

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЗ „ЛУГАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА”



Навчально-науковий інститут математики та інформаційних технологій
Кафедра математики та інформатики

Ю. Г. Козуб, О. О. Смагіна

**МАТЕМАТИЧНІ МЕТОДИ ОПТИМІЗАЦІЇ ТА ДОСЛІДЖЕННЯ
ОПЕРАЦІЙ**

**методичні вказівки до лабораторних робіт
для здобувачів вищої освіти першого (бакалаврського) рівня
спеціальності F3 – «Комп'ютерні науки»**

Лубни - 2026

УДК 004.272:37

Рекомендовано до друку вченою радою
ДЗ «Луганський національний університет імені Тараса Шевченка»,
(протокол № 7 від 30.01.2026 р.)

Укладачі:

Козуб Ю. Г. – доктор технічних наук, професор;

Смагіна О. О. – кандидат педагогічних наук доцент.

Рецензенти:

Гоменюк С. І. – доктор технічних наук, професор, декан математичного факультету
Запорізького національного університету;

Могильний Г. А. – кандидат технічних наук, доцент, директор навчально-наукового
інституту математики та інформаційних технологій ДЗ „Луганський національний
університет імені Тараса Шевченка”.

Козуб Ю. Г., Смагіна О. О. Математичні методи оптимізації та дослідження
операцій: методичні вказівки до лабораторних для здобувачів вищої освіти першого
бакалаврського рівня спеціальності F3 – «Комп'ютерні науки». Лубни: ДЗ „ЛНУ імені
Тараса Шевченка”, 2026. 46 с.

Методичні вказівки призначені для здобувачів вищої освіти першого
(бакалаврського) рівня, які навчаються за денною та заочною формами з метою розвитку та
закріплення практичних навичок під час підготовки та виконання лабораторних робіт.
Представлені основні відомості, практичні завдання та контрольні питання до
лабораторних робіт з освітнього компоненту «Математичні методи оптимізації та
дослідження операцій».

ПЕРЕДМОВА

Мета вивчення дисципліни: Основною метою дисципліни «Математичні методи оптимізації та дослідження операцій» є формування у здобувачів сучасних уявлень про методології та практики застосування математичних методів для розв'язання задач оптимізації в галузі інформаційних технологій, оволодіння практичними навичками ефективного використання алгоритмів оптимізації та інструментальних засобів в умовах вирішення реальних практичних завдань. Також до цілей освоєння дисципліни відносяться формування у здобувачів здібностей з реалізації оптимального підходу під час аналізу задач прийняття рішень, вибору методів оптимізації та проєктування ефективних алгоритмічних рішень для програмних систем.

В результаті освоєння курсу здобувач опанує: основи теорії оптимізації та класифікацію задач оптимізації; методи лінійного програмування та симплекс-алгоритм; алгоритми розв'язання транспортних задач та задач про призначення; методи одновимірної та багатовимірної оптимізації; градієнтні методи та їх застосування в машинному навчанні; принципи еволюційних та генетичних алгоритмів; методологію динамічного програмування; основи теорії ігор та прийняття рішень; алгоритми оптимізації на графах; сучасні бібліотеки та інструменти для розв'язання задач оптимізації.

Вміти: формалізувати практичні задачі у вигляді математичних моделей оптимізації; застосовувати методи лінійного програмування для розв'язання ресурсних задач; розв'язувати транспортні задачі та задачі про призначення; реалізовувати градієнтні методи оптимізації на мові програмування; використовувати генетичні алгоритми для комбінаторних задач; застосовувати динамічне програмування до класичних алгоритмічних задач; моделювати ігрові ситуації та знаходити оптимальні стратегії; реалізовувати алгоритми на графах для задач маршрутизації; використовувати спеціалізовані бібліотеки оптимізації; аналізувати складність та ефективність алгоритмів оптимізації.

Володіти: навичками програмної реалізації алгоритмів оптимізації на Python; методиками формалізації реальних задач у термінах оптимізації; техніками вибору відповідного методу оптимізації для конкретної задачі; інструментами візуалізації результатів оптимізації; практиками використання бібліотек SciPy, NumPy для обчислень; засобами роботи з солверами лінійного програмування; методами налаштування гіперпараметрів алгоритмів; техніками порівняльного аналізу ефективності методів; інструментами профілювання та оптимізації коду; навичками документування та тестування оптимізаційних рішень.

ЗМІСТ

ТЕМА 1. ВСТУП ДО ОПТИМІЗАЦІЇ: ОСНОВНІ ПОНЯТТЯ ТА ЗАСТОСУВАННЯ В ІТ	6
ТЕМА 2. ЛІНІЙНЕ ПРОГРАМУВАННЯ ТА СИМПЛЕКС-МЕТОД.....	8
ТЕМА 3. ТРАНСПОРТНА ЗАДАЧА ТА ЗАДАЧА ПРО ПРИЗНАЧЕННЯ ...	10
ТЕМА 4. МЕТОДИ ОДНОВИМІРНОЇ ОПТИМІЗАЦІЇ.....	12
ТЕМА 5. ГРАДІЄНТНІ МЕТОДИ БАГАТОВИМІРНОЇ ОПТИМІЗАЦІЇ	15
ТЕМА 7. ДИНАМІЧНЕ ПРОГРАМУВАННЯ	20
ТЕМА 8. ТЕОРІЯ ІГОР ТА ПРИЙНЯТТЯ РІШЕНЬ ЧАСТИНА 1.....	23
ТЕМА 8. ТЕОРІЯ ІГОР ТА ПРИЙНЯТТЯ РІШЕНЬ ЧАСТИНА 2.....	26
ТЕМА 9. ТЕОРІЯ ГРАФІВ У ЗАДАЧАХ ОПТИМІЗАЦІЇ ЧАСТИНА 1	30
ТЕМА 9. ТЕОРІЯ ГРАФІВ У ЗАДАЧАХ ОПТИМІЗАЦІЇ ЧАСТИНА 2	33
ТЕМА 10. СУЧАСНІ БІБЛІОТЕКИ ТА ІНСТРУМЕНТИ ОПТИМІЗАЦІЇ ЧАСТИНА 1.....	37
ТЕМА 10. СУЧАСНІ БІБЛІОТЕКИ ТА ІНСТРУМЕНТИ ОПТИМІЗАЦІЇ ЧАСТИНА 2.....	41

ТЕМА 1. ВСТУП ДО ОПТИМІЗАЦІЇ: ОСНОВНІ ПОНЯТТЯ ТА ЗАСТОСУВАННЯ В ІТ

Мета роботи: набути практичних навичок формалізації реальних задач у вигляді математичних моделей оптимізації, освоїти основні поняття теорії оптимізації, навчитися класифікувати задачі оптимізації за типами, розвинути розуміння застосування методів оптимізації в різних галузях інформаційних технологій, опанувати базові підходи до визначення цільової функції та обмежень.

Практичне завдання

1. Необхідно провести аналіз та формалізацію задачі оптимізації розміщення серверів для веб-додатку. Компанія має 5 регіональних дата-центрів з різною вартістю оренди серверного обладнання на місяць: Київ 150 доларів за сервер, Львів 120 доларів, Одеса 130 доларів, Харків 110 доларів, Дніпро 125 доларів. Кожен дата-центр має обмежену кількість доступних серверних слотів: Київ 10 слотів, Львів 8 слотів, Одеса 6 слотів, Харків 12 слотів, Дніпро 7 слотів. Компанії потрібно розмістити мінімум 20 серверів для забезпечення роботи додатку з мінімальними витратами.

2. Для цієї задачі необхідно визначити змінні рішення, які представляють кількість серверів що розміщуються в кожному дата-центрі, позначити їх як x_1 для Києва, x_2 для Львова, x_3 для Одеси, x_4 для Харкова, x_5 для Дніпра. Сформулювати цільову функцію, яка мінімізує загальні витрати на оренду серверів, що дорівнює $150 * x_1 + 120 * x_2 + 130 * x_3 + 110 * x_4 + 125 * x_5$

3. Визначити обмеження задачі, які включають обмеження на мінімальну кількість серверів: $x_1 + x_2 + x_3 + x_4 + x_5 \geq 20$, обмеження на доступність слотів у кожному дата-центрі: $0 \leq x_1 \leq 10$; $0 \leq x_2 \leq 8$; $0 \leq x_3 \leq 6$; $0 \leq x_4 \leq 12$; $0 \leq x_5 \leq 7$.

усі x_1, x_2, x_3, x_4, x_5 — цілі числа, змінних оскільки не можна розмістити дробову кількість серверів.

Контрольні питання

1. Опишіть основні компоненти задачі оптимізації та їх взаємозв'язок, зокрема необхідно пояснити що таке змінні рішення та як їх правильно визначати в залежності від контексту задачі, що являє собою цільова функція та яких типів вона може бути з прикладами мінімізації та максимізації, яку роль відіграють обмеження в задачі оптимізації та які типи обмежень існують включаючи рівності, нерівності, обмеження на знак змінних, що таке область допустимих розв'язків та оптимальний розв'язок.

