

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

**Навчально-науковий інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу**

До захисту допущено
Завідувач кафедри
_____ Оксана ТИМОЩУК
« ____ » _____ 2025 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системний аналіз і управління»

спеціальності 124 «Системний аналіз»

**на тему: «Розподіл потоків в великих мережах промислового
призначення з оптимальним перерозподілом джерел постачання»**

Виконав:

Студент IV курсу, групи КА-12

Рудий Дмитро Ігорович _____

Керівник:

проф., доктор фіз.-мат. наук Касьянов Павло Олегович _____

Консультант з економічного розділу:

доцент, к.е.н., Рощина Надія Василівна _____

Консультант з нормоконтролю:

к.ф.-м.н. Статкевич Віталій Михайлович _____

Рецензент:

проф., доктор фіз.-мат. наук Капустян Олексій Володимирович _____

Засвідчую, що у цій дипломній
роботі немає запозичень з праць
інших авторів без відповідних
посилань.

Студент _____

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 124 «Системний аналіз»

Освітньо-професійна програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Оксана ТИМОЩУК

«__» _____ 2025 р.

ЗАВДАННЯ

на дипломну роботу студенту

Рудому Дмитру Ігоровичу

1. Тема роботи «Розподіл потоків в великих мережах промислового призначення з оптимальним перерозподілом джерел постачання», керівник роботи Касьянов Павло Олегович проф., доктор фіз.-мат. наук, затверджені наказом по університету від «26» травня 2025 р. № 1759-с.
2. Термін подання студентом роботи _____
3. Вихідні дані до роботи: Орієнтовані графи з вагами дуг і обмеженнями.
4. Зміст роботи: 1 – Дослідження предметної області; 2 – Теоретичні основи обраного математичного підходу; 3 – Практична реалізація алгоритму; 4 – Функціонально-вартісний аналіз програмного забезпечення.
5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): Ілюстрації роботи програмного продукту, презентація.

6. Консультанти розділів роботи.

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Рощина Н.В., доц., к.е.н.		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Аналіз предметної області та формулювання завдання	01.04.2025-13.04.2025	Виконано
2.	Вивчення літератури та обробка даних за темою роботи	14.04.2025-21.04.2025	Виконано
3.	Підготовка математичних моделей, розробка програмного коду їх реалізацій	22.04.2025-10.05.2025	Виконано
4.	Розробка програми для тестування методів	11.05.2025-20.05.2025	Виконано
5.	Оформлення та узгодження пояснювальної записки і розділів роботи	21.05.2025-01.06.2025	Виконано
6.	Попередній захист дипломної роботи	02.06.2025-15.06.2025	Виконано
7.	Захист дипломної роботи	16.06.2025-22.06.2025	Виконано

Студент

Дмитро РУДИЙ

Керівник

Павло КАСЬЯНОВ

РЕФЕРАТ

Дипломна робота: 88 с., 10 рис., 9 табл., 2 додатки, 18 джерел.

ГРАФИ, ПРОМИСЛОВІ ПОТОКИ, ОПТИМІЗАЦІЯ ПОТОКІВ

Об'єкт дослідження — процес розподілу ресурсів у мережах промислового типу з урахуванням структури графа, обмежень на потоки та динамічних характеристик джерел і споживачів.

Предмет дослідження — математичні методи оптимізації потоків у мережах, зокрема побудова двоїстої задачі на основі методу множників Лагранжа та алгоритми чисельного розв'язання з використанням потенціалів вузлів.

Мета роботи — розробка та реалізація математичної моделі оптимального розподілу потоків у мережах із частково змінними вузловими подачами, побудова відповідної двоїстої задачі та апробація ефективного алгоритму її розв'язання на тестових графах різної топології.

Актуальність — оптимальний розподіл потоків у мережах є критично важливим завданням у сферах енергетики, транспорту, водопостачання та інших інфраструктурних систем.

Результати роботи — розроблено програмний застосунок, що реалізує повний цикл оптимізації потоків у мережах на основі нелінійної моделі збереження розподілу потоків, аналізом втрат і збереження оптимізованих графів.

