделей до даних. Завдяки своїй універсальності та широкому спектру застосувань він відіграє важливу роль у розв'язанні різноманітних задач.

Цей метод використовується для знаходження найкращої прямої (або деколи кривої), що представляє взаємозв'язок між цими змінними. Англійською мовою лінія має назву *least squares line,* що означає лінія найменших квадратів, або *regression line,* тобто лінія регресії, чи також *best fit line*, що перекладається як найбільш підходяща лінія; до прикладу, це може бути $y = mx + b$ (або $y = b_0 + b_1 x$).

Головна мета цього методу – зменшити суму квадратів помилок настільки, наскільки це можливо, що і є причиною назви [1].

Треба розглянути наочний приклад задля кращого розуміння версатильності методу. Уявімо, що нам потрібно знайти параметри лінійного рівняння, що найкраще задовольняють такі дані:

Таблиця 1 – Дані

$\boldsymbol{x}$	1	2	3	4	5	6	7
$\boldsymbol{y}$	1,5	3,8	6,7	9,0	11,2	13,6	16

Перевіривши дані, можна побачити, що точки не лежать на одній лінії, тобто необхідно знайти таку пряму, щоб кожне задане умовою значення знаходилось якнайближче до неї.

Перший пункт виконання завдання – визначення шуканих параметрів. Для знаходження задовільного рівняння прямої $y = mx + b$ потрібно скористатися такими формулами:

$$m = \frac{n\sum xy - \sum x\sum y}{[n\sum x^2 - (\sum x)^2]}; \qquad b = \frac{\sum y - m\sum x}{n},$$

де n – кількість заданих точок.

З формул робимо висновок, що спочатку треба знайти показник m, який у статистиці буде також мати позначення b_1. Для цього заповнюємо таблицю:

Таблиця 2 – Дані, необхідні для підрахунку m

$\boldsymbol{x}$	$\boldsymbol{y}$	$\boldsymbol{xy}$	$\boldsymbol{x^2}$
1	1.5	1.5	1
2	3.8	7.6	4
3	6.7	20.1	9
4	9	36	16
5	11.2	56	25
6	13.6	81.6	36
7	16	112	49

Також необхідно знайти суму кожного стовпчика таблиці для подальшої калькуляції тому, порахувавши, ми визначили, що $\sum x = 28, \sum y = 61.8$, $\sum xy = 61.8$, $\sum x^2 = 140$. Також занотовуємо, що $n = 7$.

Тепер підставляємо значення у попередньо вказані формули.

$$m = \frac{n\sum xy - \sum x\sum y}{[n\sum x^2 - (\sum x)^2]} = \frac{7(314.8) - (28)(61.8)}{7(140) - (28)^2} = 2.4142857;$$

$$b = \frac{\sum y - m\sum x}{n} = \frac{(61.8) - 2.4142857(28)}{7} = -0.828571.$$

Округлюючи значення і підставляючи їх у рівняння лінійної функції, отримуємо $y = 2.41x - 0.83$, що і є шуканою відповіддю.

Ми можемо переглянути точність виконання, підставивши початкові дані в отриману пряму. Візьмемо декілька випадкових аргументів для перевірки:

$$y = 2.41(2) - 0{,}83 = 3.99;$$

$$y = 2.41(5) - 0{,}83 = 11.22;$$

$$y = 2.41(2) - 0{,}83 = 16.04.$$

Отримані числа є дуже хорошими апроксимаціями, адже вони неймовірно близькі до вхідних.

Для вирішального переконання варто проілюструвати усі дані.

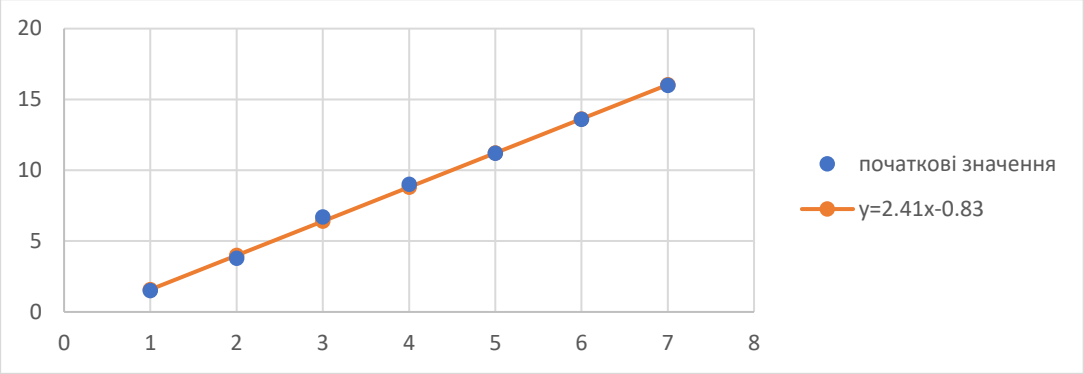


Рис. 1. Зображення початкових і отриманих даних

Тепер остаточного можна впевнитись, що виконані обчислення зроблені правильно.

Одне з головних використань методу найменших квадратів – регресійний аналіз. Він передбачає вивчення взаємозв'язку між залежною та однією або кількома незалежними змінними. Хоча значення методу найменших квадратів може виходити далі його застосування в регресійному аналізі.

Концепція мінімізації суми квадратів різниць між спостережуваними та прогнозованими значеннями виходить далеко за межі статистики. Вона знаходить застосування у різних галузях, наприклад, обробка сигналів, машинне навчання та навіть квантова механіка [2].

Ще одним важливим застосуванням методу найменших квадратів є апроксимація кривих, що дає змогу знаходити найкращу криву, яка проходить через набір точок. Цей підхід широко використовується, коли дані мають нелінійну природу і потребують більш складної моделі для точного відображення залежності між змінними. Наприклад, поліноміальна апроксимація кривих дає змогу створити математичну модель, яка мінімізує суму квадратів відхилень між фактичними та розрахованими значеннями [3]. Такий підхід є надзвичайно корисним у фізиці, біології, економіці та інших галузях, де складні залежності можна описати за допомогою поліномів, експонент або логарифмів.

Цей метод також був використаний у галузі геодезії для визначення найкращої сфери, що представляє форму Землі [4]. Це застосування передбачає складний математичний процес, який враховує безліч геодезичних вимірювань для створення моделі, що мінімізує суму квадратів різниць між спостережуваними та розрахованими значеннями. Також він використовувався для оцінки параметрів у цій же сфері, як-от гравітаційне поле та форма планети [4].

До того ж метод найменших квадратів знайшов своє місце у сфері обробки сигналів, де його також використовують для оцінки параметрів моделі сигналу, щоб найкраще відповідати спостережуваним даним. Це застосування є особливо важливим у таких галузях, як-от телекомунікації, радіолокація та сонар, де точна оцінка параметрів сигналу є необхідною для ефективного аналізу даних [4].

Ці різноманітні застосування та теоретичні основи підкреслюють метод найменших квадратів як потужний та універсальний інструмент, що проникає в різні дисципліни, стаючи незамінною технікою для аналізу даних та підгонки моделей.

Висновки. Отже, метод найменших квадратів пропонує надійний підхід до підгонки математичних моделей до даних. Зводячи до мінімуму суму квадратів різниць між спостережуваними та прогнозованими значеннями, він не лише забезпечує спосіб кількісної оцінки відповідності, але й дає змогу ідентифікувати закономірності та тенденції в даних. Загалом метод найменших квадратів є фундаментальною технікою в області аналізу та моделювання даних.

Список використаних джерел

1. Least Square Method: вебсайт. URL: https://www.cuemath.com/data/least-squares/
2. Chakraborty S., Morolia A., Peduri A. Quantum Regularized Least Squares. 2023. Vol. 7. 988 p. URL: https://quantum-journal.org/papers/q-2023-04-27-988/
3. Seber G. A. F., Wild C. J. Nonlinear Regression. John Wiley & Sons. 2003. URL: https://onlinelibrary.wiley.com/doi/book/10.1002/0471725315
4. Eshagh M. The Earth's Gravity Field Role in Geodesy and Large-Scale Geophysics. 23 October 2020. URL: https://www.intechopen.com/chapters/76431

УДК: 004.58:004.8:007.5

Сивак Д. І., *здобувач 2 курсу спеціальності 122 Комп'ютерні науки,*
Комаров В. Ф., *канд. техн. наук, старший викладач кафедри інформаційних технологій*

РОЗВИТОК ТЕХНОЛОГІЙ МАШИННОГО НАВЧАННЯ ДЛЯ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ТА БЕЗПЕКИ АВТОНОМНИХ ТРАНСПОРТНИХ ЗАСОБІВ

Донецький національний університет імені Василя Стуса, м. Вінниця

У сучасному світі автомобільна промисловість швидко рухається в напрямі автономності, переходячи від традиційних автомобілів до безпілотних систем. Зростання інтересу до автономних автомобілів спонукає нас до розуміння та розробки комплексу технологій, спрямованих на підвищення ефективності та безпеки транспортних систем. Одним із ключових компонентів цього переходу є використання технік машинного навчання для розвитку інтелектуальних систем керування безпілотними автомобілями.

В сучасних безпілотних автомобілях широко використовуються різноманітні методи машинного навчання для забезпечення автономної роботи. Декілька основних методів включають:

1. Нейронні мережі: зокрема, згорткові нейронні мережі (CNN) та рекурентні нейронні мережі (RNN) широко використовуються з технологією глибокого навчання для розпізнавання об'єктів на дорозі, виявлення дорожніх знаків та інших дорожніх об'єктів, а також для прогнозування руху інших учасників дорожнього руху [1].

2. Навчання з підкріпленням: використовуються для навчання автономних автомобілів приймати рішення в реальному часі на основі взаємодії з довкіллям. Ці методи дають змогу автомобілям навчатися від своїх власних дій і отримувати відповідний зворотний зв'язок для постійного покращення рішення.

3. Метод обробки природної мови (NPL): Використання методів NPL дає змогу автономним автомобілям аналізувати та розуміти голосові команди водіїв, комунікації з іншими автомобілями та системами керування рухом, а також обробляти текстову інформацію, що надходить від зовнішніх джерел.

Практичні застосування машинного навчання в сучасних безпілотних автомобілях включають: системи прогнозування та управління трафіком, які допомагають автономним автомобілям прогнозувати рух інших транспортних засобів і оптимізувати свій маршрут для уникнення заторів та небезпечних ситуацій; системи виявлення перешкод, де автономні автомобілі використовують машинне навчання для виявлення та класифікації перешкод на дорозі, як-от інші автомобілі, пішоходи, велосипедисти, тварини тощо; виявлення та уникнення небезпечних ситуацій на дорозі; автоматичну діагностику та обслуговування технічних систем автомобілів; автоматичне управління, де машинне навчання застосовується для керування автомобілем, зокрема керування динамікою руху, що забезпечує оптимальне та безпечне переміщення автомобіля по дорозі [2].

Тож системи на основі технологій штучного інтелекту можуть виявляти проблеми та несправності в роботі автомобіля, передбачати майбутні витрати на обслуговування й ремонт, а також надавати рекомендації щодо підтримки оптимальної ефективності та надійності автомобіля.