2. Поясніть класифікацію задач оптимізації за різними критеріями, де необхідно розкрити різницю між лінійними та нелінійними задачами з прикладами з практики програмування, пояснити що таке опукла та неопукла оптимізація та чому це важливо для вибору методу розв'язання, описати відмінність між задачами з неперервними та дискретними змінними, розглянути багатокритеріальну оптимізацію коли потрібно оптимізувати декілька цілей одночасно, навести приклади стохастичної оптимізації де є невизначеність у вхідних даних.

3. Потрібно описати практичне застосування оптимізації в різних галузях інформаційних технологій, навести приклади з веб-розробки такі як оптимізація навантаження на сервери.

Література

1. Jinyi Jiang. Algorithm Optimization and Performance Enhancement in Computer Science. 2025. URL: <https://is.gd/iuirQ2>
2. Volodymyr Yakhno, Vadym Kolumbet, Petar Halachev, Vladyslav Khambir, Ruslan Ivanenko. Methods and algorithms of optimization in computer engineering: review and comparative analysis. 2024. URL: <https://is.gd/WUp7B6>

ТЕМА 2. ЛІНІЙНЕ ПРОГРАМУВАННЯ ТА СИМПЛЕКС-МЕТОД

Мета роботи: набути практичних навичок розв'язання задач лінійного програмування графічним методом та методом симплекс-таблиць, освоїти процес побудови математичної моделі лінійної оптимізації, навчитися інтерпретувати геометричний зміст задачі лінійного програмування, опанувати техніку аналізу чутливості розв'язку до зміни параметрів, розвинути розуміння застосування лінійного програмування для розв'язання ресурсних задач в IT-проектах.

Практичне завдання

1. Необхідно розв'язати задачу оптимізації розподілу робочого часу програміста між двома проектами. Програміст працює на фрілансі та отримав два замовлення: розробка веб-сайту з оплатою 30 доларів за годину та розробка мобільного додатку з оплатою 40 доларів за годину. У програміста є обмеження: загальний робочий час на тиждень не більше 40 годин, на розробку веб-сайту клієнт вимагає мінімум 15 годин на тиждень, на мобільний додаток потрібно виділити мінімум 10 годин, через складність мобільної розробки програміст може працювати над нею максимум 25 годин на тиждень.

2. Сформулювати математичну модель задачі де x_1 це кількість годин на веб-сайт, x_2 це кількість годин на мобільний додаток, цільова функція максимізує дохід: $30x_1 + 40x_2$. Обмеження включають загальний час: $x_1 + x_2 \leq 40$, мінімум для веб-сайту: $x_1 \geq 15$, мінімум для мобільного додатку: $x_2 \geq 10$, максимум для мобільного додатку: $x_2 \leq 25$, невід'ємність: $x_1 \geq 0$; $x_2 \geq 0$.

3. Розв'язати задачу графічним методом побудувавши систему координат де вісь x один горизонтальна та вісь y два вертикальна, побудувати прямі що відповідають кожному обмеженню, визначити область допустимих розв'язків як перетин усіх напівплощин, знайти координати вершин багатокутника допустимих розв'язків, обчислити значення цільової функції в кожній вершині, визначити оптимальний розв'язок як вершину з

максимальним значенням цільової функції. Побудувати лінії рівня цільової функції щоб візуалізувати напрямок зростання доходу.

Контрольні запитання

1. Опишіть структуру канонічної форми задачі лінійного програмування, пояснити що таке базисні та небазисні змінні в симплекс-методі, розглянути поняття допустимого базисного розв'язку та його зв'язок з вершинами області допустимих розв'язків, описати геометричну інтерпретацію симплекс-методу як руху по вершинах многогранника в напрямку покращення цільової функції, пояснити критерій оптимальності розв'язку в симплекс-таблиці.

2. Поясніть алгоритм графічного методу розв'язання задач лінійного програмування з двома змінними, описати послідовність кроків від побудови обмежень до знаходження оптимуму, розглянути можливі випадки: задача має єдиний оптимум, задача має безліч оптимальних розв'язків на ребрі многокутника, задача необмежена тобто цільова функція може зростати необмежено, область допустимих розв'язків порожня тобто обмеження несумісні, навести геометричну інтерпретацію кожного випадку.

3. Опишіть практичне застосування лінійного програмування в IT-індустрії, навести приклади задач оптимального розподілу обчислювальних ресурсів у хмарних платформах, розглянути задачі планування робочого навантаження в DevOps та CI CD системах.

Література

1. Mengyu Huang, Yuxing Zhong, Huiwen Yang, Jiazheng Wang, Fan Zhang, Bo Bai, Ling Shi. Simplex Initialization: A Survey of Techniques and Trends. 2021. URL: <https://arxiv.org/abs/2111.03376>
2. Anqi Li, Tiande Guo, Congying Han, Bonan Li, Haoran Li. On the optimal pivot path of simplex method for linear programming based on reinforcement learning. 2022. URL: <https://arxiv.org/abs/2210.02945>

ТЕМА 3. ТРАНСПОРТНА ЗАДАЧА ТА ЗАДАЧА ПРО ПРИЗНАЧЕННЯ

Мета роботи: набути практичних навичок розв'язання транспортних задач методом потенціалів, освоїти алгоритм знаходження початкового опорного плану методами північно-західного кута та мінімального елемента, навчитися формалізувати та розв'язувати задачі про призначення угорським алгоритмом, опанувати техніки перевірки оптимальності плану перевезень, розвинути розуміння застосування цих задач в логістиці та розподілі ресурсів в ІТ-системах.

Практичне завдання

1. Необхідно розв'язати транспортну задачу розподілу навантаження між серверами та користувачами. Компанія має 3 сервери розташовані в різних дата-центрах з доступною пропускнуою здатністю: сервер А може обробити сто запитів за секунду, сервер Б може обробити 80 запитів за секунду, сервер В може обробити 120 запитів за секунду. Є 4 групи користувачів з потребами: група один потребує 60 запитів за секунду, група два потребує 70 запитів за секунду, група три потребує 50 запитів за секунду, група чотири потребує 80 запитів за секунду.

2. Латентність відповіді від кожного сервера до кожної групи користувачів у мілісекундах подана у вигляді таблиці де рядки це сервери а стовпці це групи користувачів: від А до груп 10, 15, 20, 25 відповідно, від Б до груп 20, 10, 15, 30 відповідно, від В до груп 15, 25, 10, 15 відповідно. Метою є мінімізація сумарної зваженої латентності де кожен маршрут має вагу пропорційну кількості запитів що по ньому йдуть.

3. Перевірити баланс задачі порівнявши сумарну пропускну здатність серверів триста запитів за секунду з сумарними потребами користувачів двісті шістдесят запитів за секунду, визначити що задача є незбалансованою з надлишком пропускнуої здатності. Ввести фіктивного споживача з потребою 40 запитів за секунду та 0 вартістю доставки від усіх серверів. Знайти

початковий опорний план методом північно-західного кута та методом мінімального елемента, порівняти значення цільової функції для обох планів.

Контрольні запитання

1. Опишіть математичну постановку транспортної задачі, пояснити що таке постачальники споживачі обсяги постачання обсяги попиту та вартості перевезення, розглянути умови збалансованості задачі та що робити якщо вона не збалансована, описати структуру плану перевезень як матриці де елементи показують кількість вантажу від постачальника до споживача, пояснити поняття опорного плану та базисних змінних, розглянути правило про кількість базисних змінних яка повинна дорівнювати m плюс n мінус один де m кількість постачальників та n кількість споживачів.

2. Поясніть метод потенціалів для розв'язання транспортної задачі, описати алгоритм обчислення потенціалів для рядків та стовпців, розглянути як перевіряється оптимальність плану через обчислення оцінок для незайнятих клітинок, пояснити побудову циклу перерахунку для покращення плану, описати як визначається кількість вантажу що перерозподіляється по циклу, розглянути критерій зупинки алгоритму коли досягається оптимальний план.

3. Опишіть задачу про призначення як окремий випадок транспортної задачі, пояснити відмінності коли кожен постачальник може постачати тільки одну одиницю та кожен споживач потребує тільки одну одиницю.

Література

1. Reena Gajanandbhai Patel, P. H. Bhathawala, Bhavin S. Patel. OPTIMAL SOLUTION OF AN ASSIGNMENT PROBLEM AS A SPECIAL CASE OF TRANSPORTATION PROBLEM. 2018. URL: <https://is.gd/ZyvSdY>
2. Milos Seda. The Assignment Problem and Its Relation to Logistics Problems. 2022. URL: <https://www.mdpi.com/1999-4893/15/10/377>

ТЕМА 4. МЕТОДИ ОДНОВИМІРНОЇ ОПТИМІЗАЦІЇ

Мета роботи: набути практичних навичок застосування методів одновимірної оптимізації для мінімізації функцій однієї змінної, освоїти алгоритм методу золотого перерізу та методу дихотомії, навчитися реалізовувати градієнтний спуск для одновимірних функцій, опанувати техніки оцінки швидкості збіжності різних методів, розвинути розуміння застосування одновимірної оптимізації як підзадачі в більш складних методах оптимізації.

Практичне завдання

1. Необхідно знайти оптимальне значення гіперпараметра швидкості навчання для простої моделі машинного навчання. Функція втрат залежить від швидкості навчання α за формулою:

$$f(\alpha) = (\alpha - 0.5)^2 + 0.2 \cdot \sin(10\alpha) + 0.1$$

Потрібно знайти мінімум цієї функції на інтервалі $[0, 1]$, що відповідає оптимальній швидкості навчання.