ABSTRACT

Thesis: 88 pages, 10 figures, 9 tables, 2 appendices, 18 references.

GRAPHS, INDUSTRIAL FLOWS, FLOW OPTIMIZATION

Object of research: the process of resource distribution in industrial-type networks, taking into account the graph structure, flow constraints, and dynamic characteristics of sources and consumers.

Subject of research: mathematical methods for optimizing flows in networks, in particular, the construction of the dual problem using the method of Lagrange multipliers and numerical solution algorithms based on node potentials.

Goal of work: to develop and implement a mathematical model for optimal flow distribution in networks with partially variable node supply, to construct the corresponding dual problem, and to test an effective algorithm for its solution on test graphs of various topologies.

Relevance: optimal flow distribution in networks is a critically important task in areas such as energy, transportation, water supply, and other infrastructure systems.

Results: a software application was developed that implements a full cycle of flow optimization in networks based on a nonlinear model, ensuring the conservation of flow distribution, analyzing losses, and saving optimized graphs.

ЗМІСТ

ВСТУП.....	8
РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ.....	10
1.1. Актуальність проблеми.....	10
1.2. Історія розв’язання подібних задач.....	11
1.3. Особливості розв’язування задач оптимізації потоків з використанням графів.....	14
1.4. Формалізація задачі з використанням графів.....	16
Висновки до розділу 1.....	17
РОЗДІЛ 2 ТЕОРЕТИЧНІ ОСНОВИ ОБРАНОГО МАТЕМАТИЧНОГО ПІДХОДУ.....	19
2.1. Загальні положення.....	19
2.2. Побудова двоїстої задачі та функції Лагранжа.....	20
2.3. Алгоритм чисельного розв’язання задачі.....	25
2.4. Структура тестових графів та постановка обчислювального експерименту.....	26
2.4.1. Формальна структура мережі.....	26
2.4.3. Узагальнені топологічні характеристики.....	28
Висновки до розділу 2.....	30
РОЗДІЛ 3 ПРАКТИЧНА РЕАЛІЗАЦІЯ АЛГОРИТМУ.....	32
3.1 Інструменти, які використовувались для розробки програмного застосунку.....	32
3.2 Генерація графів.....	34
3.3 Реалізація алгоритму оптимального розподілу потоків.....	38
3.5 Збереження та візуалізація результатів.....	41
3.6 Тестування результатів роботи програми.....	43

Висновки до 3 розділу.....	46
РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ РЕАЛІЗАЦІЇ ТА ПОСТІЙНОЇ РОБОТИ АЛГОРИТМУ ОПТИМІЗАЦІЇ ПОТОКІВ.....	49
4.1 Постановка задачі проектування.....	49
4.2 Обґрунтування функцій програмного продукту.....	50
4.3 Обґрунтування системи параметрів програмного продукту.....	54
4.4 Аналіз експертного оцінювання параметрів.....	57
4.5 Аналіз рівня якості варіантів реалізації функцій.....	61
4.6 Економічний аналіз варіантів розробки ПП.....	62
4.7 Вибір кращого варіанту ПП техніко-економічного рівня.....	68
Висновки до розділу 4.....	69
ВИСНОВКИ.....	70
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ.....	72
ДОДАТОК А ЛІСТИНГ КОДУ ПРОГРАМНОГО ПРОДУКТУ.....	74
ДОДАТОК Б ПРЕЗЕНТАЦІЯ.....	82