Також вони можуть покращити комфорт. Наприклад, системи автоматичного управління можуть оптимізувати маршрути для забезпечення максимальної швидкості та економії пального, а системи розпізнавання голосу можуть надавати пасажирам можливість взаємодіяти з автомобілем безпосередньо та зручно.

З розвитком алгоритмів та моделей машинного навчання з'являються нові можливості для безпілотних автомобілів. Важливою перспективою є вдосконалення систем навігації та розпізнавання об'єктів. Розвиток нових методів візуального сприйняття, як-от обробка зображень та використання сенсорів, допоможе безпілотним автомобілям ефективніше реагувати на навколишнє середовище та уникати небезпек [3].

Оскільки дорожнє середовище постійно змінюється, безпілотні автомобілі повинні бути здатні адаптуватися до цих змін. Розробка механізмів адаптації до змінних умов дорожнього руху є ключовим аспектом для забезпечення ефективності та безпеки безпілотних автомобілів.

Проте використання машинного навчання в безпілотних автомобілях стикається з низкою технічних обмежень та вимог до апаратного забезпечення. Серед них висока обчислювальна потужність, оскільки алгоритми машинного навчання, особливо глибокі нейронні мережі, потребують значних обчислювальних ресурсів не тільки для навчання, а й для виконання в реальному часі. До того ж автономність транспортних засобів суттєво залежить від раціонального споживання ресурсів встановленими на них системами, що може ускладнювати використання потужних моделей машинного навчання.

І оскільки проблема реакції автоматичних систем керування на непередбачувані ситуації на дорозі не має універсального рішення через неможливість прогнозу таких ситуацій, лишаються відкритими етичні питання безпеки та відповідальності під час використання безпілотних систем і людино-машинної взаємодії на дорогах, коли треба визначити ролі виробника автомобіля, програмістів, власника транспортного засобу або інших сторін [4].

У підсумку, розвиток технологій машинного навчання для безпілотних автомобілів відкриває безліч можливостей для покращення ефективності та безпеки на дорозі. Інтеграція новітніх алгоритмів та моделей машинного навчання, вдосконалення систем навігації та розпізнавання об'єктів, розробка механізмів адаптації до змінних умов дорожнього руху та підвищення рівня безпеки і надійності автономних систем є ключовими напрямами розвитку. Підвищення рівня безпеки та надійності автономних транспортних засобів може бути досягнуто вдалим втіленням адаптивних систем, здатних ефективно реагувати на непередбачувані ситуації. Розглянуті технології мають потенціал змінити обличчя транспортної системи, забезпечуючи більш безпечне, зручне та стабільне транспортне сполучення для як для окремої людини, так і для суспільства загалом.

Список використаних джерел

1. Ucar A., Karakose M., Kırımça N. Artificial Intelligence for Predictive Maintenance Applications: Key Components, Trustworthiness, and Future Trends. Appl. Sci. 2024. № 14. С. 898. URL: https://www.mdpi.com/2076-3417/14/2/898

2. Learning-Based Safe Control for Robot and Autonomous Vehicle Using Efficient Safety Certificate / H. Zheng, C. Chen, S. Li, S. Zheng. Jenuary 2023. URL: https://www.researchgate.net/publication/371142694_Learning-Based_Safe_Control_for_Robot_and_Autonomous_Vehicle_using_Efficient_Safety_Certificate

3. Application of Machine learning Algorithms in Autonomous Vehicles Navigation System / G. D. Suriya Prasath, M. K. Rahgul Poopathi, P. Sarvesh, A. Samuel. Sep. 2020. 912 с. URL: https://www.researchgate.net/publication/344870766_Application_of_Machine_learning_Algorithms_in_Autonomous_Vehicles_Navigation_System

4. Anthony C., Elgenaidi W., Rao M. Intrusion Detection System for Autonomous Vehicles Using Non-Tree Based Machine Learning Algorithms. 2024. № 13. 809 с. URL: https://www.mdpi.com/2079-9292/13/5/809

УДК: 004.9

Сивак Д. І., *здобувач 2 курсу спеціальності 122 Комп'ютерні науки,*
науковий керівник:
Якубич К. О., *асистент кафедри інформаційних технологій*

ПОРІВНЯННЯ МЕТОДУ ТРАПЕЦІЇ ТА МЕТОДУ СІМПСОНА

Донецький національний університет імені Василя Стуса, м. Вінниця

В аналізі даних і обчисленнях чисельні методи інтегрування, як-от метод трапеції та метод Сімпсона, відіграють ключову роль у вирішенні інтегральних задач. Переважно задача полягає в отриманні наближених значень визначених

інтегралів, необхідних для подальшого аналізу даних, моделювання фізичних явищ або розв'язання математичних задач. Порівняння методів трапеції та Сімпсона дає змогу визначити їх переваги та недоліки з метою вибору найбільш підходящого методу для конкретної задачі. Такий аналіз допомагає підвищити ефективність обчислень, забезпечити потрібну точність результатів і вибрати оптимальний підхід до розв'язання задачі залежно від її умови та вимог.

Метод трапеції та метод Сімпсона – це два чисельні методи інтегрування, використовувані для наближеного обчислення визначених інтегралів.

Метод трапеції полягає в розділенні підінтегральної області на невеликі трапеції, апроксимації площі кожної трапеції та підсумовуванні цих площ для отримання наближеного значення інтегралу. Однак необхідно зазначити, що цей метод може мати обмежену точність для деяких типів функцій та може залежати від вибору кроку інтегрування [1].

Метод Сімпсона ґрунтується на розділенні підінтегральної області на рівномірні частини, апроксимації кожної з них параболою та обчисленні площі цих парабол за допомогою формули Сімпсона. Він зазвичай має вищу точність та швидшу збіжність, але може бути складнішим у реалізації та більш чутливим до вибору кроку.

Метод трапеції є чисельним методом інтегрування, що ґрунтується на апроксимації підінтегральної області відрізками прямих ліній, утвореними з вершин функції та осі x. Відрізок інтегрування розділяється на рівні частини, і для кожної з них обчислюється площа трапеції, утвореної відповідно частиною графіка функції, віссю x та прямими, які з'єднують кінці відрізка з графіком. Загальна апроксимація підінтегральної області здійснюється шляхом сумування площ усіх трапецій [2]. Формула для обчислення:

$$\int_{x_{i-1}}^{x_i} f(x)dx \approx \frac{h}{2} * (f(x_0) + 2\sum_{i=1}^{n-1} f(x_i) + f(x_n)).$$

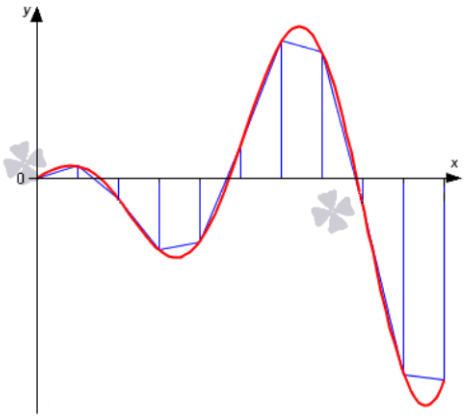


Рис. 1. Графічна ілюстрація методу трапеції

Метод Сімпсона – це чисельний метод інтегрування, що базується на апроксимації підінтегральної області за допомогою парабол. Відрізок інтегрування розділяється на рівні частини, і для кожної з них обчислюється площа параболи, що проходить через три точки: початкову, кінцеву та середню. Загальна апроксимація підінтегрованої області здійснюється шляхом сумування площ усіх парабол [3].

Формула для обчислення:

$$\int_a^b f(x)dx \approx \frac{h}{3}(f(x_0) + 4\sum_{i=1}^{n} f(x_{2i-1}) + 2\sum_{i=1}^{n-1} f(x_{2i}) + f(x_{2n})).$$

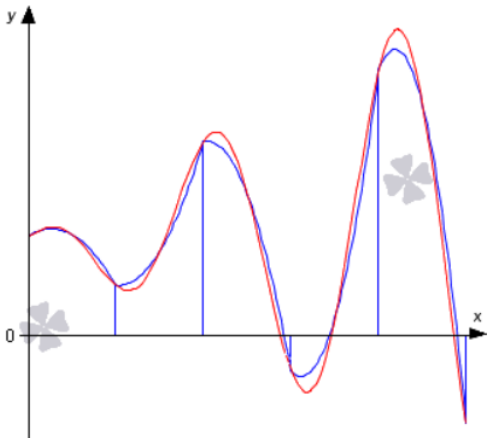


Рис. 2. Графічна ілюстрація методу Сімпсона

Таблиця 1 – Переваги та недоліки методу трапеції та методу Сімпсона

	Переваги	**Недоліки**
Метод трапеції	1. Простота реалізації. 2. Універсальність. 3. Достатня точність	1. Низька швидкість збіжності. 2. Обмеження точність для деяких типів функцій. 3. Залежність від вибору кроку
Метод Сімпсона	1. Висока точність. 2. Швидка збіжність. 3. Універсальність	1. Складність реалізації. 2. Вимоги до гладкості функцій. 3. Чутливість до вибору кроку

Під час порівняння методів трапеції та Сімпсона можна зробити такі висновки:

Метод Сімпсона зазвичай є більш точним та швидшим, порівняно з методом трапеції, особливо для гладких функцій з простими формами. Висока точність та швидка збіжність роблять метод Сімпсона перевагою у багатьох випадках, особливо коли потрібно отримати точні результати з мінімальною кількістю обчислень.

Проте варто враховувати, що метод Сімпсона може бути складнішим у реалізації та вимагати більше обчислювальних ресурсів, а також він може давати менш точні результати для деяких складних функцій, особливо якщо функція має швидко змінюваний характер. Тому вибір між методами повинен залежати від конкретних вимог задачі та характеристик функції, яку потрібно інтегрувати.

Список використаних джерел

1. Верещак Р. Метод Трапецій: Основи та Практичне Використання. *Чисельне диференціювання та інтегрування.* 25.02.2024. URL: https://www.mathros.net.ua/obchyslennja-vyznachenyh-integraliv-metodom-trapecij.html

2. Мальцевская И. Метод трапеций. *Интегралы, методы интегрирования.* 14.04.2023. URL: https://zaochnik.com/spravochnik/matematika/integraly-integrirovanie/metod-trapetsij/

3. Мальцевская И. Метод Симпсона (парабол). *Интегралы, методы интегрирования.* 29.05.2023. URL: https://zaochnik.com/spravochnik/matematika/integraly-integrirovanie/metod-simpsona-parabol/

УДК 004.01

Куцмай В. Я., *здобувач 3 курсу спеціальності 122 Комп'ютерні науки,*
Ніколюк П. К., *д-р фіз.-мат. наук, професор, професор кафедри інформаційних технологій*

РОЗВ'ЯЗАННЯ ЗАДАЧ ОПТИМІЗАЦІЇ ЗА ДОПОМОГОЮ МЕТОДУ СИМПЛЕКСУ

Донецький національний університет імені Василя Стуса, м. Вінниця

Вступ. В умовах сучасного розвитку математичного програмування та збільшення складності задач оптимізації метод симплексу є одним з найпопулярніших та найефективніших методів для розв'язання задач лінійного програмування. Застосування методу симплексу дає змогу знаходити оптимальні рішення для широкого спектра економічних, інженерних та управлінських завдань. У цій роботі ми досліджуємо основні принципи та алгоритми методу симплексу, а також розглядаємо його застосування на прикладах реальних задач.