2. Реалізувати метод золотого перерізу для знаходження мінімуму функції, використовуючи константу золотого перерізу:

$$\varphi = (1 + \sqrt{5})/2 \approx 1.618$$

На кожній ітерації інтервал невизначеності звужується шляхом обчислення двох внутрішніх точок:

$$x_1 = b - (b - a)/\varphi$$

$$x_2 = a + (b - a)/\varphi$$

де $[a, b]$ — це поточний інтервал пошуку. Порівнюються значення функції $f(x_1)$ та $f(x_2)$:

якщо $f(x_1) > f(x_2)$, то новий інтервал $[x_1, b]$

інакше новий інтервал $[a, x_2]$

3. Продовжувати ітерації методу золотого перерізу, поки довжина інтервалу $(b - a)$ не стане меншою за задану точність $\varepsilon = 0.01$. Вести таблицю ітерацій з наступними стовпцями:

Номер ітерації k

Поточний інтервал $[a, b]$

Внутрішні точки x_1 та x_2

Значення функції $f(x_1)$ та $f(x_2)$

Довжина інтервалу $(b - a)$

Після завершення алгоритму визначити оптимальне значення α^* як середину фінального інтервалу та обчислити $f(\alpha^*)$.

Контрольні питання

1. Опишіть принцип роботи методів звуження інтервалу для одновимірної оптимізації, пояснити різницю між методом дихотомії та методом золотого перерізу, розглянути чому метод золотого перерізу є більш ефективним з точки зору кількості обчислень функції, описати константу золотого перерізу та її математичні властивості, пояснити як швидкість збіжності методу пов'язана з співвідношенням звуження інтервалу на кожній ітерації, розглянути оцінку кількості ітерацій необхідних для досягнення заданої точності.
2. Поясніть застосування градієнтного спуску для одновимірної оптимізації, описати як обирається крок градієнтного спуску та його вплив на збіжність, розглянути проблеми що можуть виникнути такі як коливання біля мінімуму при занадто великому кроці або повільна збіжність при занадто малому кроці, пояснити що таке локальні мінімуми та як вони впливають на результат градієнтного методу, описати модифікації градієнтного спуску з адаптивним кроком, розглянути метод Ньютонів який використовує другу похідну для швидшої збіжності.
3. Опишіть практичне застосування одновимірної оптимізації в машинному навчанні та програмуванні.

Література

1. Zhiting Tong, Zedong Song, Chao Zhang, Hao Xing, Yuxuan Sun. One-Dimensional Optimization Design of Supercritical Carbon Dioxide Radial Inflow Turbine and Performance Analysis under Off-Design Conditions. 2022. URL: <https://www.mdpi.com/2076-3417/12/8/3882>

2. Vicent Girbés-Juan, Joaquín Moll, Antonio Sala, Leopoldo Armesto. Cautious Bayesian Optimization: A Line Tracker Case Study. 2023. URL: <https://www.mdpi.com/1424-8220/23/16/7266>

ТЕМА 5. ГРАДІЄНТНІ МЕТОДИ БАГАТОВИМІРНОЇ ОПТИМІЗАЦІЇ

Мета роботи: набути практичних навичок застосування градієнтних методів для мінімізації функцій багатьох змінних, освоїти реалізацію методу найшвидшого спуску та стохастичного градієнтного спуску, навчитися обчислювати градієнти функцій аналітично та чисельними методами, опанувати техніки налаштування гіперпараметрів градієнтних методів, розвинути розуміння застосування градієнтної оптимізації в тренуванні моделей машинного навчання та нейронних мереж.

Практичне завдання

1. Необхідно мінімізувати функцію Розенброка двох змінних, яка часто використовується для тестування алгоритмів оптимізації:

$$f(x_1, x_2) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2$$

Ця функція має глобальний мінімум в точці $(1, 1)$, де $f(1, 1) = 0$, але характеризується вузькою долиною подібною областю, де функція майже не змінюється, що створює труднощі для оптимізації.

2. Обчислити градієнт функції Розенброка аналітично:

$$\partial f / \partial x_1 = -2(1 - x_1) - 400x_1(x_2 - x_1^2)$$

$$\partial f / \partial x_2 = 200(x_2 - x_1^2)$$

Реалізувати метод градієнтного спуску з постійним кроком $\alpha = 0.001$, використовуючи ітераційну формулу:

$$x_{k+1} = x_k - \alpha \cdot \nabla f(x_k)$$

3. Почати оптимізацію з початкової точки $x_0 = (-1.2, 1)$, виконати 500 ітерацій градієнтного спуску, зберігаючи траєкторію руху точки та значення функції на кожній ітерації. Побудувати графік траєкторії оптимізації на площині (x_1, x_2) з нанесенням ліній рівня функції Розенброка, побудувати графік залежності значення функції $f(x_k)$ від номера ітерації k для аналізу збіжності. Обчислити фінальну точку та значення функції в ній, порівняти з теоретичним оптимумом $(1, 1)$.

Контрольні питання

1. Опишіть математичні основи методу градієнтного спуску, пояснити чому напрямок антиградієнта $-\nabla f(x)$ є напрямком найшвидшого спаду функції, розглянути умови збіжності градієнтного спуску для опуклих та сильно опуклих функцій, описати вплив обумовленості матриці Гессе на швидкість збіжності, пояснити різницю між градієнтним спуском з постійним кроком та методом найшвидшого спуску де крок оптимізується на кожній ітерації, розглянути модифікації з моментом та адаптивним кроком.

2. Поясніть стохастичний градієнтний спуск (SGD) та його відмінності від звичайного градієнтного спуску, описати чому в машинному навчанні використовується оцінка градієнта на підвибірці даних замість всього датасету, розглянути поняття міні-батчу та його оптимальний розмір, пояснити проблеми що виникають при стохастичній оптимізації такі як шум в оцінці градієнта та коливання біля мінімуму, описати сучасні модифікації SGD такі як SGD з моментом, Adam, RMSprop, AdaGrad та їх переваги.

3. Опишіть практичне застосування градієнтних методів в глибокому навчанні, навести приклади тренування нейронних мереж де градієнти обчислюються методом зворотного поширення помилки (backpropagation), розглянути техніки прискорення збіжності такі як нормалізація батчів, ініціалізація ваг, функції активації, пояснити проблему зникаючого та вибухаючого градієнта, описати застосування градієнтної оптимізації для налаштування гіперпараметрів моделей, розглянути використання автоматичного диференціювання в сучасних фреймворках таких як PyTorch та TensorFlow.

Література

1. Vagelis Plevris, German Solorzano. A Collection of 30 Multidimensional Functions for Global Optimization Benchmarking. 2022. URL: <https://www.mdpi.com/2306-5729/7/4/46>
2. Iyanuoluwa Emiola, Robson Adem. Comparison of Minimization Methods for Rosenbrock Functions. 2021. URL: <https://arxiv.org/abs/2101.10546>

ТЕМА 6. ГЕНЕТИЧНІ АЛГОРИТМИ ТА ЕВРИСТИЧНІ МЕТОДИ

Мета роботи: набути практичних навичок застосування генетичних алгоритмів для розв'язання комбінаторних задач оптимізації, освоїти принципи природного відбору та еволюції в алгоритмічному контексті, навчитися реалізовувати операції селекції, схрещування та мутації, опанувати техніки кодування розв'язків задач у вигляді хромосом, розвинути розуміння застосування евристичних методів коли точні алгоритми неефективні через високу обчислювальну складність.

Практичне завдання

1. Необхідно розв'язати задачу складання оптимального розкладу занять для невеликого навчального закладу, використовуючи генетичний алгоритм. Є 5 викладачів, які ведуть 8 різних дисциплін, 4 аудиторії з різною місткістю (25, 30, 35, 40 студентів), та 5 часових слотів протягом дня (з 9:00 до 18:00). Кожна дисципліна вимагає 2 пари на тиждень, деякі викладачі мають обмеження на доступність у певні часові слоти, деякі дисципліни вимагають спеціального обладнання, доступного тільки в певних аудиторіях.

2. Кодувати розв'язок задачі у вигляді хромосоми, де кожен ген представляє одне заняття та містить інформацію про: дисципліну, викладача, аудиторію, день тижня та часовий слот. Хромосома — це масив з 16 генів (оскільки всього 8 дисциплін по 2 пари кожна).

Визначити функцію пристосованості $F(x)$, яка оцінює якість розкладу:

$$F(x) = -P_{\text{hard}} - \lambda \cdot P_{\text{soft}}$$

де:

P_{hard} — штраф за порушення жорстких обмежень (викладач не може вести дві пари одночасно, аудиторія не може бути зайнята для двох занять одночасно, місткість аудиторії повинна відповідати кількості студентів)

P_{soft} — штраф за порушення м'яких обмежень (мінімізація вікон у розкладі викладачів, рівномірний розподіл навантаження по днях тижня, врахування переваг викладачів)

$\lambda = 0.1$ — ваговий коефіцієнт для м'яких обмежень

3. Реалізувати генетичний алгоритм з такими параметрами:

Розмір популяції: $N = 50$ особин

Кількість поколінь: $G = 100$ ітерацій

Ймовірність схрещування: $p_c = 0.8$

Ймовірність мутації: $p_m = 0.1$

Початкова популяція генерується випадково з перевіркою базових жорстких обмежень. Використовувати:

Турнірну селекцію з розміром турніру 5 особин для вибору батьків

Одноточкове схрещування: випадково обирається точка розділу хромосоми, і частини двох батьків обмінюються

Мутацію: випадково змінює один або декілька генів в хромосомі, обираючи нові значення для викладача, аудиторії або часового слоту.