ВСТУП

У сучасних технічних, логістичних та енергетичних системах дедалі гострішою стає потреба в ефективному управлінні потоками ресурсів у складних розгалужених мережах. Водопостачання, електромережі, газотранспортні системи, транспортна логістика, телекомунікаційні інфраструктури — всі ці сфери функціонують на основі мережевих структур, у межах яких здійснюється передача ресурсів між джерелами та споживачами. Зростання масштабу систем, складності конфігурацій і динамічності змін вимагає застосування нових математичних методів, що здатні забезпечити оптимальне планування розподілу потоків у таких мережах. Особливої актуальності задача набуває в умовах обмежень — як технічних (наприклад, пропускна здатність трубопроводів чи кабелів), так і економічних (витрати на транспортування, енергоспоживання, втрати). У багатьох практичних випадках також необхідно враховувати змінні характеристики джерел і споживачів, що ускладнює класичні підходи до розв’язання подібних задач. У цій роботі розглядається задача оптимального розподілу потоків у мережах промислового типу з частково фіксованими та змінними правими частинами балансових рівнянь. Теоретичною основою дослідження є метод множників Лагранжа з переходом до двоїстої задачі, що дозволяє здійснювати оптимізацію за потенціалами вузлів. Такий підхід відкриває можливість ефективного чисельного розв’язання задач великої розмірності, зокрема за допомогою субградієнтних методів. Також розроблено генератор тестових графів із реалістичними характеристиками, що дозволяє перевірити працездатність методу на практиці. Об’єкт дослідження — процес розподілу ресурсів у мережах промислового типу з урахуванням структури графа, обмежень на потоки та динамічних характеристик джерел і споживачів. Предмет дослідження — математичні методи оптимізації потоків у мережах, зокрема побудова двоїстої задачі на основі методу множників Лагранжа та алгоритми чисельного розв’язання з

використанням потенціалів вузлів. Мета роботи — розробка та реалізація математичної моделі оптимального розподілу потоків у мережах із частково змінними вузловими подачами, побудова відповідної двоїстої задачі та апробація ефективного алгоритму її розв’язання на тестових графах різної топології.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Актуальність проблеми

Управління потоками в мережах стало фундаментальною задачею сучасного технічного, економічного й соціального середовища. З розвитком інформаційних технологій, інфраструктури міст, зростанням глобального товарообігу та інтеграції енергетичних систем світу, потреба в ефективному керуванні потоками ресурсів стрімко зростає. Оптимізація потоків — це не лише про зменшення витрат чи час доставлення. Це про стійкість інфраструктури, безпеку, адаптивність до змін попиту, можливість гнучко реагувати на кризові ситуації [1].

Енергетичний сектор потребує точного балансування генерації і споживання. В умовах децентралізації енергетичних джерел, таких як сонячні та вітрові установки, складність задачі суттєво зростає. Оптимізація потоків тут дозволяє зменшити втрати електроенергії, уникнути перевантаження мереж, зменшити витрати на виробництво (Kirschen & Strbac, 2004 [6]). В умовах енергетичної кризи або військової агресії такі задачі набувають ще більшого значення: система має бути здатна динамічно перенаправляти потоки між вузлами в обхід ушкоджених ділянок, забезпечуючи безперервність постачання.

У водопостачанні — задачі оптимального наповнення резервуарів, балансування тиску, мінімізації втрат через витоки. Навіть у розвинених країнах втрати води сягають 20–30% [9]. В Україні, особливо в сільській місцевості та промислових районах, ці показники можуть бути ще вищими. З огляду на дефіцит водних ресурсів у ряді регіонів, а також втрати через зношеність інфраструктури, оптимізація розподілу води є надзвичайно актуальною. Зокрема, ефективне моделювання дозволяє не лише знизити втрати, а й зменшити споживання енергії насосними станціями, що є додатковою перевагою.

У сфері логістики – планування маршрутів транспорту, мінімізація витрат пального, врахування обмежень на вантажність доріг, часових вікон доставки, робочих графіків водіїв. Це також стосується оптимізації постачання в умовах війни або гуманітарних місій. У постковідну епоху, коли зросла залежність від онлайн-торгівлі, логістичні виклики пов'язані з навантаженням на інфраструктуру, динамікою попиту та коливанням запасів стали ще складнішими [8]. Оптимізація потоків у логістиці дозволяє збалансувати завантаження складів, покращити планування маршрутів, уникнути дублювання постачання, зменшити час на доставку, а також знизити вуглецевий слід, що відповідає стратегіям сталого розвитку.