Метод симплексу, розроблений Джорджем Данцигом у 1947 році, базується на ітеративному підході до знаходження оптимального розв'язку задачі лінійного програмування. Цей метод використовує поняття базисного розв'язку, що відповідає вершині багатогранника, який представляє множину допустимих рішень. Процес симплексу полягає в переході від однієї вершини до іншої уздовж ребер багатогранника до досягнення оптимального рішення.

Основні етапи методу симплексу включають:

1. Формулювання початкового базисного розв'язку.
2. Оцінка оптимальності поточного базисного розв'язку.
3. Вибір напряму покращення (вибір вхідних та вихідних змінних).
4. Оновлення базисного розв'язку та повторення процесу до досягнення оптимального рішення.

Застосування методу симплексу. Завдяки своїй ефективності метод симплексу широко застосовується в різних галузях:

– економіка: оптимізація виробничих планів, розподіл ресурсів, максимізація прибутку;

– логістика: планування маршрутів, мінімізація витрат на транспортування, оптимізація складування;

– планування виробництва: визначення оптимального обсягу випуску продукції, мінімізація витрат;

– управління запасами: оптимізація обсягу запасів, мінімізація витрат на зберігання.

Наприклад, у логістиці метод симплексу використовується для розв'язання задач оптимального розподілу ресурсів, планування маршрутів та мінімізації витрат на транспортування. Це дає змогу компаніям ефективно управляти своїми ресурсами та підвищувати продуктивність.

Важливим аспектом застосування методу симплексу є його реалізація у вигляді програмного забезпечення. Сьогодні існує безліч програмних продуктів, як-

от LINDO, CPLEX, MATLAB, що дають змогу ефективно використовувати метод симплексу для розв'язання складних задач з оптимізації. Використання цих програм забезпечує швидкість і точність розрахунків, що є критично важливим для сучасних досліджень та бізнес-процесів.

- LINDO: надає інструменти для розв'язання задач лінійного, нелінійного та цілочислового програмування.
- CPLEX: широко використовується для розв'язання задач оптимізації в промисловості та дослідженнях.
- MATLAB: забезпечує потужні можливості для числових розрахунків, зокрема розв'язання задач лінійного програмування за допомогою симплексу.

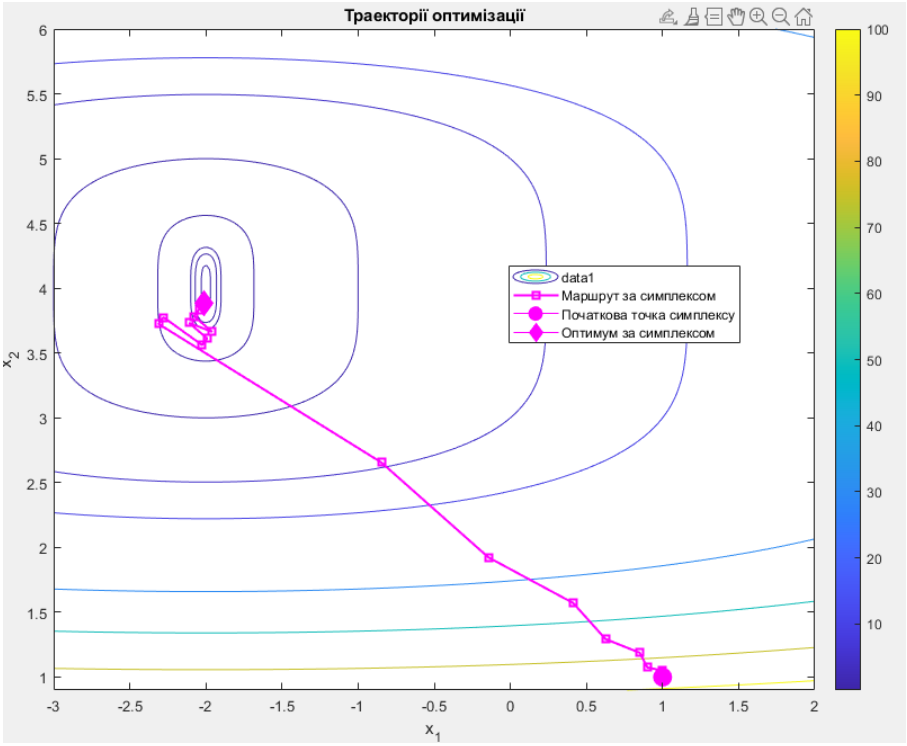


Рис. 1. Траєкторія оптимізації за Симплекс-методом з використанням MatLab

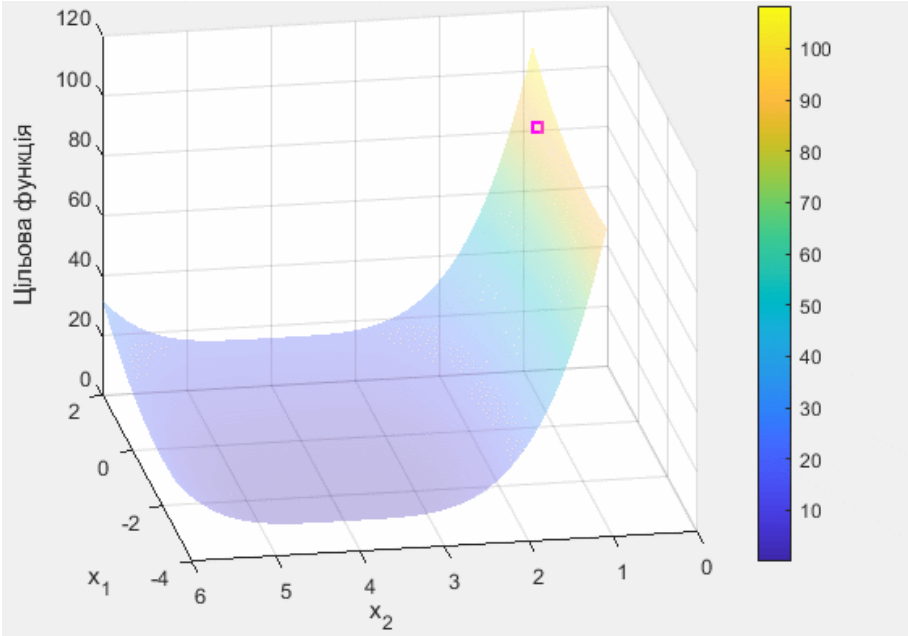


Рис. 2. Анімація маршруту за Симплекс-методом

Висновки. Отже, метод симплексу є потужним інструментом для розв'язання задач лінійного програмування, що дає змогу знаходити оптимальні рішення в різних галузях. Його застосування сприяє підвищенню ефективності управлінських та виробничих процесів, що робить його невід'ємною частиною сучасного математичного програмування.

Список використаних джерел

1. John N. Tsitsiklis. Introduction to Linear Optimization by Dimitris Bertsimas. URL: https://www.amazon.com/Introduction-Linear-Optimization-Scientific-Computation/dp/1886529191
2. Taha H. A. Operations Research: An Introduction. URL: https://zalamsyah.staff.unja.ac.id/wp-content/uploads/sites/286/2019/11/9-Operations-Research-An-Introduction-10th-Ed.-Hamdy-A-Taha.pdf
3. Bazaraa M. S., Jarvis J. J., Sherali H. D. Linear Programming and Network Flows. URL: https://industri.fatek.unpatti.ac.id/wp-content/uploads/2019/03/006-Linear-Programming-and-Network-Flow-Mokhtar-S.-Bazaraa-John-J.-Jarvis-Hanif-D.-Sherali-Edisi-4-2010.pdf
4. Сайт MATLAB. URL: https://www.mathworks.com/products/matlab.html
5. Сайт LINDO. URL: https://www.lindo.com/
6. Сайт CPLEX. URL: https://www.ibm.com/products/ilog-cplex-optimization-studio

УДК 004.6

Гапоянц Д. В., *здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,*
науковий керівник:
Горяшин А. С., *асистент кафедри інформаційних технологій*

АНАЛІЗ ТА УПРАВЛІННЯ ПОХИБКАМИ В МЕТОДАХ ОБЧИСЛЕНЬ

Донецький національний університет імені Василя Стуса, м. Вінниця

Методи обчислень є важливим інструментом для вирішення задач різноманітних галузей науки: економіки, фізики, математики, географії тощо. Їх використання є надзвичайно ефективним на практиці, адже вони дають змогу проводити розрахунки у тих ситуаціях, коли аналітичний точний результат поставленого завдання отримати важко або ж взагалі не можливо. Невід'ємною частиною цього процесу є виникнення певної похибки обчислень – величини, яка характеризує точність результату.

Залежно від джерела виникнення розрізняють такі види похибок:

1) неусувні похибки;
2) похибки методу;
3) похибки обчислень;
4) повна похибка.

Неусувні похибки можуть бути пов'язані з:

– некоректністю вхідних даних;
– невідповідністю математичної моделі.

Неусувні похибки виникають у разі неточності вхідних даних, що можуть бути спричинені помилками вимірювальних приладів, випадковими відхиленя-

ми, помилками людини під час введення даних та ін. Іншою причиною їх появи є вибір математичної моделі, що не відповідає суті явища, її спрощення, використання припущень, які не мають нічого спільного з реальними умовами.

Похибки методу є результатом використання наближених методів замість точних. Вони зазвичай можуть бути врегульовані, залежно від конкретного методу, зміною його параметрів, встановленням іншого кроку інтегрування тощо. Ще однією можливою причиною їх виникнення може бути перетворення неперервного процесу на дискретний, в такому разі ми маємо справу з похибками дискретизації.

Похибки обчислень пов'язані з заокругленням вхідних та вихідних даних для проведення подальших операцій чисельних методів.

Повною похибкою називають сумарну кількість похибок, що виникли під час розв'язання певної задачі.

Ще одною класифікацією похибок є їх розділення на абсолютні та відносні. ***Абсолютна похибка*** визначається як модуль різниці між точним A та наближеним a значеннями числа: $\Delta = |a - A|$. Ця похибка вимірюється в тих самих одиницях, що і вихідне значення. Формула ***відносної похибки:*** $\sigma = \frac{\Delta}{|a|}$. Проведення обрахунку абсолютної та відносної похибки дає ширше уявлення про її загальну величину.

Метод обчислення ми можемо назвати стійким, коли похибка обчислень протягом виконання обрахунків не перевищує заданих значень. Якщо ж незначна похибка вхідних даних суттєво впливає на кінцевий результат – метод називають нестійким. До стійких методів можна віднести методи Рунге–Кутта, метод трапецій, метод центральних різниць. Прикладом нестійкого методу може слугувати метод прямокутників, адже у разі великих кроків метод може давати значні похибки, оскільки не враховує зміни функції на інтервалі інтегрування.

Задля мінімізації неусувних похибок необхідно покращити якість вхідних даних, забезпечивши максимально можливу точність вимірювань та правильно підібравши математичну модель, що гарно відображає природу досліджуваного процесу.