Контрольні питання

1. Опишіть біологічні аналогії в генетичних алгоритмах, пояснити що таке хромосома, ген, алель, популяція, покоління в контексті еволюційної оптимізації, розглянути принципи природного відбору, виживання найпристосованіших та їх відображення в алгоритмі, описати різницю між генотипом та фенотипом розв'язку, пояснити чому генетичні алгоритми є стохастичними методами та як це впливає на їх застосування, розглянути переваги та недоліки еволюційного підходу порівняно з детерміністичними методами оптимізації.

2. Поясніть основні оператори генетичного алгоритму та їх роль, описати різні методи селекції такі як турнірна селекція, рулеточна селекція, ранжування з поясненням їх переваг та недоліків, розглянути типи схрещування включаючи одноточкове, багатоточкове, однорідне схрещування та їх вплив на дослідження простору пошуку, пояснити роль мутації як оператора що забезпечує генетичне різноманіття та допомагає уникати локальних оптимумів, описати параметри алгоритму такі як розмір популяції, ймовірності схрещування та мутації та їх налаштування.

3. Опишіть практичне застосування генетичних алгоритмів в інформаційних технологіях, навести приклади розв'язання задачі комівояжера (TSP), складання розкладів, оптимізації мережевих топологій, розглянути застосування в машинному навчанні для підбору архітектури нейронних мереж (neural architecture search) та гіперпараметрів, пояснити використання в ігровому AI для еволюції поведінки персонажів, описати застосування для оптимізації алгоритмів та програмного коду (генетичне програмування), розглянути коли евристичні методи переважні за точні алгоритми з огляду на складність задачі та доступний час обчислень.

Література

1. Aymeric Vie, Alissa M. Kleinnijenhuis, Doyne J. Farmer. Qualities, challenges and future of genetic algorithms: a literature review. 2021. URL: <https://arxiv.org/abs/2011.05277>
2. Mariana A. Londe, Luciana S. Pessoa, Carlos E. Andrade, Mauricio G. C. Resende. Biased Random-Key Genetic Algorithms: A Review. 2023. URL: <https://arxiv.org/abs/2312.00961>

ТЕМА 7. ДИНАМІЧНЕ ПРОГРАМУВАННЯ

Мета роботи: набути практичних навичок застосування методу динамічного програмування для розв'язання задач оптимізації з оптимальною підструктурою, освоїти принцип оптимальності Беллмана, навчитися розпізнавати задачі, які можна ефективно розв'язати динамічним програмуванням, опанувати техніки мемоізації та табуляції, розвинути розуміння застосування динамічного програмування в алгоритмах на рядках, графах та інших структурах даних.

Практичне завдання

1. Необхідно розв'язати класичну задачу про рюкзак (0/1 Knapsack Problem) методом динамічного програмування. Програміст збирає рюкзак для роботи і має обмеження ваги $W = 5$ кг. Доступні предмети для упаковки:

Ноутбук: вага $w_1 = 3$ кг, цінність $v_1 = 100$ балів

Зовнішня клавіатура: вага $w_2 = 1$ кг, цінність $v_2 = 30$ балів

Миша: вага $w_3 = 0.5$ кг, цінність $v_3 = 15$ балів

Навушники: вага $w_4 = 0.3$ кг, цінність $v_4 = 20$ балів

Зарядний пристрій: вага $w_5 = 0.8$ кг, цінність $v_5 = 40$ балів

Павербанк: вага $w_6 = 0.4$ кг, цінність $v_6 = 25$ балів

Документи: вага $w_7 = 0.2$ кг, цінність $v_7 = 10$ балів

2. Сформулювати задачу математично:

Максимізувати: $\sum_{i=1}^n v_i \cdot x_i$

при обмеженні: $\sum_{i=1}^n w_i \cdot x_i \leq W$

де $x_i \in \{0, 1\}$ — змінна рішення (1 якщо предмет i упаковується, 0 інакше).

Застосувати метод динамічного програмування, створивши таблицю D розміром $(n+1) \times (10W+1)$, де ваги представлені в одиницях 100 грамів для спрощення обчислень ($W = 50$ одиниць).

3. Заповнити таблицю динамічного програмування, де $D[i][w]$ представляє максимальну цінність, яку можна досягти використовуючи перші i предметів з максимальною вагою w .

Рекурентне співвідношення:

$D[0][w] = 0$ для всіх w (базовий випадок — немає предметів)

Для $i = 1..n$ та $w = 0..W$:

$D[i][w] = D[i-1][w]$, якщо $w_i > w$ (предмет занадто важкий)

$D[i][w] = \max \{D[i-1][w], v_i + D[i-1][w-w_i]\}$, інакше.

4. Після заповнення таблиці значення $D[n][W]$ дає максимальну цінність. Відновити набір предметів, які входять в оптимальний розв'язок зворотним проходом по таблиці, починаючи з $D[n][W]$:

```
i = n, w = W
while i > 0:
    if D[i][w] != D[i-1][w]:
        x_i = 1 // предмет i включений
        w = w - w_i
    i = i - 1
```

Вивести список обраних предметів, їх сумарну вагу та цінність, переконатися що вага не перевищує 5 кг.

Контрольні питання

1. Опишіть принцип оптимальності Беллмана та його значення для динамічного програмування, пояснити що таке оптимальна підструктура задачі та чому вона необхідна для застосування методу, розглянути що таке перекриття підзадач (overlapping subproblems) та як це дозволяє уникнути повторних обчислень, описати різницю між підходами "згори вниз" (memoізація, top-down) та "знизу вгору" (табуляція, bottom-up) з поясненням їх переваг та недоліків, пояснити часову та просторову складність розв'язків задач динамічним програмуванням.

2. Поясніть застосування динамічного програмування для задач на рядках, розглянути класичні задачі такі як найдовша спільна підпоследовність (LCS), відстань редагування Левенштейна (Edit Distance), найдовша зростаюча підпоследовність (LIS) з детальним поясненням рекурентних співвідношень,

описати як будуються таблиці динамічного програмування для цих задач, пояснити процес відновлення оптимального розв'язку — не тільки значення, а й самої підпоследовності або последовності операцій.

3. Навести практичні застосування в текстових редакторах, системах контролю версій, біоінформатиці.

Література

1. Kyriakos Axiotis, Christos Tzamos. Capacitated Dynamic Programming: Faster Knapsack and Graph Algorithms. 2018. URL: <https://arxiv.org/abs/1802.06440>

2. Patcharin Buayen, Jeeraporn Werapun. Efficient 0/1-Multiple-Knapsack Problem Solving by Hybrid DP Transformation and Robust Unbiased Filtering. 2022. URL: <https://www.mdpi.com/1999-4893/15/10/366>

ТЕМА 8. ТЕОРІЯ ІГОР ТА ПРИЙНЯТТЯ РІШЕНЬ ЧАСТИНА 1

Мета роботи: набути практичних навичок моделювання конфліктних ситуацій та задач прийняття рішень засобами теорії ігор, освоїти поняття стратегій гравців та платіжних матриць, навчитися знаходити рівновагу Неша в матричних іграх, опанувати методи розв'язання ігор в чистих та змішаних стратегіях, розвинути розуміння застосування теорії ігор в розробці ігрових AI, аукціонних системах та системах прийняття рішень.

Практичне завдання

1. Необхідно проаналізувати конкурентну ситуацію між двома IT-компаніями, які виводять на ринок нові продукти. Компанія А та Компанія В одночасно обирають стратегії ціноутворення для своїх нових програмних продуктів. Кожна компанія може обрати одну з трьох стратегій:

S₁: Низька ціна (агресивна стратегія захоплення ринку)

S₂: Середня ціна (збалансована стратегія)

S₃: Висока ціна (преміум позиціонування)

2. Платіжна матриця для Компанії А представлена у таблиці (виграш в тисячах доларів прибутку) залежно від комбінації стратегій обох компаній представлена таблицею:

Таблиця 1 Платіжна матриця для Компанії А

	B: S ₁	B: S ₂	B: S ₃
A: S ₁	30	50	70
A: S ₂	40	45	60
A: S ₃	20	35	55

Таблиця 2 Платіжна матриця для Компанії В (виграш в тисячах доларів):

	B: S ₁	B: S ₂	B: S ₃
A: S ₁	30	20	10
A: S ₂	40	45	30

A: S ₃	60	50	55
-------------------	----	----	----

Визначити тип гри: це біматрична гра (гра двох осіб з ненульовою сумою), оскільки сума виграшів гравців в кожній клітинці не є константою.