Оптимізація інформаційних потоків (в інтернеті, хмарних обчисленнях, дата-центрах) має значення не лише для продуктивності, а й для безпеки. Наприклад, блокування надлишкових потоків, оптимізація навантаження на маршрутизатори, забезпечення якості сервісу (QoS). З огляду на кіберзагрози, задача правильного розподілу потоків може також розглядатись у контексті виявлення атак типу DoS (Denial of Service) та побудови стійких до відмов архітектур.

Фінансові потоки — складна динамічна мережа платежів, банківських операцій, біржових транзакцій. Тут задачі оптимізації дозволяють покращити ліквідність, зменшити час клірингу, забезпечити дотримання регуляторних обмежень. У випадку кризових явищ або атак на банківську систему — можливість переадресації платежів через альтернативні канали має критичне значення для забезпечення стабільності ринку.

1.2. Історія розв'язання подібних задач

Перші концепції оптимізації потоків можна простежити ще в античні часи — управління водопостачанням у стародавньому Римі, спорудження зрошувальних систем у Межиріччі та Єгипті, військове планування

постачання у давніх арміях. Втім, наукове формулювання задачі відбулося лише у XX столітті.

У 1941 році Гітчкок [2] опублікував першу математичну постановку транспортної задачі. Його модель описувала доставку продукції з кількох джерел у кілька пунктів споживання з мінімальними витратами. У 1956 році Форд і Фалкерсон представили знаменитий алгоритм для задачі максимального потоку [7]. Ця робота стала основою теорії потоків у мережах. Їхня теорема «максимальний потік = мінімальний розріз» заклала зв'язок між оптимізацією потоку та структурою мережі.

Протягом 1960–70-х років задачі потоків знайшли застосування в телекомунікаціях, водогоні, плануванні виробництва. З'явилися нові алгоритми, серед яких метод Едмондса–Карпа, проштовхування передпотоків (Push-Relabel), модифікації симплекс-методу. У 1980-х роках Kennington та Helgason [11] узагальнили методи програмування для мереж.

Розвиток комп'ютерної техніки дав змогу моделювати задачі з десятками тисяч змінних. Це зробило оптимізацію потоків практичним інструментом для планування транспорту, енергомереж, логістики. Зокрема, великі корпорації типу FedEx або DHL використовували такі моделі для побудови маршрутів доставки з урахуванням обмежень на потужність складів, час доби, затори тощо.

З початку 1990-х задачі оптимізації потоків знайшли застосування в інтернет-інфраструктурі: маршрутизація, балансування навантаження, управління чергами, оптимізація трафіку в реальному часі. Зокрема, дослідження Tassiulas і Ephremides [12] заклали основи планування трафіку в бездротових мультимедійних мережах.

У 2000-х — бурхливий розвиток e-commerce і логістичних мереж, у тому числі в умовах глобалізації. Тут задачі оптимізації потоків дозволили зменшити логістичні витрати, оптимізувати об'єднані маршрути, враховувати сезонність попиту. Також активно досліджувались задачі в енергетичних

системах: оптимальне навантаження, баланс у Smart Grid, розподіл між зонами споживання та генерації.

У 2010-х роках — поширення задач в умовах невизначеності (стохастичні моделі), адаптивного управління (мультиагентні системи), поєднання з машинним навчанням (прогнозування попиту, часові ряди). У логістиці — інтеграція задач із блокчейн-реєстрами поставок, у фінансах — оптимізація платежів у реальному часі, у міському плануванні — побудова стійких до криз транспортних мереж [3].

Таким чином, історія задач поточкорозподілу демонструє безперервну еволюцію від теоретичних моделей до надзвичайно прикладних систем глобального масштабу. Залежно від контексту застосування змінювались методи, але залишалась спільна мета — ефективно, надійно й мінімальновитратно організувати рух ресурсів у складній системі.