Похибки методу можна мінімізувати різними способами, одним з них є зменшення кроку дискретизації, якщо це доцільно для даної задачі. Задля досягнення меншого впливу похибки треба підбирати стійкі методи з вищим порядком точності. Таким є, наприклад, метод Рунге–Кутта.

Не менш важливим способом для мінімізації похибок є детальний аналіз математичної моделі і проведення ретельного вивчення процесу, що дасть змогу ідентифікувати критичні точки, в яких виникнення похибки є найімовірнішим.

Також управляти похибками можна через використання адаптивних методів, які автоматично змінюють крок дискретизації залежно від локальної поведінки функції. Наприклад, адаптивні методи Рунге–Кутта змінюють крок залежно від оцінок похибки на кожному кроці.

Отже, можна впливати на різні види похибок, як-от неусувні похибки, похибки методу, похибки обчислень. Ключовими засобами для цього є забезпечення максимальної точності початкових даних, проведення детального дослідження

предметної області, вдале визначення математичної моделі, а також вибір відповідного до нашої задачі методу, який дасть змогу обчислити найбільш точний результат.

Список використаних джерел

1. Чисельні методи: навчальний посібник / Л. О. Волонтир, О. В. Зелінська, Н. А. Потапова, І. А. Чіков. Вінниця: ВНАУ. 2020 322 с.

2. Задачин В. М., Конюшенко І. Г. Чисельні методи: навчальний посібник. Харків: Вид. ХНЕУ ім. С. Кузнеця, 2014. 180 с.

3. Гончаров О. А., Васильєва Л. В., Юнда А. М. Чисельні методи розв'язання прикладних задач: навч. посіб. Суми: Сумський державний університет, 2020. 142 с.

УДК 004.6

Гончар А. А., *здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,*
науковий керівник:
Волонтир Л. О., *старший викладач кафедри інформаційних технологій*

СУТНІСТЬ ІНТЕРПОЛЯЦІЇ В ОЦІНЦІ ВИПАДКОВИХ ПРОЦЕСІВ

Донецький національний університет імені Василя Стуса, м. Вінниця

Випадкові процеси, або стохастичні процеси, є послідовностями випадкових змінних, що представляють системи або явища, які розвиваються невизначеним способом з часом. Точне оцінювання цих процесів має велике значення в різних наукових та інженерних галузях. Інтерполяція, метод побудови нових точок даних у межах діапазону дискретного набору відомих точок, є необхідною для перетворення дискретних спостережень у неперервні оцінки, що дає змогу більш детального та тонкого аналізу випадкових процесів.

Інтерполяцію можна визначити математично так. Маємо набір точок даних $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$, мета – знайти функцію $f(x)$, таку, що: $f(x_i) = y_i$ для $i = 0, 1, \dots, n$.

У контексті стохастичних процесів інтерполяція використовується для оцінки відсутніх або невідомих значень процесу, згладжування шумових даних та відновлення сигналів з дискретних зразків. Вона заповнює прогалину між теоретичними моделями та реальними спостереженнями, підвищуючи точність оцінюваного процесу. Методи інтерполяції:

– **Лінійна інтерполяція:** найпростіша форма інтерполяції, де інтерполяційна функція є прямою лінією між кожною парою точок даних:

$$f(x) = y_i + \frac{(x - x_i)}{(x_{i+1} - x_i)}(y_{i+1} - y_i) \text{ для } x_i \le x \le x_{i+1}.$$

Лінійна інтерполяція часто використовується через свою простоту та обчислювальну ефективність. Проте вона може не захопити тонкощів підлягаючого випадкового процесу, особливо якщо процес демонструє значну нелінійність.

– **Поліноміальна інтерполяція:** використовує поліном ступеня n для проходження через $n + 1$ точок даних:

$$f(x) = \sum_{i=0}^{n} y_i \prod_{\substack{0<j<n \\ j\neq i}} \frac{x - x_j}{x_i - x_j}.$$

Поліноміальна інтерполяція, хоча і більш гнучка, ніж лінійна, страждає від феномену Рунге, коли на краях інтервалу можуть виникати осциляції. Ця проблема особливо критична для поліномів високого ступеня.

– **Сплайнова інтерполяція:** використовує часткові поліноми, зазвичай кубічні сплайни, які забезпечують гладкість у точках даних:

$$f(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 \text{ для } x_i \leq x \leq x_{i+1}.$$

Сплайнова інтерполяція, зокрема кубічні сплайни, забезпечує баланс між гнучкістю та гладкістю. Підхід з частковими поліномами зменшує проблеми осциляції, що виникають у поліноміальній інтерполяції високого ступеня, і забезпечує гладку неперервну криву, яка точно слідує за точками даних.

– **Кригінг** є методом інтерполяції, що враховує просторову кореляцію між даними. Використовується коваріаційна матриця C, що описує залежність між точками. Оцінка значення у точці x здійснюється як зважена сума відомих значень:

$$\hat{Z}(x) = \sum_{i=1}^{n} \lambda_i Z(x_i),$$

де коефіцієнти λ_i обираються так, щоб мінімізувати дисперсію оцінки:

$$\sum_{j=1}^{n} \lambda_j C(x_i, x_j) = C(x_i, x).$$

Важливою сферою застосування інтерполяції є обробка сигналів, де відновлення безперервного сигналу з дискретних зразків є ключовим завданням. Тут методи інтерполяції, як-от інтерполяція синк та сплайнова інтерполяція, знаходять широке застосування. У фінансовому моделюванні інтерполяція використовується для оцінювання відсутніх даних у часових рядах цін активів, процентних ставок та інших економічних показників.

До того ж інтерполяція важлива в геостатистиці, де методи кригінгу використовуються для просторової інтерполяції даних, як-от прогнозування концентрацій корисних копалин та забруднювачів навколишнього середовища.

В екологічному моніторингу техніки інтерполяції незамінні для оцінки рівнів забруднення, прогнозування екологічних тенденцій та оцінки впливу людської діяльності. Приклади демонструють застосування методів інтерполяції у картографуванні якості повітря, забруднення води та забруднення ґрунту, допомагаючи приймати аргументовані рішення у сфері управління довкіллям та збереження.

Інтерполяція відіграє важливу роль у медичних застосуваннях зображення, як-от магнітно-резонансна томографія (MRI) та комп'ютерна томографія (CT).

Високоякісна реконструкція зображень залежить від точних алгоритмів інтерполяції для заповнення відсутніх точок даних та підвищення роздільної здатності. Новаторські методи інтерполяції, адаптовані для медичних зображень, як-от тензорна інтерполяція та техніки суперроздільності, сприяють поліпшенню діагностики та плануванню лікування у сфері охорони здоров'я.

Незважаючи на значний прогрес у розвитку методів інтерполяції, залишається кілька викликів, як-от обробка розріджених та нерегулярно вибраних даних, зменшення артефактів інтерполяції та вирішення обчислювальної складності для масштабних застосувань. Майбутні дослідження можуть зосередитися на розробці адаптивних алгоритмів інтерполяції, які динамічно адаптуються до змінних патернів даних, та на включенні доменної інформації для поліпшення продуктивності інтерполяції в спеціалізованих застосуваннях.

Список використаних джерел

1. Чисельні методи: навчальний посібник / Л. О. Волонтир, О. В. Зелінська, Н. А. Потапова, І. А. Чіков. Вінниця: ВНАУ. 2020 322 с.

2. Numerical Recipes: The Art of Scientific Computing / W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery. Cambridge University Press. 2007. URL: https://assets.cambridge.org/97805218/80688/frontmatter/9780521880688_frontmatter.pdf

3. Hastie T., Tibshirani R., Friedman J. Елементи статистичного навчання: видобування даних, висновки та прогнозування. *Springer*. 2009. URL: https://hastie.su.domains/Papers/ESLII.pdf

УДК 004.01

Олійник Б. С., *здобувач 1 курсу*
ОС «Магістр»
спеціальності 122 Комп'ютерні науки,
Волонтир Л. О., *старший викладач*
кафедри інформаційних технологій

АНАЛІЗ ДАНИХ ФІНАНСОВИХ РИНКІВ ЗА ДОПОМОГОЮ ІНФОРМАЦІЙНИХ СИСТЕМ

Донецький національний університет імені Василя Стуса, м. Вінниця

У сучасному фінансовому середовищі, яке характеризується високою волатильністю і складністю, аналіз даних фінансових ринків стає вирішальним для прийняття ефективних інвестиційних рішень. Інформаційні системи, які здатні збирати, обробляти та аналізувати великі обсяги фінансових даних, стають необхідними інструментами для трейдерів, аналітиків і фінансових менеджерів.

З розвитком фінансових ринків і збільшенням обсягів даних виникає необхідність у створенні інформаційних систем, які здатні швидко і точно аналізувати ці дані. Використання таких систем дає змогу підвищити ефективність прийняття рішень, знизити ризики і підвищити прибутковість інвестицій.

Сучасні дослідження в галузі аналізу даних фінансових ринків зосереджені на використанні методів машинного навчання та штучного інтелекту. Наприклад, методи глибокого навчання дають змогу прогнозувати динаміку ринків на основі

історичних даних. До того ж значна увага приділяється аналізу новин і соціальних медіа для виявлення настроїв ринку.

Інформаційна система для аналізу даних фінансових ринків здатна оперативно обробляти великі обсяги даних і надавати аналітичні звіти для підтримки прийняття інвестиційних рішень.

Основні завдання, які необхідно вирішувати під час проєктування цієї системи:

1) розробка архітектури інформаційної системи аналізу даних;
2) вибір та впровадження методів обробки і аналізу даних;
3) інтеграція джерел даних, зокрема ринкові дані, новини і соціальні медіа;
4) створення інтерфейсу користувача для зручного доступу до аналітичних звітів.

Розроблена інформаційна система включає в себе модулі для збору, обробки і аналізу даних. Для аналізу використовуються методи машинного навчання, як-от нейронні мережі і регресійні моделі. Система інтегрується з різними джерелами даних і надає можливість аналізувати як історичні дані, так і поточні ринкові умови. Інтерфейс користувача допомагає отримувати аналітичні звіти в реальному часі, що значно підвищує ефективність прийняття рішень.

Інформаційна система аналізу даних фінансових ринків є важливим інструментом для сучасних фінансових аналітиків. Вона дає змогу підвищити точність прогнозів, знизити ризики і приймати обґрунтовані інвестиційні рішення. Подальші дослідження можуть бути зосереджені на покращенні алгоритмів аналізу даних і розширенні функціональності системи.

Список використаних джерел

1. Іваненко О. Аналіз фінансових ринків з використанням методів машинного навчання. Київ: Наукова думка, 2020. С. 65–67.
2. Петров І., Коваленко М. Прогнозування динаміки ринку на основі історичних даних. Харків: Видавництво ХНУ, 2019. С. 31–33.
3. Сидоренко О. Аналіз настроїв ринку за допомогою соціальних медіа. Львів: ЛНУ імені Івана Франка, 2018. С. 145–151.