3. Знайти рівновагу Неша в чистих стратегіях, використовуючи метод послідовного виключення домінованих стратегій та метод найкращих відповідей. Для кожної клітинки (i, j) перевірити чи є вона рівновагою Неша:

Умова рівноваги Неша: пара стратегій (S_i^*, S_j^*) є рівновагою Неша, якщо:

$$\pi_a(S_i, S_j) \geq \pi_a(S_i, S_j^{**}) \text{ для всіх стратегій } S_j \text{ компанії B}$$

$$\pi_b(S_i, S_j) \geq \pi_b(S_i^*, S_j) \text{ для всіх стратегій } S_i \text{ компанії A}$$

де π_a та π_b — функції виграшу компаній A та B відповідно.

4. Для кожної стратегії Компанії A знайти найкращу відповідь Компанії B (підкреслити максимальний виграш B в кожному рядку), для кожної стратегії Компанії B знайти найкращу відповідь Компанії A (підкреслити максимальний виграш A в кожному стовпці). Клітинки де обидва виграші підкреслені є рівновагами Неша. Якщо знайдено декілька рівноваг Неша, пояснити їх економічний зміст та який з них більш стабільний. Обчислити виграші обох компаній в рівноважній точці та порівняти з іншими можливими комбінаціями стратегій.

Контрольні питання

1. Опишіть основні поняття теорії ігор, пояснити що таке гравці, стратегії, функції виграшу (платіжні функції) в контексті формального визначення гри, розглянути класифікацію ігор за кількістю гравців (два гравці, n гравців), за характером взаємодії (кооперативні та некооперативні), за інформацією (повна та неповна інформація), за сумою виграшів (ігри з нульовою сумою, ігри з ненульовою сумою), пояснити різницю між стратегічною формою гри (матрична форма) та розгорнутою формою (дерево гри), навести приклади з практики ІТ-індустрії де застосовується теорія ігор.

2. Поясніть концепцію рівноваги Неша та її значення, описати інтуїтивний зміст рівноваги як ситуації де жоден гравець не має стимулу односторонньо змінювати свою стратегію, розглянути відмінність між рівновагою Неша та оптимумом Парето де неможливо покращити ситуацію одного гравця без погіршення іншого, пояснити чому може існувати декілька рівноваг Неша в одній грі та як гравці обирають між ними, описати проблему координації коли обидва гравці зацікавлені в досягненні певної рівноваги, розглянути теорему про існування рівноваги Неша в змішаних стратегіях для скінченних ігор.

3. Опишіть практичне застосування теорії ігор в ІТ-сфері, навести приклади моделювання конкуренції між технологічними компаніями за ринкові частки та стратегії ціноутворення.

Література

1. Lefeng Cheng, Mengya Zhang, Pengrong Huang, Wentian Lu. Game-Theoretic Approaches for Power-Generation Companies' Decision-Making in the Emerging Green Certificate Market. 2024. URL: <https://www.mdpi.com/2071-1050/17/1/71>

2. Paolo Renna. A Review of Game Theory Models to Support Production Planning, Scheduling, Cloud Manufacturing and Sustainable Production Systems. 2024. URL: <https://www.mdpi.com/2411-9660/8/2/26>

ТЕМА 8. ТЕОРІЯ ІГОР ТА ПРИЙНЯТТЯ РІШЕНЬ ЧАСТИНА 2

Мета роботи: набути практичних навичок розв'язання матричних ігор в змішаних стратегіях, освоїти методи знаходження оптимальних змішаних стратегій та очікуваних виграшів, навчитися застосовувати критерії прийняття рішень в умовах невизначеності, опанувати техніки побудови дерев рішень для послідовних ігор, розвинути розуміння застосування теорії ігор для моделювання багатокрокових стратегічних взаємодій.

Практичне завдання

1. Необхідно знайти оптимальні змішані стратегії для гри двох осіб з нульовою сумою. Два стартапи конкурують за інвестиції венчурного фонду, презентуючи свої продукти. Кожен стартап може обрати одну з двох стратегій презентації:

Стартап А: стратегія A_1 (акцент на технічних перевагах), стратегія A_2 (акцент на бізнес-моделі)

Стартап В: стратегія B_1 (агресивна критика конкурента), стратегія B_2 (фокус на власних перевагах)

2. Платіжна матриця для Стартапу А представлена у таблиці (ймовірність отримання інвестиції, виражена в процентах переваги):

Таблиця 1 Платіжна матриця для Стартапу А

	B_1	B_2
A_1	3	-2
A_2	-1	4

Знайти змішані стратегії: нехай Стартап А обирає стратегію A_1 з ймовірністю p та A_2 з ймовірністю $(1-p)$, Стартап В обирає стратегію B_1 з ймовірністю q та B_2 з ймовірністю $(1-q)$.

3. Обчислити очікувані виграші Стартапу А для кожної чистої стратегії Стартапу В:

$$E_1(p) = 3p + (-1)(1-p) = 4p - 1 \text{ — очікуваний виграш А, якщо В грає } B_1$$

$E_2(p) = (-2)p + 4(1-p) = 4 - 6p$ — очікуваний виграш А, якщо В грає B_2

Оптимальна змішана стратегія Стартапу А знаходиться з умови що Стартап В байдужий між своїми стратегіями:

$$E_1(p) = E_2(p)^{**}$$

$$4p - 1 = 4 - 6p^{**}$$

$$10p = 5^*$$

$$p = 0.5^*$$

Отже, Стартап А повинен обирати стратегію A_1 з ймовірністю 0.5 та A_2 з ймовірністю 0.5.

4. Аналогічно знайти оптимальну змішану стратегію для Стартапу В. Обчислити очікувані виграші Стартапу В (які є протилежними до виграшів А) для кожної чистої стратегії Стартапу А:

$$F_1(q) = -3q - (-2)(1-q) = -q - 2 \text{ — очікуваний виграш В, якщо А грає } A_1$$

$$F_2(q) = -(-1)q - 4(1-q) = 5q - 4 \text{ — очікуваний виграш В, якщо А грає } A_2$$

З умови байдужості для Стартапу А:

$$-3q + 2(1-q) = q^* - 4(1-q^*)^{**}$$

$$-3q + 2 - 2q = q^* - 4 + 4q^{***}$$

$$2 + 4 = q + 4q + 3q^* + 2q^{***}$$

$$6 = 10q^*$$

$$q = 0.6^*$$

Обчислити значення гри (очікуваний виграш для обох гравців при оптимальній грі):

$$V = E_1(p) = 4(0.5) - 1 = 1^*$$

Інтерпретувати результат: при оптимальній змішаній стратегії Стартап А має очікувану перевагу 1%, а Стартап В має очікуваний програш -1%.

Контрольні запитання

1. Опишіть концепцію змішаних стратегій в теорії ігор, пояснити чому гравці можуть використовувати рандомізацію при виборі стратегій замість детерміністичного вибору, розглянути теорему про мінімакс в іграх з

нульовою сумою яка гарантує існування значення гри та оптимальних змішаних стратегій, описати як обчислюється очікуваний виграш при грі в змішаних стратегіях, пояснити принцип байдужості який дозволяє знайти оптимальні ймовірності вибору стратегій, розглянути геометричну інтерпретацію змішаних стратегій як точок в симплексі.

2. Поясніть критерії прийняття рішень в умовах невизначеності, описати критерій Вальда (максимінний критерій) який максимізує мінімальний можливий виграш для песимістичного гравця, розглянути критерій максимаксу який максимізує максимальний можливий виграш для оптимістичного гравця, пояснити критерій Севіджа (мінімізація максимального жалю) де жаль це різниця між оптимальним виграшем та фактичним, описати критерій Гурвіца який балансує між песимізмом та оптимізмом через параметр α , розглянути критерій Лапласа який припускає рівноймовірність станів природи.

3. Опишіть дерева рішень та послідовні ігри, пояснити розгорнуту форму гри де гравці роблять ходи послідовно та мають інформацію про попередні ходи, розглянути метод зворотної індукції (backward induction) для знаходження рівноваги в довершених підіграх, описати концепцію довершеної рівноваги Неша (subgame perfect equilibrium) яка виключає недостовірні загрози, навести приклади застосування в багатокрокових стратегічних взаємодіях таких як переговори про зарплату, послідовне входження компаній на ринок, моделювання дилеми в'язня з повтореннями, розглянути застосування в розробці AI для ігор з послідовними ходами таких як шахи та го.

Література

1. Lefeng Cheng, Mengya Zhang, Pengrong Huang, Wentian Lu. Game-Theoretic Approaches for Power-Generation Companies' Decision-Making in the Emerging Green Certificate Market. 2024. URL: <https://www.mdpi.com/2071-1050/17/1/71>

2. Paolo Renna. A Review of Game Theory Models to Support Production Planning, Scheduling, Cloud Manufacturing and Sustainable Production Systems. 2024. URL: <https://www.mdpi.com/2411-9660/8/2/26>

ТЕМА 9. ТЕОРІЯ ГРАФІВ У ЗАДАЧАХ ОПТИМІЗАЦІЇ ЧАСТИНА 1

Мета роботи: набути практичних навичок моделювання задач оптимізації за допомогою графів, освоїти алгоритм Дейкстри для знаходження найкоротших шляхів від одного джерела до всіх інших вершин, навчитися реалізовувати алгоритм на практиці та аналізувати його складність, опанувати техніки представлення графів в пам'яті комп'ютера, розвинути розуміння застосування алгоритмів на графах в мережевій маршрутизації, навігаційних системах та соціальних мережах.