Важливим етапом цієї еволюції є практичне застосування оптимізації потоків у сучасних технічних системах. Наприклад, у місті Суми реалізується проєкт із модернізації систем водовідведення, що передбачає реконструкцію очисних споруд і оптимізацію роботи насосних станцій для зменшення енерговитрат.

Ще один приклад — сільське водопостачання в межах проєкту DESPRO у Вінницькій області, де впроваджено регулювання тиску та облік споживання для забезпечення рівномірного розподілу води.

У газотранспортній галузі можна навести приклад Юліївського ЦВНГК, де оптимізовано експлуатацію газоконденсатних свердловин за рахунок побудови нових газопроводів, що знизили робочий тиск і підвищили ефективність видобутку.

Також заслуговує на увагу модель оптимізації роботи компресорних станцій у газових мережах, запропонована у роботі на платформі arXiv, що дозволяє мінімізувати витрати палива за дотримання граничних умов на тиск.

Крім того, дослідження EMPC (Economic Model Predictive Control) демонструє ефективність у керуванні насосними станціями у водопостачанні

з урахуванням змінних тарифів на електроенергію, що дозволяє мінімізувати витрати. Залежно від контексту застосування змінювались методи, але залишалось одне завдання — ефективно, надійно й мінімальновитратно організувати рух ресурсів у складній системі.

1.3. Особливості розв’язування задач оптимізації потоків з використанням графів

Графи є потужним математичним інструментом для моделювання та аналізу структурованих систем, де ресурси передаються від джерел до споживачів. Основною перевагою графового підходу є можливість відображення складної мережевої топології, зокрема різноманітних з’єднань, маршрутів, обмежень та пріоритетів. Графічна модель дозволяє описати мережу у вигляді набору вузлів (вершин) і зв’язків (дуг), де кожна дуга може мати атрибути, такі як пропускна здатність, вартість, втрати чи обмеження на напрямки руху.

Задачі оптимізації потоків традиційно формалізуються як задачі лінійного або нелінійного програмування з обмеженнями на баланс потоків у вузлах та пропускні спроможності на дугах. Серед найпоширеніших є задача максимального потоку, задача мінімального вартісного потоку, транспортна задача, стохастичні та динамічні моделі.

Для розв’язання таких задач розроблено велику кількість ефективних алгоритмів:

- 1) алгоритм Форда–Фалкерсона (максимальний потік);
- 2) алгоритм Едмондса–Карпа (модифікація Форда–Фалкерсона із використанням BFS);
- 3) алгоритм проштовхування передпотіку (Push-Relabel);
- 4) мережевий симплекс-метод (для задач мінімального вартісного потоку);

5) алгоритм Дейкстри, Беллмана–Форда (для пошуку найкоротшого шляху у зважених графах);

6) алгоритми для багатотоварних потоків (розділення задач на незалежні підзадачі або використання двоїстих методів).

У графових задачах також широко застосовуються методи динамічного програмування, методи множників Лагранжа, а в складних випадках — еволюційні алгоритми (генетичні, рої часток) та методи глибокого навчання.

У практиці технічних систем графові моделі застосовуються для таких прикладів:

1. Водопостачання – побудова та оптимізація схем насосних станцій, резервуарів, розгалужень трубопроводів. Для прикладу, у Сумах в межах програми НЕФКО реалізовано модернізацію системи водовідведення із графовою оптимізацією насосних станцій .
2. Газорозподіл – оптимізація тиску на мережах, маршрутизація газових потоків з урахуванням втрат, моделювання режимів компресорних станцій. Приклад: геометрична оптимізація на компресорних станціях у дослідженні з arXiv.
3. Енергосистеми – розподіл навантажень у Smart Grid, балансування активної і реактивної потужності між генераторами і споживачами. Приклад — використання методів LMP (Locational Marginal Pricing) для розподілу потужності в енергомережі [6].
4. Транспорт і логістика – побудова маршрутів, уникнення перевантажень, оптимізація вікон доставки. У FedEx та UPS щодня вирішуються задачі з мільйонами обмежень за допомогою графових моделей.
5. IT-інфраструктура – маршрутизація трафіку, резервування каналів, побудова топологій дата-центрів. Багато сучасних протоколів маршрутизації базуються на графах (OSPF, BGP).