УДК 004.9

Калько Д. Р., *здобувач 2 курсу спеціальності 122 Комп'ютерні науки,*
науковий керівник:
Хмелівський Ю. С., *асистент кафедри інформаційних технологій*

ТЕОРЕТИЧНІ АСПЕКТИ ПІДХОДІВ ЧИСЕЛЬНОГО ДИФЕРЕНЦІЮВАННЯ В АНАЛІЗІ ДАНИХ

Донецький національний університет імені Василя Стуса, м. Вінниця

Чисельне диференціювання є важливим інструментом у наукових дослідженнях та інженерних розрахунках, особливо коли аналітичне знаходження похідних є складним або неможливим. Метод чисельного диференціювання дає

змогу отримувати наближені значення похідних функцій на основі їх дискретних значень. Сучасні програмні засоби, як-от MATLAB, Python та R, забезпечують зручні інструменти для реалізації чисельних методів та обробки великих обсягів даних. Це дає змогу дослідникам швидко і точно обчислювати похідні, що є важливим етапом у багатьох наукових та інженерних дослідженнях.

Сучасні дослідження зосереджені на підвищенні точності та стабільності чисельних методів. Дослідження також спрямовані на застосування чисельного диференціювання у нових галузях, як-от машинне навчання та аналіз великих даних.

Основу методів чисельного диференціювання становить обчислення кінцевих різниць. Цей метод має кілька варіантів, зокрема прямі, зворотні та центральні різниці. Прямі та зворотні різниці використовуються для обчислення похідних першого порядку, тоді як центральні різниці забезпечують більш точні результати завдяки використанню середніх значень. Однак всі ці методи мають певні обмеження щодо точності та стабільності, особливо під час обчислення похідних вищих порядків [1].

Метод кінцевих різниць першого порядку можна записати як:

$$f'(x) \approx \frac{f(x+h) - f(x)}{h},$$

де h – малий крок.

Для підвищення точності обчислень використовується метод центральних різниць:

$$f''(x) \approx \frac{f(x+h) - f(x-h)}{2h}.$$

Інший підхід до чисельного диференціювання базується на використанні інтерполяційних поліномів. Метод Лагранжа та метод Ньютона дають змогу обчислювати похідні інтерполяційних поліномів, що забезпечує більш точні результати, порівнянно з методами кінцевих різниць. Однак, складність цих методів зростає зі збільшенням кількості точок, що може призвести до втрати точності через ефект Рунге [2].

Поліном Лагранжа першого порядку виглядає так:

$$P_1(x) = \frac{(x - x_1)}{(x_0 - x_1)} f(x_0) + \frac{(x - x_0)}{(x_1 - x_0)} f(x_1),$$

де x_0 та x_1 – інтерполяційні вузли.

Метод Ньютона для обчислення похідних базується на використанні скінченних різниць. Наприклад, друга похідна функції може бути обчислена за допомогою центральної різниці другого порядку:

$$f''(x) \approx \frac{(x+h) - 2f(x) + f(x-h)}{h^2}.$$

Чисельне диференціювання знаходить широке застосування у багатьох галузях науки та техніки. Наприклад, у фізиці та механіці чисельні методи використовуються для аналізу динамічних систем, моделювання руху та обчислення силових характеристик. У біології та медицині ці методи застосовуються для обробки експериментальних даних та моделювання біологічних процесів. До того ж чисельне диференціювання є ключовим інструментом у машинному навчанні та

аналізі даних, де воно використовується для оптимізації функцій та обчислення градієнтів.

В інженерних розрахунках чисельне диференціювання використовується для розв'язання задач теплопровідності, гідродинаміки та багатьох інших. Наприклад, обчислення градієнтів температури у задачах теплопередачі або швидкостей у задачах механіки рідин є важливим етапом для визначення розподілу температур та швидкостей у різних матеріалах і середовищах. У цих випадках точність чисельних методів має вирішальне значення для отримання коректних результатів[3].

Отже, чисельне диференціювання є важливим інструментом для аналізу та обробки даних у багатьох галузях науки й техніки. Використання сучасних програмних засобів та алгоритмів дає змогу підвищити точність та ефективність обчислень, що сприяє розвитку нових технологій та методів досліджень.

Список використаних джерел

1. Burden R. L., Faires J. D. Numerical Analysis. Brooks Cole, 2010. 888 p.
2. Press W. H., Teukolsky S. A., Vetterling W. T., Flannery B. P. Numerical Recipes: The Art of Scientific Computing. Cambridge University Press, 2007. 1235 p.
3. Chapra S. C., Canal R. P. Numerical Methods for Engineers. McGraw-Hill, 2015. 542 p.

СЕКЦІЯ 5
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНІ ТЕХНОЛОГІЇ

УДК 004.01

Чайковський П. А., *здобувач*
1 курсу ОС «Магістр»
спеціальності 122 Комп'ютерні науки,
Штовба С. Д., *д-р техн. наук, професор,*
професор кафедри
інформаційних технологій

ПІДХІД ДО АВТОМАТИЧНОЇ КЛАСИФІКАЦІЇ ТА ПРІОРИТЕЗАЦІЇ ТЕКСТОВИХ ПОВІДОМЛЕНЬ ЗА ДОПОМОГОЮ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ

Донецький національний університет імені Василя Стуса, м. Вінниця

Інформаційне перевантаження є значною проблемою сучасного суспільства. Користувачі щодня отримують великий обсяг повідомлень з різних джерел, що часто призводить до втрати важливої інформації. Автоматична класифікація з подальшою пріоритезацією текстових повідомлень може значно покращити керування інформаційними потоками. Наприклад, користувач отримує багато електронних листів щодня. Під автоматичною класифікацією мається на увазі, що система аналізує вхідні листи і сортує їх за категоріями: робочі листи, особисті повідомлення, реклама тощо, використовуючи підходи до машинного навчання та обробки природної мови, як-от використання LLM (Large Language Model). Потім на основі налаштувань користувача система пріоритезує ці листи, щоб найважливіші повідомлення (наприклад, робочі завдання) з'являлися у видачі першими. Також користувач зацікавлений у автоматично сформованому звіті за деякий часовий інтервал, у якому буде узагальнено зміст ланцюжка повідомлень.

У цій роботі пропонується підхід до використання великих мовних моделей (LLM) для автоматичної класифікації та пріоритезації текстових повідомлень. Він складається з послідовного виконання таких чотирьох етапів.

Збір даних – інтеграція з різними джерелами для збору текстових повідомлень. За допомогою API-інтерфейсів можна забезпечити безперервний потік даних з месенджерів, соціальних мереж та електронної пошти [1]. Наприклад, використання JSON Mode для обробки даних забезпечує більш ефективний збір та форматування даних [6].

Аналіз та класифікація – використання LLM для ідентифікації важливості, терміновості та тематики повідомлень. Наприклад, метод in-context learning може ефективно класифікувати текстові дані, використовуючи великі мовні моделі, як показано у дослідженнях [2, 3]. CARP (Clue And Reasoning Prompting) є інноваційним підходом, який використовує поетапне знаходження підказок та діагностичне міркування для покращення класифікації тексту [4].

Пріоритезація – сортування повідомлень за важливістю та налаштуваннями користувача. Це дає змогу користувачам отримувати тільки найбільш релевантну інформацію, уникаючи перевантаження другорядними повідомленнями [5]. Користувач матиме змогу використати особисті запити у текстовому вигляді, або автоматично згенеровану систему пріоритетів, на основі історичних даних. Для

виконання цього етапу будуть використані результати виконання попереднії пунктів.

Підсумовування – узагальнення ключової інформації за деякий період (день, тиждень). Використання методів синтетичної генерації даних допомагає створювати якісні тренувальні набори даних, що покращують продуктивність класифікації та підсумовування повідомлень [3, 4].

У системі управління повідомленнями можна використовувати LLM для аналізу вхідних електронних листів. За допомогою методів in-context learning модель може визначати тематику та важливість кожного повідомлення, класифікуючи їх за категоріями «робочі завдання», «особисті листи», «сповіщення» тощо. Використовуючи JSON Mode для передачі даних між компонентами системи, можна забезпечити зручний формат обміну інформацією, що спрощує інтеграцію різних джерел даних [6].

Реалізація запропонованого підходу дасть змогу користувачам зосередитися на найважливіших повідомленнях та уникнути перевантаження інформацією. Передбачається, що ефективність підходу буде перевірено експериментально на реальних даних. Очікується, що використання великих мовних моделей значно покращить точність і швидкість класифікації повідомлень, як це відбувалось в експериментах за схожими тематиками [1, 2, 4].

Список використаних джерел

1. Text Classification via Large Language Models / X. Sun et al. *Findings of the Association for Computational Linguistics: EMNLP 2023.* 2023. DOI: 10.18653/v1/2023.emnlp-main.122.

2. Synthetic Data Generation with Large Language Models for Text Classification: Potential and Limitations / Z. Li et al. *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing.* 2023. URL: https://aclanthology.org/2023.emnlp-main.143

3. Task-Level Thinking Steps Help Large Language Models for Challenging Classification Task / C. Du et al. *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing.* 2023. URL: https://aclanthology.org/2023.emnlp-main.105

4. Text Classification via Large Language Models. Papers With Code. 2023. URL: https://paperswithcode.com/paper/text-classification-via-large-language-models

5. A Comprehensive Overview of Large Language Models. 2023. URL: https://arxiv.org/abs/2307.06435

6. Text Generation Guide: JSON Mode. OpenAI Documentation. *OpenAI.* URL: https://platform.openai.com/docs/guides/text-generation/json-mode

УДК 004.01

Костенко Р. О., *здобувач 3 курсу спеціальності 122 Комп'ютерні науки,*
Ніколюк П. К., *д-р фіз.-мат. наук, професор, професор кафедри інформаційних технологій*

РОЗРОБКА АНДРОЇД-ДОДАТКА SNAPTRACK

Донецький національний університет імені Василя Стуса, м. Вінниця

Зважаючи на швидкий технологічний прогрес та розповсюдження мобільних пристроїв, соціальні мережі стають не просто платформами для спілкування, а важливим інструментом для організації та планування нашого життя. У цьому контексті особливо важливим стає забезпечення максимальної зручності та ефективності взаємодії з іншими користувачами. Відтак актуальним є розвиток мобільних додатків, які дають змогу відстежувати місцезнаходження та спілкуватися з друзями і знайомими.

З огляду на цей контекст і зростання потреб користувачів у зручних інструментах спілкування та організації подій розробка мобільного додатка, що об'єднує можливості соціальної мережі, чату та віртуальної карти місцезнаходження друзів, є надзвичайно актуальною. Метою цієї роботи є розробка андроїд-додатка під назвою «SnapTrack», який буде поєднувати в собі можливості соціальної мережі, чату та віртуальної карти місцезнаходження користувачів з додатковими функціями історії подорожей.

У розробці додатка SnapTrack використовується Clean Architecture, яка дає змогу розбити додаток на окремі модулі та шари, що забезпечує його гнучкість та легкість розширення. Цей допомагає дає змогу виділити різні компоненти додатка, як-от Domain, Presentation та Data шари, і забезпечити їх незалежність один від одного. Для шару Presentation додатка використовується MVVM (Model-View-ViewModel) – архітектура, що дає змогу розділити бізнес-логіку та відображення даних.

Основною ідеєю Clean Architecture є створення програмного забезпечення, яке складається з незалежних від конкретних реалізацій деталей компонентів, що входять у додаток. Це досягається шляхом поділу програми на рівні, кожен з яких відповідає за певні аспекти функціональності та бізнес-логіку [1]. MVVM-архітектура дає змогу розділити логіку бізнес-логіки та відображення даних у додатку, що полегшує його розробку та підтримку. Використання цього патерну допомагає забезпечити чистий та ефективний код, а також полегшує тестування окремих компонентів програми [2].