Практичне завдання

1. Необхідно знайти найкоротші шляхи від сервера-джерела до всіх інших серверів в мережі дата-центру. Мережа представлена зваженим орієнтованим графом $G = (V, E)$, де:

$V = \{S, A, B, C, D, E, F\}$ — множина серверів (вершин)

E — множина мережових з'єднань (ребер)

$w(u,v)$ — латентність з'єднання між серверами u та v в мілісекундах

Структура мережі ($u \rightarrow v$: латентність):

$S \rightarrow A$: 4 мс

$S \rightarrow B$: 2 мс

$A \rightarrow B$: 1 мс

$A \rightarrow C$: 5 мс

$B \rightarrow C$: 8 мс

$B \rightarrow D$: 10 мс

$C \rightarrow D$: 2 мс

$C \rightarrow E$: 6 мс

$D \rightarrow E$: 3 мс

$D \rightarrow F$: 7 мс

$E \rightarrow F$: 1 мс

2. Реалізувати алгоритм Дейкстри для знаходження найкоротших шляхів від джерела S до всіх інших вершин. Алгоритм використовує наступні структури даних:

$d[v]$ — поточна найкраща оцінка відстані від S до вершини v

$\pi[v]$ — попередник вершини v на найкоротшому шляху

Q — множина вершин, для яких найкоротша відстань ще не знайдена

Ініціалізація:

$d[S] = 0$

$d[v] = \infty$ для всіх $v \neq S$

$\pi[v] = \text{NULL}$ для всіх v

$Q = V$

3. Виконати ітерації алгоритму Дейкстри:

```
while  $Q \neq \emptyset$ :
```

```
     $u$  = вершина з  $Q$  з мінімальним  $d[u]$ 
```

```
    видалити  $u$  з  $Q$ 
```

```
    for кожної вершини  $v$  суміжної з  $u$ :
```

```
        if  $d[u] + w(u, v) < d[v]$ :
```

```
             $d[v] = d[u] + w(u, v)$       // релаксація ребра
```

```
             $\pi[v] = u$ 
```

Вести таблицю ітерацій з наступними стовпцями: номер ітерації, вибрана вершина u , оновлені відстані $d[v]$ для всіх вершин, множина Q невідвіданих вершин. На кожному кроці пояснювати які ребра релаксуються та як змінюються значення відстаней.

4. Після завершення алгоритму побудувати дерево найкоротших шляхів використовуючи масив попередників $\pi[v]$, для кожної вершини v відновити повний шлях від S до v проходячи назад по попередниках:

```
Path( $S, v$ ):
```

```
    if  $v = S$ :
```

```
        return [ $S$ ]
```

```
    else:
```

```
        return Path( $S, \pi[v]$ ) + [ $v$ ]
```

Вивести для кожної вершини: найкоротшу відстань від S , повний шлях від S , загальну латентність шляху. Візуалізувати граф з виділенням дерева найкоротших шляхів.

Контрольні запитання

1. Опишіть основні поняття теорії графів та способи представлення графів, пояснити різницю між орієнтованими та неорієнтованими графами зваженими та незваженими, розглянути способи представлення графів в пам'яті: матриця суміжності розміру $n \times n$ зі складністю пам'яті $O(n^2)$, список суміжності зі складністю $O(n+m)$ де m кількість ребер, матриця інцидентності розміру $n \times m$, порівняти переваги та недоліки кожного представлення для різних типів графів та операцій, пояснити поняття шляху простого шляху циклу зв'язності графа.

2. Пояснить алгоритм Дейкстри та умови його коректності, описати принцип жадібного вибору вершини з мінімальною поточною відстанню та операцію релаксації ребра, пояснити чому алгоритм коректний тільки для графів з невід'ємними вагами ребер та що відбувається при наявності від'ємних ваг, розглянути часову складність алгоритму: $O(n^2)$ при наївній реалізації з лінійним пошуком мінімуму, $O((n+m) \log n)$ при використанні бінарної купи для ефективного вибору мінімальної вершини, $O(n \log n + m)$ при використанні фібоначчієвої купи, описати можливі оптимізації та варіанти реалізації.

3. Опишіть практичне застосування алгоритмів найкоротших шляхів в ІТ-системах.

Література

1. Abdul Majeed, Ibtisam Rauf. Graph Theory: A Comprehensive Survey about Graph Theory Applications in Computer Science and Social Networks. 2020. URL: <https://www.mdpi.com/2411-5134/5/1/10>

2. Ireneusz Szcześniak, Andrzej Jajszczyk, Bożena Woźna-Szcześniak. Generic Dijkstra for optical networks. 2018. URL: <https://arxiv.org/abs/1810.04481>

ТЕМА 9. ТЕОРІЯ ГРАФІВ У ЗАДАЧАХ ОПТИМІЗАЦІЇ ЧАСТИНА 2

Мета роботи: набути практичних навичок розв'язання задачі комівояжера та задачі максимального потоку в мережі, освоїти евристичні методи для NP-важких задач на графах, навчитися реалізовувати алгоритм Форда-Фалкерсона для знаходження максимального потоку, опанувати техніки моделювання реальних задач у вигляді задач на графах, розвинути розуміння застосування алгоритмів теорії графів в логістиці, мережевих системах та оптимізації ресурсів.

Практичне завдання

1. Необхідно розв'язати задачу комівояжера (Traveling Salesman Problem, TSP) для планування маршруту технічного спеціаліста, який має відвідати всі філії компанії та повернутися до головного офісу. Є 6 локацій: головний офіс (O) та 5 філій (F1, F2, F3, F4, F5). Матриця відстаней між локаціями в кілометрах представлена у вигляді таблиці:

Таблиця 1 Матриця відстаней між локаціями

	O	F1	F2	F3	F4	F5
O	0	10	15	20	25	30
F1	10	0	35	25	30	20
F2	15	35	0	30	10	15
F3	20	25	30	0	15	12
F4	25	30	10	15	0	18
F5	30	20	15	12	18	0

Задача: знайти найкоротший гамільтонів цикл (маршрут що відвідує кожную вершину рівно один раз та повертається до початкової).

2. Застосувати жадібний евристичний алгоритм найближчого сусіда (Nearest Neighbor Heuristic):

Алгоритм найближчого сусіда:

1. Почати з початкової вершини O
2. `current = O, visited = {O}, tour = [O], total_distance = 0`

```

3. while існують невідвідані вершини:
    найти найближчу невідвідану вершину v до current:
        v = argmin{d(current, u) : u ∉ visited}
    додати v до tour та visited
    total_distance += d(current, v)
    current = v
4. повернутися до 0: total_distance += d(current, 0)
5. tour += [0]

```

Виконати алгоритм покроково, документуючи на кожній ітерації: поточну вершину, невідвідані вершини, обрану найближчу вершину, додану відстань, загальну відстань маршруту. Після завершення отримати маршрут та його загальну довжину.

3. Розв'язати задачу максимального потоку в мережі для оптимізації пропускної здатності дата-центру. Мережа представлена орієнтованим графом де ребра мають пропускні здатності (максимальна кількість даних що може передаватися в секунду, в Гбіт/с):

Вершини: S (джерело), A, B, C, D, T (стік)

Ребра з пропускними здатностями $c(u,v)$:

$S \rightarrow A$: 10 Гбіт/с

$S \rightarrow B$: 8 Гбіт/с

$A \rightarrow C$: 5 Гбіт/с

$A \rightarrow D$: 7 Гбіт/с

$B \rightarrow C$: 6 Гбіт/с

$B \rightarrow D$: 4 Гбіт/с

$C \rightarrow T$: 9 Гбіт/с

$D \rightarrow T$: 8 Гбіт/с

Застосувати алгоритм Форда-Фалкерсона для знаходження максимального потоку від S до T.

4. Реалізувати алгоритм Форда-Фалкерсона:

Ініціалізація: $f(u,v) = 0$ для всіх ребер (початковий потік)

while існує збільшувачий шлях P від S до T в залишковій мережі:

$cf(P) = \min\{c(u,v) - f(u,v) : (u,v) \in P\}$ // пропускна здатність шляху

```

for кожного ребра (u,v) в P:
    f(u,v) = f(u,v) + cf(P)      // збільшити потік
    f(v,u) = f(v,u) - cf(P)      // зменшити зворотний потік

max_flow =  $\sum f(S,v)$  для всіх v суміжних з S

```

Для кожної ітерації вести таблицю: номер ітерації, знайдений збільшуючий шлях, пропускна здатність шляху $cf(P)$, оновлені значення потоків на ребрах, поточне значення потоку. Використати пошук в ширину (BFS) або глибину (DFS) для знаходження збільшуючих шляхів. Після завершення визначити максимальний потік та мінімальний розріз мережі.