Реальні задачі рідко є статичними. Тому використання графів дозволяє не лише знайти одне оптимальне рішення, а й оновлювати його у відповідь на зміну умов (наприклад, аварія, зміна попиту, оновлення конфігурації).

Візуалізація мереж за допомогою графів є додатковим інструментом для прийняття рішень: наочне подання структури допомагає виявити вузькі місця, надлишкові зв'язки, залежності.

Отже, графи є фундаментальним інструментом у сучасному арсеналі інженера або аналітика, який працює із задачами потоків: від простого водогону до динамічної системи постачання електроенергії чи керування логістичною мережею у реальному часі.

1.4. Формалізація задач з використанням графів

При формалізації задач оптимального розподілу потоків у мережах важливо надати точне абстрактне представлення технічної системи, яка складається з джерел (колодязів), споживачів, каналів передачі ресурсу та обмежень, які регулюють ці потоки. Одним із найбільш зручних і загальноприйнятих підходів є подання системи у вигляді орієнтованого графа $G = (N, V)$, де:

N — множина вершин, що представляють елементи інфраструктури: джерела, споживачі, проміжні вузли;

V — множина дуг (ребер), що моделюють можливі напрямки передачі потоку між вузлами.

У цій моделі джерела (колодязі) — це вузли, які мають надлишок ресурсу, який потрібно передати в систему. Наприклад, у водопостачанні це насосні станції чи водозабори; у газопостачанні — газові свердловини або компресорні станції. Для кожного джерела і визначено параметр , який означає обсяг ресурсу, що постачається.

Споживачі — це вузли, що поглинають ресурс. Вони моделюються вершинами з параметром , який визначає потребу у ресурсі. Наприклад, це житлові квартали, промислові підприємства або об'єкти критичної інфраструктури.

Можуть також бути присутніми проміжні вузли, які не виробляють і не споживають ресурсу, а лише передають його далі (транзитні точки). Для них встановлюється умова .

Ребра графа (i, j) характеризують наявність можливого потоку від вузла i до вузла j . Кожне ребро має наступні характеристики:

- 1) пропускна здатність — максимальна кількість ресурсу, яку можна передати через канал;
- 2) можливі втрати або ефективність передачі, яка враховується у вигляді коефіцієнтів або обмежень;
- 3) вартість передачі — у задачах економічної оптимізації.

Ціль задачі формулюється як досягнення мінімізації сумарних витрат на транспортування, втрат, енергії або часу — залежно від конкретної галузі. У рамках загальної постановки це завдання може бути зведене до задачі лінійного програмування, але без занурення у математичний апарат цілком достатньо розуміти логіку: знайти такий розподіл потоків по наявних каналах, який задовольняє попит усіх споживачів і не перевищує можливості джерел та каналів.

Цей підхід дозволяє адаптувати модель до будь-якої галузі: у водоканалі — це може бути граф із насосами та водорозподільними станціями; у логістиці — граф із постачальниками, складами, транспортом; у енергетиці — генераторами, трансформаторами, лініями електропередачі.

Формалізація задачі через граф дозволяє моделювати ситуації з надлишковим або дефіцитним постачанням, балансувати потоки між різними шляхами, аналізувати критичні ділянки, розглядати сценарії аварій або модернізацій.

Висновки до розділу 1

У цьому розділі було розглянуто ключові теоретичні аспекти задач оптимального розподілу потоків у мережах. Було встановлено, що

актуальність подібних задач обумовлена зростанням складності сучасних технічних систем, потребою в ефективному використанні ресурсів, а також необхідністю забезпечення стабільності та надійності функціонування мереж різної природи: енергетичних, транспортних, водопровідних, телекомунікаційних тощо.