Загалом використання Clean Architecture та MVVM-підходу викликає значні переваги, трансформуючи код додатка в більш структурований, читабельний та підтримуваний. Це сприяє полегшенню процесу розробки, підвищенню якості коду та швидкому впровадженню нових функцій і змін у додаток SnapTrack.

Однак вибір оптимального стеку технологій для додатка SnapTrack є також ключовим етапом розробки, оскільки він впливає на продуктивність, швидкість

розробки і майбутню розширюваність. Важливі складники технологічного стеку включають мову програмування Kotlin, яка забезпечує безпеку та продуктивність коду, бібліотеку Hilt для управління залежностями, та Firebase для роботи з хмарними сервісами Google. Додаток також використовує компоненти Jetpack, як-от LiveData та Flow, для реалізації реактивного програмування, а також Navigation для управління навігацією між екранами. Використання Single Activity Architecture спрощує структуру додатка, зменшуючи складність коду та полегшуючи управління навігацією.

Загальний вибір технологій базується на принципах ефективності, продуктивності та масштабованості, що дає змогу забезпечити високу якість та ефективність додатка, а також полегшить процес розробки та підтримки.

Успіх андроїд-додатків, як-от SnapTrack, визначається можливістю масштабування і продуктивністю. Використання Firebase разом з Clean Architecture забезпечує миттєве масштабування, а Kotlin та Hilt підвищують продуктивність команди розробників. LiveData та Flow допомагають у покращенні реактивного програмування, забезпечуючи ефективну взаємодію з даними та реальний час оновлення інтерфейсу. Цей технологічний стек створює основу для масштабних і продуктивних андроїд-додатків, що легко відповідають потребам користувачів і розширюються в майбутньому [3].

У додатку SnapTrack ми використовуємо Firebase для забезпечення безпеки та конфіденційності даних користувачів. Firebase, розроблений Google, є потужною та надійною платформою для цілей мобільних додатків. Використовуючи сервіси, як-от Firebase Authentication, Realtime Database та Firebase Storage, ми гарантуємо захист персональних даних користувачів. Firebase Authentication дає змогу автентифікувати користувачів шляхом різних методів, забезпечуючи зашифрований обмін даними. Realtime Database та Firebase Storage допомагають зберігати дані в реальному часі та завантажувати файли, забезпечуючи їх безпечний обмін завдяки протоколу HTTPS. До того ж Firebase дає змогу налаштовувати такі права, що лише авторизовані користувачі мають доступ до даних. Інструменти моніторингу безпеки, як-от Firebase Security Rules Simulator та Firebase Performance Monitoring, допомагають виявляти й виправляти можливі проблеми безпеки та продуктивності. Загалом використання Firebase у SnapTrack гарантує високий рівень захисту даних, сприяючи підтримці довіри користувачів та успішному функціонуванню додатка [4].

Отже, один з можливих сценаріїв використання додатка полягає у тому, щоб військові могли користуватися ним для ведення комунікації та обміну важливою інформацією під час бойових операцій. Наприклад, вони можуть використовувати його для обміну координатами місцезнаходження, передачі повідомлень про стратегічні об’єкти або подачі сигналів про небезпеку. Завдяки захищеному зберіганню даних у Firebase інформація буде захищена від несанкціонованого доступу, що є критично важливим для успішної реалізації місій. До того ж SnapTrack може бути використаний для орієнтування та навігації у складних територіальних умовах. Військові можуть використовувати його для планування маршрутів, визначення оптимальних шляхів переміщення та взаємозв’язку з іншими підрозділами. Забезпечення безпеки даних через Firebase дасть змогу впевнено користуватися додатком у реальному часі, не турбуючись про можливі ризики.

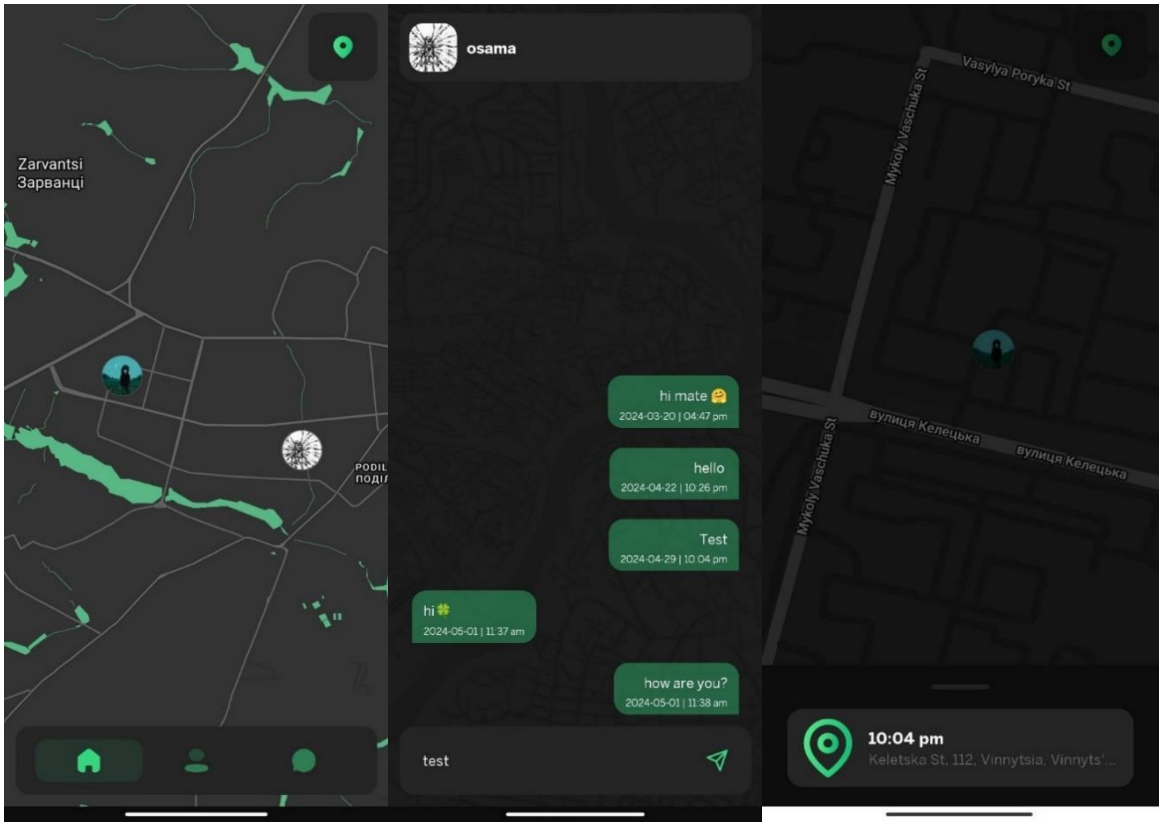


Рис. 1. Результат розробки додатка

Загалом SnapTrack не лише полегшує комунікацію та навігацію в умовах ведення бойових дій, але й забезпечує високий рівень безпеки та конфіденційності даних завдяки використанню Firebase. Це робить його важливим інструментом для військових підрозділів у досягненні їх стратегічних цілей.

Список використаних джерел

1. Kantamani S. P. Detailed Guide on Android Clean Architecture. Dec. 5, 2019. URL: https://medium.com/android-dev-hacks/detailed-guide-on-android-clean-architecture-9eab262a9011

2. Raj A. MVVM Clean Architecture Pattern in Android with Use Cases. Jun. 26, 2023. URL: https://medium.com/@ami0275/mvvm-clean-architecture-pattern-in-android-with-use-cases-eff7edc2ef76

3. Aung K. S. M. Flow Vs LiveData in Android Architecture Components. Oct. 3, 2023. URL: https://medium.com/@khunsoemoeaung/flow-vs-livedata-in-android-architecture-components-51745200e42c

4. Get Started with Firebase Authentication on Android. *Firebase.* URL: https://firebase.google.com/docs/auth/android/start

УДК 519.7:316.472.4]:004.832.2-042.2

Афанасьєва Д. С., *здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,*
Вєтров О. С., *старший викладач кафедри інформаційних технологій*

ПОРІВНЯННЯ АЛГОРИТМІВ ПОШУКУ В ШИРИНУ ТА В ГЛИБИНУ ДЛЯ ОБХОДУ ГРАФІВ У СОЦІАЛЬНИХ МЕРЕЖАХ

Донецький національний університет імені Василя Стуса, м. Вінниця

Соціальні мережі перетворилися на невичерпне джерело даних і відкрили нові можливості для аналізу поведінки користувачів, передбачення трендів та взаємозв'язків між людьми. Це спонукає до необхідності розробки та оптимізації алгоритмів обходу графів, які є важливим складником для ефективного аналізу й використання даних у соціальних мережах. Обрання правильного алгоритму стає ключовим етапом якісної та швидкої обробки великих масивів даних.

Алгоритм пошуку в ширину (BFS) розглядає граф як структуру, де вершини відвідуються у порядку росту відстані від початкової вершини. Цей алгоритм має схожість з обходом бінарного дерева, єдина відмінність у тому, що на відміну від них, графи можуть містити цикли. Для пошуку в ширину розглядається простий зв'язний граф з числом вершин V та числом ребер E [1]. Тому для початку обирається перша вершина, з якої починається алгоритм, далі відвідуються усі вершини, які знаходяться на відстані одного ребра від початкової, потім усі вершини на відстані двох ребер і так далі [2]. Під час обходу графу спершу вершини додаються в чергу і після їх проходу заносяться в масив відвіданих вершин. Часова складність алгоритму залежатиме від кількості вершин та ребер у графі й буде становити O(V+E). На рис. 1 показано реалізацію цього алгоритму мовою Python.

```python
def bfs(graph, start):
    visited = set()
    queue = []
    visited.add(start)
    queue.append(start)
    visited_nodes = []

    while queue:
        vertex = queue.pop(0)
        visited_nodes.append(vertex)

        for neighbor in graph[vertex]:
            if neighbor not in visited:
                visited.add(neighbor)
                queue.append(neighbor)

    return visited_nodes
```

Рис. 1. Реалізація алгоритму пошуку в ширину [2]

Для алгоритму пошуку в глибину (DFS) розглядається простий зв'язний граф з числом вершин V та числом ребер E [1]. Обхід графу розпочинається з вибору будь-якої вершини графу, яку ми поміщаємо до стека. Потім з вершини з верху стека ми беремо першу вершину, додаємо її до списку відвіданих і додаємо всі її сусідні вершини, які ще не були відвідані, до вершини стека. Ці кроки повторюються, поки стек не стане пустим, тобто всі вершини будуть відвідані. Часова складність алгоритму буде такою ж, як в алгоритму пошуку в ширину. Вона залежатиме від кількості вершин та ребер у графі й буде становити O(V+E). Отже, різниця в реалізації цих алгоритмів полягає в тому, що у BFS використовується черга для обходу вершин графу, а у DFS використовується стек, переходячи максимально можливо далеко вглиб від стартової вершини, перш ніж повернутися до інших сусідніх вершин. На рис. 2 показано реалізацію DFS мовою Python.

```python
def dfs(graph, start):
    visited = []
    stack = [start]

    while stack:
        current_vertex = stack.pop()
        if current_vertex not in visited:
            visited.append(current_vertex)
            print(current_vertex)

        for neighbor in graph[current_vertex]:
            if neighbor not in visited:
                stack.append(neighbor)

    return visited
```

Рис. 2. Реалізація алгоритму пошуку в глибину [3]

У контексті соціальних мереж застосування BFS та DFS залежить від конкретних потреб та постановки задачі. DFS має менші вимоги до пам'яті, порівняно з BFS, оскільки потрібно зберігати лише шлях від початкової вершини до поточної. Проте це може зробити DFS вразливим до нескінчених циклів, якщо граф містить такі цикли [4].