Контрольні запитання

1. Опишіть задачу комівояжера та її складність, пояснити чому TSP є NP-важкою задачею та що це означає для можливості знаходження оптимального розв'язку за поліноміальний час, розглянути точні методи розв'язання такі як повний перебір зі складністю $O(n!)$, метод гілок та меж (branch and bound), динамічне програмування алгоритм Хелда-Карпа зі складністю $O(n^2 2^n)$, описати евристичні методи які дають наближені розв'язки: жадібний алгоритм найближчого сусіда, алгоритм покращення 2-opt та 3-opt, генетичні алгоритми, розглянути гарантії якості апроксимаційних алгоритмів.
2. Поясніть задачу максимального потоку в мережі та алгоритм Форда-Фалкерсона, описати поняття залишкової мережі де для кожного ребра з потоком $f(u,v)$ існує залишкова пропускна здатність $cf(u,v) = c(u,v) - f(u,v)$ та зворотне ребро з $cf(v,u) = f(u,v)$.
3. Опишіть практичне застосування задач на графах в ІТ-системах.

Література

1. Abdul Majeed, Ibtisam Rauf. Graph Theory: A Comprehensive Survey about Graph Theory Applications in Computer Science and Social Networks. 2020. URL: <https://www.mdpi.com/2411-5134/5/1/10>
2. Ireneusz Szcześniak, Andrzej Jajszczyk, Bożena Woźna-Szcześniak. Generic Dijkstra for optical networks. 2018. URL: <https://arxiv.org/abs/1810.04481>

ТЕМА 10. СУЧАСНІ БІБЛІОТЕКИ ТА ІНСТРУМЕНТИ ОПТИМІЗАЦІЇ ЧАСТИНА 1

Мета роботи: набути практичних навичок використання спеціалізованих Python-бібліотек для розв'язання задач оптимізації, освоїти роботу з бібліотеками SciPy та PuLP, навчитися формулювати задачі оптимізації в термінах API бібліотек, опанувати техніки налаштування солверів та інтерпретації результатів, розвинути розуміння переваг використання готових інструментів оптимізації замість реалізації алгоритмів з нуля.

Практичне завдання

1. Необхідно розв'язати задачу оптимізації розподілу серверних ресурсів для хмарної платформи, використовуючи бібліотеку PuLP. Платформа надає три типи віртуальних машин (VM):

VM типу A: 2 vCPU, 4 GB RAM, вартість 50 грн/год

VM типу B: 4 vCPU, 8 GB RAM, вартість 90 грн/год

VM типу C: 8 vCPU, 16 GB RAM, вартість 160 грн/год

Клієнт замовив ресурси з наступними мінімальними вимогами: мінімум 50 vCPU, мінімум 100 GB RAM. Максимальна кількість кожного типу VM обмежена наявністю: максимум 10 VM типу A, максимум 8 VM типу B, максимум 6 VM типу C. Потрібно мінімізувати вартість при задоволенні вимог клієнта.

2. Встановити бібліотеку PuLP та створити модель оптимізації:

```
from pulp import *

# Створити проблему мінімізації
prob = LpProblem("VM_Allocation", LpMinimize)

# Змінні рішення (кількість VM кожного типу)
x_A = LpVariable("VM_type_A", lowBound=0, upBound=10, cat='Integer')
x_B = LpVariable("VM_type_B", lowBound=0, upBound=8, cat='Integer')
x_C = LpVariable("VM_type_C", lowBound=0, upBound=6, cat='Integer')
```

Визначити цільову функцію мінімізації вартості:

```

minimize: 50·x_A + 90·x_B + 160·x_C
# Цільова функція
prob += 50*x_A + 90*x_B + 160*x_C, "Total_Cost"

3. Додати обмеження до моделі:

Обмеження на vCPU:
 $2 \cdot x_A + 4 \cdot x_B + 8 \cdot x_C \geq 50$ 

Обмеження на RAM:
 $4 \cdot x_A + 8 \cdot x_B + 16 \cdot x_C \geq 100$ 

# Обмеження
prob += 2*x_A + 4*x_B + 8*x_C >= 50, "vCPU_requirement"
prob += 4*x_A + 8*x_B + 16*x_C >= 100, "RAM_requirement"

# Розв'язати задачу
prob.solve()

# Вивести результати
print(f"Статус: {LpStatus[prob.status]}")
print(f"Оптимальна кількість VM типу A: {value(x_A)}")
print(f"Оптимальна кількість VM типу B: {value(x_B)}")
print(f"Оптимальна кількість VM типу C: {value(x_C)}")
print(f"Мінімальна вартість: {value(prob.objective)} грн/год")

```

Використати бібліотеку SciPy для розв'язання задачі нелінійної оптимізації: мінімізація функції витрат на енергоспоживання серверного кластера. Споживання енергії залежить від завантаження серверів нелінійно:

$$E(x_1, x_2) = 100 + 5x_1^2 + 3x_2^2 + 2x_1x_2$$

де x_1 та x_2 — рівні завантаження двох серверів (від 0 до 100%).
 Обмеження: сумарне завантаження повинно забезпечити обробку навантаження:

$$x_1 + x_2 \geq 150$$

$$0 \leq x_1 \leq 100$$

$$0 \leq x_2 \leq 100$$

```

from scipy.optimize import minimize
import numpy as np

```

```

# Цільова функція
def energy_cost(x):
    x1, x2 = x
    return 100 + 5*x1**2 + 3*x2**2 + 2*x1*x2

# Обмеження-нерівність:  $x_1 + x_2 \geq 150 \Rightarrow -x_1 - x_2 + 150 \leq 0$ 
constraints = [{'type': 'ineq', 'fun': lambda x: x[0] + x[1] - 150}]

# Межі для змінних
bounds = [(0, 100), (0, 100)]

# Початкова точка
x0 = np.array([75, 75])

# Розв'язати
result = minimize(energy_cost, x0, method='SLSQP',
                  bounds=bounds, constraints=constraints)

print(f"Оптимальне завантаження: x1 = {result.x[0]:.2f}%, x2 = {result.x[1]:.2f}%")
print(f"Мінімальне споживання енергії: {result.fun:.2f} кВт")

```

Контрольні запитання

1. Опишіть архітектуру та можливості бібліотеки PuLP для лінійного програмування, пояснити як формуються задачі оптимізації через API бібліотеки включаючи створення змінних рішення цільової функції та обмежень, розглянути типи змінних які підтримуються: неперервні цілочисельні бінарні, описати вбудовані солвери такі як CBC (COIN-OR Branch and Cut) GLPK (GNU Linear Programming Kit) та можливість підключення комерційних солверів Gurobi CPLEX, пояснити як інтерпретувати результати розв'язання включаючи статус оптимальності значення змінних та цільової функції двоїсті змінні для аналізу чутливості.

2. Поясніть можливості модуля scipy.optimize для різних типів оптимізації, описати функції для мінімізації без обмежень minimize з методами Nelder-Mead BFGS CG Powell.

3. Розглянути функції для мінімізації з обмеженнями minimize з методом SLSQP (Sequential Least Squares Programming) COBYLA trust-constr, пояснити використання функцій для спеціалізованих задач: curve_fit для підгонки кривих linear_sum_assignment для задачі про призначення linprog для лінійного програмування, описати параметри налаштування алгоритмів такі як толерантність максимальна кількість ітерацій метод обчислення градієнта.

Література

1. Oscar Castro, Pierrick Bruneau, Jean-Sébastien Sottet, Dario Torregrossa. Landscape of High-performance Python to Develop Data Science and Machine Learning Applications. 2023. URL: <https://arxiv.org/abs/2302.03307>
2. Daniela Lopez-Betancur, Efrén González-Ramírez, Carlos Guerrero-Mendez, Tonatiuh Saucedo-Anaya, Martín Montes Rivera, Edith Olmos-Trujillo, Salvador Gomez Jimenez. Evaluation of Optimization Algorithms for Measurement of Suspended Solids. 2024. URL: <https://www.mdpi.com/2073-4441/16/13/1761>

ТЕМА 10. СУЧАСНІ БІБЛІОТЕКИ ТА ІНСТРУМЕНТИ ОПТИМІЗАЦІЇ ЧАСТИНА 2

Мета роботи: набути практичних навичок використання бібліотек NumPy та CVXPY для розв'язання задач опуклої оптимізації, освоїти методи чисельної оптимізації з використанням відкритих Python-бібліотек, навчитися візуалізувати результати оптимізації за допомогою Matplotlib, опанувати техніки порівняльного аналізу різних методів оптимізації, розвинути розуміння практичного застосування оптимізаційних інструментів для аналізу даних та машинного навчання.

Практичне завдання

1. Необхідно розв'язати задачу оптимізації портфелю акцій для мінімізації ризику при заданому рівні очікуваного прибутку, використовуючи бібліотеку CVXPY. Інвестор має можливість інвестувати в 4 акції технологічних компаній. Історичні дані показують наступні очікувані річні прибутки та ризики:

Акція А: очікуваний прибуток $\mu_1 = 12\%$, стандартне відхилення $\sigma_1 = 18\%$

Акція В: очікуваний прибуток $\mu_2 = 10\%$, стандартне відхилення $\sigma_2 = 15\%$

Акція С: очікуваний прибуток $\mu_3 = 8\%$, стандартне відхилення $\sigma_3 = 10\%$

Акція D: очікуваний прибуток $\mu_4 = 15\%$, стандартне відхилення $\sigma_4 = 25\%$

Матриця кореляцій між прибутками акцій:

Таблиця 1 Матриця кореляцій між прибутками акцій

	A	B	C	D
A	1.00	0.60	0.30	0.50
B	0.60	1.00	0.40	0.45
C	0.30	0.40	1.00	0.20
D	0.50	0.45	0.20	1.00

2. Встановити бібліотеки та підготувати дані:

```
import cvxpy as cp
import numpy as np
```

```

import matplotlib.pyplot as plt

# Очікувані прибутки (річні)
mu = np.array([0.12, 0.10, 0.08, 0.15])

# Стандартні відхилення
sigma = np.array([0.18, 0.15, 0.10, 0.25])

# Матриця кореляцій
corr_matrix = np.array([
    [1.00, 0.60, 0.30, 0.50],
    [0.60, 1.00, 0.40, 0.45],
    [0.30, 0.40, 1.00, 0.20],
    [0.50, 0.45, 0.20, 1.00]
])

# Обчислити коваріаційну матрицю
# Cov[i,j] =  $\sigma_i \cdot \sigma_j \cdot \rho_{ij}$ 
cov_matrix = np.outer(sigma, sigma) * corr_matrix

print("Коваріаційна матриця:")
print(cov_matrix)

```

3. Створити модель оптимізації портфелю з використанням CVXPY:

Змінні рішення: $w = (w_1, w_2, w_3, w_4)$ — частки інвестицій в кожну акцію.

Цільова функція (мінімізація ризику, вимірюного як варіанта портфелю):

minimize: $w^T \cdot \Sigma \cdot w$

де Σ — коваріаційна матриця

Обмеження:

Сума часток дорівнює 1: $\sum_i w_i = 1$

Невід'ємність часток: $w_i \geq 0$ для всіх i

Мінімальний очікуваний прибуток: $\mu^T \cdot w \geq r_target$

```

# Змінні рішення (частки портфелю)
w = cp.Variable(4)

# Цільова функція: мінімізувати ризик (варіансу портфелю)
portfolio_variance = cp.quad_form(w, cov_matrix)
objective = cp.Minimize(portfolio_variance)

# Обмеження

```

```

constraints = [
    cp.sum(w) == 1,          # Сума часток = 1
    w >= 0,                  # Невід'ємність
    mu @ w >= 0.11          # Мінімальний очікуваний прибуток 11%
]

# Створити та розв'язати задачу
problem = cp.Problem(objective, constraints)
problem.solve()

# Вивести результати
print(f"\nСтатус розв'язання: {problem.status}")
print(f"\nОптимальний портфель:")
for i in range(4):
    print(f"    Акція {chr(65+i)}: {w.value[i]*100:.2f}%")

portfolio_return = mu @ w.value
portfolio_risk = np.sqrt(problem.value)

print(f"\nОчікуваний прибуток портфелю: {portfolio_return*100:.2f}%")
print(f"Ризик портфелю (std dev): {portfolio_risk*100:.2f}%")

```

4. Побудувати ефективну границю (Efficient Frontier) — графік можливих комбінацій ризику та прибутку для різних значень цільового прибутку:

```

# Побудувати ефективну границю
target_returns = np.linspace(0.08, 0.15, 50)
risks = []
returns = []

for r_target in target_returns:
    # Змінити обмеження на мінімальний прибуток
    constraints_temp = [
        cp.sum(w) == 1,
        w >= 0,
        mu @ w >= r_target
    ]

    problem_temp = cp.Problem(objective, constraints_temp)
    problem_temp.solve()

```

```

        if problem_temp.status == 'optimal':
            risks.append(np.sqrt(problem_temp.value))
            returns.append(r_target)

# Візуалізація
plt.figure(figsize=(10, 6))
plt.plot(np.array(risks)*100, np.array(returns)*100, 'b-', linewidth=2,
label='Ефективна границя')
plt.scatter(portfolio_risk*100, portfolio_return*100, color='red',
s=100,
            marker='*', label='Оптимальний портфель (11% прибуток)',
zorder=5)

# Додати окремі акції на графік
for i in range(4):
    plt.scatter(sigma[i]*100, mu[i]*100, s=50, label=f'Акція
{chr(65+i)}')

plt.xlabel('Ризик (стандартне відхилення, %)', fontsize=12)
plt.ylabel('Очікуваний прибуток (%)', fontsize=12)
plt.title('Ефективна границя портфелю', fontsize=14, fontweight='bold')
plt.legend()
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.savefig('efficient_frontier.png', dpi=300, bbox_inches='tight')
plt.show()

print("\nГрафік збережено у файл 'efficient_frontier.png'")

```

5. Порівняти результати CVXPY з наївною стратегією рівномірного розподілу (25% на кожну акцію):

```

# Наївна стратегія: рівномірний розподіл
w_naive = np.array([0.25, 0.25, 0.25, 0.25])

naive_return = mu @ w_naive
naive_variance = w_naive @ cov_matrix @ w_naive
naive_risk = np.sqrt(naive_variance)

print("\n" + "="*50)
print("ПОРІВНЯННЯ СТРАТЕГІЙ:")
print("="*50)

```

```

print("\nНаївна стратегія (рівномірний розподіл):")
print(f"  Очікуваний прибуток: {naive_return*100:.2f}%")
print(f"  Ризик: {naive_risk*100:.2f}%")
print(f"  Співвідношення прибуток/ризик: {naive_return/naive_risk:.3f}")

print("\nОптимізована стратегія (CVXPY):")
print(f"  Очікуваний прибуток: {portfolio_return*100:.2f}%")
print(f"  Ризик: {portfolio_risk*100:.2f}%")
print(f"  Співвідношення прибуток/ризик: {portfolio_return/portfolio_risk:.3f}")

risk_reduction = (naive_risk - portfolio_risk) / naive_risk * 100
print(f"\nЗменшення ризику: {risk_reduction:.2f}%")

```

Контрольні запитання

1. Опишіть бібліотеку CVXPY та її призначення для опуклої оптимізації, пояснити що таке опукла оптимізація та чому вона важлива (гарантоване знаходження глобального оптимуму за поліноміальний час), розглянути синтаксис CVXPY для формулювання задач включаючи створення змінних `cp.Variable` цільових функцій `cp.Minimize` `cp.Maximize` обмежень, описати типи підтримуваних функцій: лінійні квадратичні норми експоненціальні логарифмічні та їх композиції, пояснити Disciplined Convex Programming (DCP) правила які забезпечують що задача залишається опуклою, розглянути вбудовані солвери ECOS SCS OSQP та їх автоматичний вибір залежно від типу задачі.

2. Поясніть задачу оптимізації портфелю Марковіца та її математичну постановку, описати концепцію ефективної границі як множини портфелів з оптимальним співвідношенням ризику та прибутку де неможливо зменшити ризик без зменшення прибутку, розглянути як коваріаційна матриця відображає залежності між активами та чому диверсифікація знижує ризик, пояснити різницю між систематичним ризиком (ринковий ризик який не можна диверсифікувати) та несистематичним ризиком (специфічний для активу який можна зменшити диверсифікацією), описати модифікації базової моделі: додавання обмежень на максимальну частку одного активу мінімальні

лоти трансакційні витрати короткі позиції, розглянути практичні обмеження моделі: припущення про нормальний розподіл прибутків використання історичних даних для прогнозування майбутнього нестійкість оцінок параметрів.

3. Опишіть роль візуалізації в аналізі результатів оптимізації та використання Matplotlib.

Література

1. Oscar Castro, Pierrick Bruneau, Jean-Sébastien Sottet, Dario Torregrossa. Landscape of High-performance Python to Develop Data Science and Machine Learning Applications. 2023. URL: <https://arxiv.org/abs/2302.03307>

2. Daniela Lopez-Betancur, Efrén González-Ramírez, Carlos Guerrero-Mendez, Tonatiuh Saucedo-Anaya, Martín Montes Rivera, Edith Olmos-Trujillo, Salvador Gomez Jimenez. Evaluation of Optimization Algorithms for Measurement of Suspended Solids. 2024. URL: <https://www.mdpi.com/2073-4441/16/13/1761>

Навчальне видання

**Юрій Гордійович Козуб
Ольга Олександрівна Смагіна**

**МАТЕМАТИЧНІ МЕТОДИ ОПТИМІЗАЦІЇ ТА ДОСЛІДЖЕННЯ
ОПЕРАЦІЙ**
**методичні вказівки до лабораторних робіт
для здобувачів вищої освіти першого (бакалаврського) рівня
спеціальності F3 – «Комп'ютерні науки»**

Комп'ютерне макетування – О. О. Смагіна
Коректор – Ю. Г. Козуб

Здано до склад. 23.01.2026 р. Підп. до друку 30.01.2026 р. Формат 60х84 1/16.
Папір офсет. Гарнітура Times New Roman. Друк ризографічний.
Ум. друк. арк. 1,5. Наклад 100 прим. Зам. № 132
Видавництво Державний заклад «Луганський національний університет імені Тараса
Шевченка»
м. Полтава, вул. Ковалю, 3
e-mail: mail@luguniv.edu.ua www/luguniv.edu.ua
Свідоцтво суб'єкта видавничої справи ДК № 3459 від 09.04.2009 р.