DFS може бути застосований у соціальних мережах для виявлення груп або спільнот користувачів. Наприклад, використання DFS може допомогти визначити, які користувачі утворюють тісні групи друзів або мають спільні інтереси. Також DFS може використовуватися для виявлення впливових осіб у мережі, які можуть мати важливий вплив на інших користувачів.

З іншого боку, BFS частіше використовується для вирішення конкретних завдань, як-от пошук найкоротшого шляху між користувачами або визначення рівня віддаленості між ними. Наприклад, за допомогою BFS можна знайти найшвидший спосіб зв'язку між двома користувачами або визначити, скільки зв'язків їх відокремлює. Такий аналіз може бути корисним для виявлення ступеня взаємодії між користувачами та оцінки їх впливу в мережі.

Отже, у цій роботі було досліджено два ключові алгоритми обходу графів: DFS (пошук в глибину) та BFS (пошук в ширину), і їх роль в аналізі соціальних

мереж. DFS дає змогу виявляти групи та спільноти користувачів, водночас, як BFS допомагає визначити найкоротший шлях між ними. Розуміння принципів роботи цих алгоритмів дає змогу краще аналізувати та розуміти структуру соціальних мереж, розподіл інформації та взаємозв'язки між користувачами.

Список використаних джерел

1. Кузьменко І. М. Теорія графів: навчальний посібник для здобувачів ступеня бакалавра за спеціальністю 122 «Комп'ютерні науки». Київ: КПІ ім. Ігоря Сікорського, 2020. 71 с. URL: https://ela.kpi.ua/server/api/core/bitstreams/fb0a4251-74d9-470b-88da-71abb4e85f93/content
2. Breadth First Search or BFS for a Graph. GeeksforGeeks. *Geeksforgeeks.* 26 Sep., 2024. URL: https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/#breadth-first-search-or-bfs-for-a-graph (дата звернення: 15.04.2024).
3. Depth First Search (DFS). Programiz: вебсайт. URL: https://www.programiz.com/dsa/graph-dfs (дата звернення: 10.05.2024).
4. Graph traversal: DFS vs BFS. Hyperskill: вебсайт. URL: https://hyperskill.org/learn/step/35510#depth-first-search (дата звернення: 11.05.2024).

УДК: 004.9

Молодченко Д. В., *здобувач 2 курсу спеціальності 122 Комп'ютерні науки*
науковий керівник:
Потапова Н. А., *канд. екон. наук, доцент, доцент кафедри інформаційних технологій*

МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ РОЗКЛАДУ ЗАНЯТЬ

Донецький національний університет імені Василя Стуса, м. Вінниця

Сучасна освітня система активно використовує корпоративні програми, що дають змогу здійснювати відеовиклики, листуватися та ділитися файлами, і все це можна переглянути навіть у транспорті, але до цієї групи застосунків досі не додали один тип програми – розклад. Здобувачі на початку кожного семестру стикаються з проблемою пошуку на телефоні файла, в якому міститься розклад. Аби подивитись номер аудиторії, в якій проходитиме заняття, доводиться долати шлях через файловий провідник телефона чи то ноутбука, який у здобувачів зазвичай запроваджений великою кількістю файлів. Функціонал застосунку має містити можливість додавати, редагувати та видаляти предмети, розклад, групи. До того ж у сучасному світі можливість отримати доступ до своїх даних на іншому пристрої стала вагомим плюсом для користувачів та сервісів що першими почали впроваджувати такий функціонал.

Мобільний застосунок – програмне забезпечення, платформою роботи якого є мобільні пристрої: смартфони та планшети. Зазвичай на цих пристроях встановлено одну з двох наступних операційних систем:

1. Android – відкрита операційна система, створена компанією Google у 2008 році на основі ядра Linux Kernel. Ця ОС стала надзвичайно популярною че-

рез можливість установлення на різні пристрої та легку кастомізацію під них. Для нативної розробки раніше використовували лише Java, але з 2017 року до неї додався Kotlin.

2. IOS – операційна система, розроблена компанією Apple для використання у пристроях власного виробництва. Через свою закритість і невелике різноманіття пристроїв, на які вона є націленою, пристрої з цією ОС у певний момент стали еталоном для всіх інших виробників на ринку мобільних пристроїв. Unix є ядром IOS. Для нативної розробки використовуються Swift та Objective-C. Частиною необхідного функціоналу таких застосунків є наявність акаунтів користувачів. Для створення акаунта потрібна електронна адреса та пароль. Електронна адреса використовується, наприклад, для верифікації користувача, а пароль – для захисту створеного акаунта та його даних у вашій програмі.

Створення груп / команд є способом для здійснення комунікації між великою кількістю людей. У сучасних корпоративних застосунках групи слугують для поширення файлів, відеозустрічей та інших завдань. Залежно від кількості учасників групи може виникнути потреба в комусь, хто зміг би виконувати певні функції її власника. Для таких ситуацій існує керування правами користувачів. Якщо функціонал програми є широким і групи включають велику кількість функцій, то і кількість прав у групі буде великою. Зазвичай певним користувачам надаються права адміністратора, тобто вони зі свого акаунта можуть редагувати файли, змінювати ім’я групи, додавати чи видаляти користувачів з неї і, можливо, навіть керувати складом адміністрації. Як висновок – адміністратори можуть теж мати різні права.

Основу дизайну застосунку складають UI сторінки. Сторінка для редагування предметів містить заголовок, заклик якого дає користувачу зрозуміти, що він має робити на цій сторінці (рис. 1). Такий підхід, замість рядка-заголовка, було обрано через візуальну просторість, що гармонує з концептом мінімалізму, який було вирішено використовувати для реалізації зовнішнього вигляду застосунку.

Рис. 1. UI сторінки для редагування списку предметів

Передбачено наявність сторінок: редактора розкладу, додаткової інформації, головної сторінки. Центральною точкою застосунку є головна сторінка з розкладом пар (рис. 2). Вона містить панель для відображення найближчих подій, вікно перегляду розкладу та навігаційну панель внизу для доступу до інших сторінок.

Рис. 2. Головна сторінка застосунку, що відображає розклад пар

Реалізація командного управління групою проходить через власника, адміністратора та учасника. В текстовому форматі вони відображаються так: //Owner>_>Group Name>_>Admin;Admin>_>Member;Member

Замість власника, адмінів і учасників записуються їх електронні пошти, оскільки назва папки збирається з елементів пошти. Під час створення групи адміністраторам розсилається копія файла власника. Учасники отримують лише частину, що містить власника та ім’я групи. Це робиться для того, аби люди отримали доступ до файла з подіями в папці власника.

Список використаних джерел

1. Cuello J., Vittone J. Design Mobile Apps. CreateSpace Independent Publishing Platform, 2014. 226 с.

2. Wathan A., Schoger S. Refactoring UI. The book, 2018. 252 с.

УДК 004.428/.43

Маруняк А. О., *здобувач 2 курсу спеціальності 122 Комп'ютерні науки,*
науковий керівник:
Поремський Ю. В., *канд. техн. наук, старший викладач кафедри інформаційних технологій*

ВАЖЛИВІСТЬ ВІЗУАЛЬНОГО ДИЗАЙНУ У ЗАСТОСУНКАХ, ВЕБСАЙТАХ ТА ВІДЕОІГРАХ

Донецький національний університет імені Василя Стуса, м. Вінниця

Візуальний дизайн відіграє важливу роль у створенні застосунків, вебсайтів та ігор, впливаючи на користувацький досвід (UX) і залученість користувачів. Цей аспект дизайну включає не лише естетичні елементи, як-от кольори, шрифти та графічні елементи, але й загальну структуру та взаємодію користувача з інтерфейсом.

Одним із ключових аспектів візуального дизайну є його здатність створювати перші враження. Як би ми не хотіли цього визнавати, але більшість людей насамперед сприймає зовнішній вигляд. Дослідження показують, що люди формують перше враження про застосунок, вебсайт, гру або навіть людину за перші кілька секунд взаємодії чи спілкування [2]. Це означає, що продукти з привабливим і професійним дизайном мають більші шанси привернути увагу користувачів та утримати їх зацікавленість. Проте, якщо перше враження буде негативним, користувачі можуть швидко втратити інтерес і перейти до інших продуктів.

До того ж привабливий та інтуїтивно зрозумілий дизайн може значно підвищити рівень залученості користувачів. Це означає, що користувачі проводитимуть більше часу у застосунку, а також будуть частіше до нього повертатися. Естетично приємні інтерфейси сприяють більшому задоволенню користувачів та їхній лояльності, а вдало підібраний графічний стиль взагалі часто стає візитівкою продукту. Це особливо важливо в умовах високої конкуренції на ринку, де утримання користувачів та формування клієнтської бази є ключовим фактором успіху [3].

Добре продуманий візуальний дизайн сприяє створенню позитивного користувацького досвіду. Він допомагає спростити навігацію, зробити інтерфейс зрозумілим та логічним, а також зменшити кількість помилок, що виникають під час використання застосунку, вебсайта або гри. Це особливо важливо для складних програм, де користувачі можуть легко загубитися без чіткого і зручного інтерфейсу. Наприклад, у багатьох сучасних мобільних застосунках та вебсайтах використовується мінімалістичний дизайн, який допомагає користувачам легко орієнтуватися та знаходити необхідну інформацію або функції [4].

Візуальний дизайн відіграє важливу роль у передачі емоцій та створенні впізнаваного бренду. Використання певних кольорів, форм і графічних елементів може викликати у користувачів певні емоції, що допомагає створити емоційний зв'язок з продуктом. До прикладу, яскраві кольори і динамічні графічні елементи

можуть викликати відчуття енергії та веселощів, тоді як спокійні відтінки та м'які форми можуть створювати відчуття спокою й комфорту. До того ж консистентний і якісний візуальний дизайн сприяє формуванню айдентики та впізнаваності бренду і довіри до нього. Це особливо важливо для компаній, які прагнуть створити сильний бренд та утримати своїх клієнтів.

У сучасному насиченому ринку якісний візуальний дизайн може стати вирішальним фактором конкурентоспроможності. Продукти з поганим дизайном швидко втрачають аудиторію, яка може перейти до конкурентів із більш привабливими та зручними інтерфейсами. В кожного з нас бувало таке, що на перший погляд від якогось продукту ми відчуваємо дискомфорт або розгубленість. Завдання дизайнера полягає саме в тому, щоб наш досвід як користувача був виключно позитивним, і ми запам'ятали продукт не тільки за назвою, а і за виглядом, і нам було приємно ним користуватись. Це означає, що інвестиції у візуальний дизайн є стратегічно важливими для успіху продукту. Розробники повинні приділяти значну увагу візуальному дизайну на всіх етапах створення продукту, від початкової концепції до фінального релізу.

Крім естетичних аспектів, візуальний дизайн також відіграє важливу роль у забезпеченні доступності застосунків, вебсайтів та ігор для різних категорій користувачів. Наприклад, використання контрастних кольорів та чітких шрифтів може допомогти людям з вадами зору легше сприймати інформацію на екрані. Інтерфейси, які враховують потреби людей з обмеженими можливостями, сприяють створенню інклюзивного продукту, який може бути корисним для ширшої аудиторії [5].

Візуальний дизайн також може впливати на поведінку користувачів. Дослідження показують, що певні візуальні елементи можуть стимулювати користувачів до певних дій, наприклад, до купівлі або реєстрації. Використання візуальних підказок та акцентів може допомогти користувачам легше орієнтуватися у застосунку, на вебсайті або у грі та швидше досягати своїх цілей.

Висновки. Візуальний дизайн є не просто декоративним елементом, а важливим складником, що значною мірою впливає на успіх продукту. Він сприяє створенню позитивного користувацького досвіду, емоційного зв'язку з продуктом, а також забезпечує конкурентні переваги на ринку. Інвестування у якісний візуальний дизайн є необхідним для розробників, які прагнуть створювати успішні та популярні застосунки, сайти чи навіть відеоігри. Враховуючи всі ці фактори, розробники повинні приділяти значну увагу візуальному дизайну на всіх етапах створення продукту, від початкової концепції до фінального релізу, щоб забезпечити максимальну задоволеність та лояльність користувачів.

Список використаних джерел

1. What is Visual Design. *Integration Design Foundation.* URL: http://surl.li/tzvbi (дата звернення 16.05.2024).

2. Wargo Eric. How Many Seconds to a First Impression. *Association for Psychological Science.* July 1, 2006. URL: http://surl.li/tzvbx (дата звернення 16.05.2024).

3. Visual Design And Why It Matters For Your Business: вебсайт. URL: http://surl.li/tzvcc (дата звернення 17.05.2024).

4. Anover J. 10 Benefits of Minimalism in UI/UX Design. *Free Logo Maker*. December 14, 2021. URL: http://surl.li/tzvcf (дата звернення 17.05.2024).

5. What Is Inclusive Design and What Does It Mean for the Digital World. *UITOP*. URL: http://surl.li/tzvcj (дата звернення 17.05.2024).

УДК 004.02

Коновалюк І. Л., *здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,*
науковий керівник:
Горяшин А. С., *асистент кафедри інформаційних технологій*

ВИКОРИСТАННЯ МЕТОДІВ ОБЧИСЛЕНЬ У ПРОГНОЗУВАННІ ПОВЕДІНКИ СКЛАДНИХ СИСТЕМ

Донецький національний університет імені Василя Стуса, м. Вінниця

Прогнозування – це процес передбачення майбутнього стану об'єкта чи явища на основі аналізу минулого і сьогодення й систематично оціненої інформації про якісні й кількісні характеристики майбутнього розвитку обраного об'єкта чи явища. Результатом прогнозування є передбачення, тобто знання про майбутнє і про очікуваний розвиток поточних тенденцій певних феноменальних об'єктів у майбутньому їх існуванні [1].

Методи прогнозування можна класифікувати за різними критеріями, зокрема за типом базових даних, типом методу та підходом до моделювання:

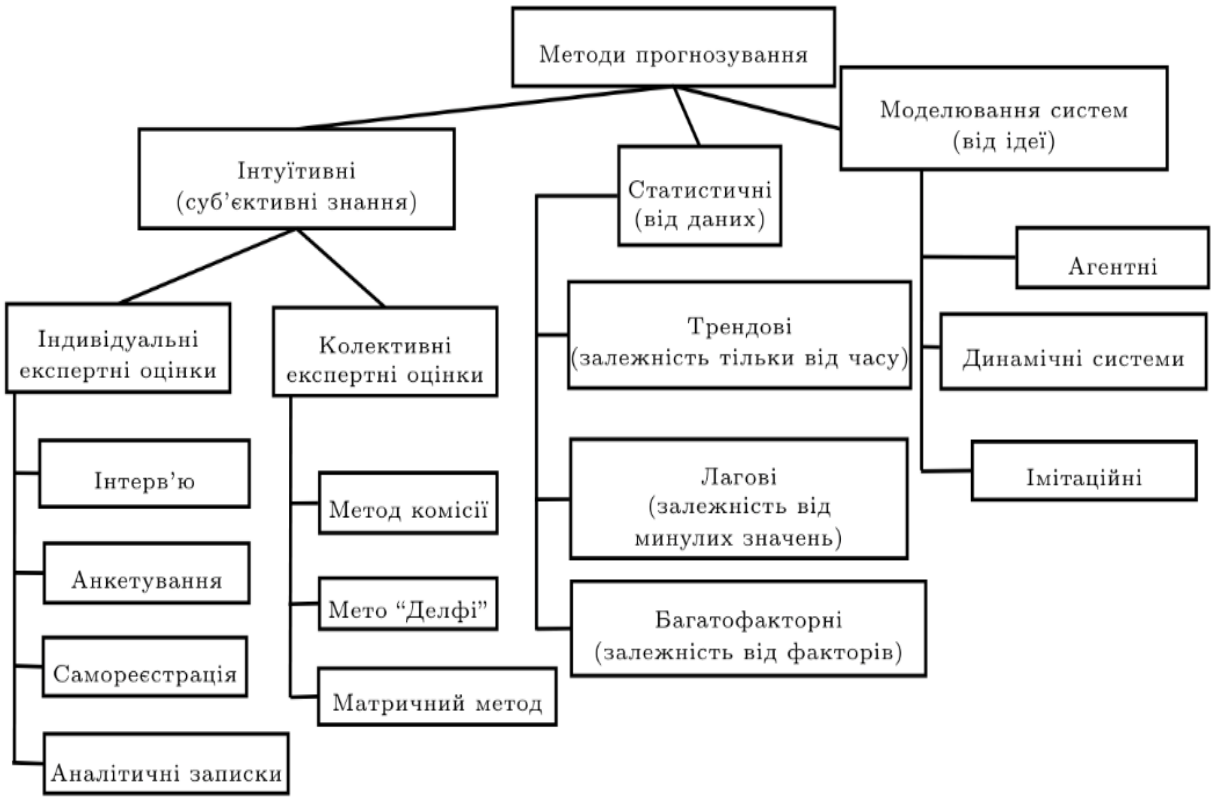


Рис. 1. Класифікація методів прогнозування

Прогнозування поведінки складних систем є важливим завданням у багатьох галузях зокрема в економіці, екології, техніці, медицині тощо.

Використання обчислювальних методів для прогнозування таких систем дає змогу моделювати та аналізувати динаміку системи, враховуючи взаємодію багатьох компонентів, і робити прогнози на основі великих обсягів даних [2]. Для прогнозування поведінки складних систем прийнято використовувати основні методи обчислень, серед яких математичне моделювання, статистичні методи та машинне навчання, методи оптимізації.

Для моделювання динаміки систем, де зміни стану системи описуються як функції часу, використовують диференціальні рівняння, що є доволі потужним інструментом. Як приклад, для моделювання зростання популяції з обмеженими ресурсами, що є доволі популярною задачею в сучасному світі з вичерпними запасами та постійним ростом населення, використовують логістичне рівняння, яке має вигляд:

$$dN/dt = rN(1-\frac{N}{K}),$$

де $N(t)$ – чисельність популяції в момент часу t, r – максимальний темп зростання популяції, K – ємність середовища (максимальна чисельність популяції, яку може підтримувати середовище). До того ж для вирішення аналогічної задачі також можна використовувати рівняння Лотки–Вольтерри, що описують динаміку двох видів у системі хижак – жертва.

Ще одним важливим завданням, яке можна розв'язати за допомогою диференціальних рівнянь, є моделювання поширення інфекційних хвороб. Такі завдання можна розв'язувати за допомогою епідеміологічної моделі, а саме, моделі SIR. Ця модель складається з трьох рівнянь:

$$dS/dt = -\beta SI;$$
$$dI/dt = \beta SI - \gamma I;$$
$$dR/dt = \gamma I.$$

Ці диференціальні рівняння є основними інструментами для моделювання динаміки складних систем у біології та епідеміології.

Вони дають змогу дослідникам аналізувати та прогнозувати поведінку систем, визначати важливі параметри, які впливають на динаміку, і приймати обґрунтовані рішення щодо керування цими системами.

Статистичні методи та методи машинного навчання широко використовуються для аналізу даних і прогнозування. Серед таких методів – регресійний аналіз, що використовується для моделювання залежностей між змінними, та прогнозування.

Основною метою є визначення форми зв'язку між залежною змінною (ціллю) та однією або кількома незалежними змінними (факторами). Модель лінійної регресії описується рівнянням:

$$y = \beta 0 + \beta 1 x\ 1\ + \beta\ 2\ x2 + \cdots + \beta\ nx\ n + \epsilon,$$

де y – залежна змінна, $x\ 1, x2, x3$ – незалежні змінні, $\beta 0$ – вільний член (перетин), $\beta 1, \beta 2, \dots, \beta n$ – коефіцієнти регресії, ϵ – випадкова похибка. Коефіцієнти

регресії оцінюються за допомогою методу найменших квадратів, що мінімізує суму квадратів відхилень між спостереженими та передбаченими значеннями. До того ж для вирішення подібних задач виокремлюють часові ряди, які змінюються з часом. Найбільш популярні моделі ARIMA і SARIMA.

Отже, використання обчислювальних методів для прогнозування поведінки складних систем дає змогу нам краще зрозуміти ці системи, передбачати їх поведінку та приймати обґрунтовані рішення. Поєднання математичних моделей, машинного навчання, оптимізації та аналітики великих даних відкриває нові можливості для аналізу складних систем і керування ними в різних галузях.

Список використаних джерел

1. Чисельні методи: навчальний посібник / Л. О. Волонтир, О. В. Зелінська, Н. А. Потапова, І. А. Чіков. Вінниця: ВНАУ. 2020 322 с.

2. Ian H. Witten and Eibe Frank Data Mining: Practical machine learning tools and techniques Morgan Kaufmann, 2011. 664 p.

Наукове видання

Прикладні інформаційні технології

МАТЕРІАЛИ
V ВСЕУКРАЇНСЬКОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ
ЗДОБУВАЧІВ, АСПІРАНТІВ ТА МОЛОДИХ ВЧЕНИХ

24 травня 2024 р.

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

Редактор О. А. Содатова
Технічний редактор О. К. Гомон

Підписано до друку 07.10.2024 р.
Формат 60×84/16. Папір офсетний.
Друк – цифровий. Умовн. друк. арк. 11,92
Тираж 100 прим. Зам. № 59

Донецький національний університет імені Василя Стуса
21021, м. Вінниця, вул. 600-річчя, 21.
Свідоцтво про внесення суб'єкта видавничої справи
до Державного реєстру
серія ДК № 5945 від 15.01.2018 р.