Історичний огляд показав, що розвиток методів оптимізації потоків пройшов шлях від класичних математичних моделей до сучасних гнучких і високоефективних алгоритмів, які дозволяють вирішувати задачі великих розмірностей із врахуванням множинних обмежень. У розділі також було наведено приклади з реального життя, які засвідчують успішне застосування теорії потоків у водопостачанні, газопостачанні, логістиці та інших сферах.

Особливу увагу було приділено графовій формалізації, яка є універсальним інструментом моделювання мережевих систем. Графовий підхід дозволяє наочно описати структуру системи, формалізувати обмеження та цілі, що робить його ефективним засобом як для аналітичного аналізу, так і для комп'ютерного моделювання.

Таким чином, теоретичні засади, викладені в цьому розділі, закладають міцне підґрунтя для подальшого практичного дослідження і реалізації методів оптимального розподілу потоків у великих промислових мережах із динамічним перерозподілом джерел постачання.

РОЗДІЛ 2 ТЕОРЕТИЧНІ ОСНОВИ ОБРАНОГО МАТЕМАТИЧНОГО ПІДХОДУ

2.1. Загальні положення

Задача оптимального розподілу потоків у мережі промислового призначення є важливою через значну кількість прикладних задач у промисловості, інфраструктурі та енергетиці. У великих розподільчих системах виникає потреба не просто передати ресурс з джерел до споживачів, а зробити це з урахуванням численних технічних, економічних та фізичних обмежень. Такі задачі охоплюють багато сфер — від газорозподільчих і водопровідних мереж до логістичних та енергетичних систем.

У класичному вигляді мережа подається як орієнтований граф $G = (N, V)$, де:

N — множина вершин (вузлів), кожна з яких моделює або джерело, або споживача, або проміжний транзитний пункт;

V — множина дуг (ребер), кожна з яких визначає можливий напрям руху потоку між вузлами.

Кожна дуга $(i, j) \in V$ характеризується змінною x_{ij} — кількістю ресурсу, що передається від вузла i до j . На змінну x_{ij} накладаються обмеження: $r_i^- j \leq x_{ij} \leq r_i^+ j$, де:

$r_i^- j$ — нижня межа потоку (може дорівнювати нулю або бути позитивною, залежно від технології);

$r_i^+ j$ — верхня межа потоку, яка відображає технічну пропускну здатність дуги.

У кожному вузлі $i \in N$ вводиться параметр d_i , який відповідає.

- 1) джерелу ($d_i > 0$);
- 2) споживачу ($d_i < 0$);
- 3) транзитному вузлу ($d_i = 0$).

Баланс потоків у вузлі вираховується за формулою (2.1):

$$\sum_{j:(i,j) \in V} x_{ij} - \sum_{j:(j,i) \in V} x_{ji} = d_i \quad (2.1)$$

Цільова функція задачі — мінімізація сумарних витрат транспортування по мережі. Для кожної дуги задається функція вартості $f_{ij}(x)$, яка є:

- 1) строго зростаючою (витрати зростають із обсягом потоку);
- 2) неперервною;
- 3) задовольняє умову $f_{ij}(0) = 0$.

Загальна функція витрат вираховується за формулою (2.2):

$$F(x) = \sum_{(i,j) \in V} \int_0^{x_{ij}} f_{ij}(t) dt \quad (2.2)$$

У статті Кірік О.Є. [18] запропоновано модель, де частина вузлових значень d_i є змінними — тобто допускається оптимізувати подачу з кількох джерел у рамках допустимого інтервалу. Це ускладнює задачу, бо додає змінні до правих частин балансових рівнянь. Таким чином, ми маємо справу із задачею оптимізації:

- 1) із обмеженнями балансу у вузлах;
- 2) з обмеженнями на змінні потоку по дугах;
- 3) із частково фіксованими та частково змінними вузовими подачами.

2.2. Побудова двоїстої задачі та функції Лагранжа

Для розв'язання задачі ефективно використовується метод множників Лагранжа, який дозволяє перенести обмеження у цільову функцію і перейти до задачі без обмежень, яка краще придатна для чисельного аналізу.

Згадана вище задача має цільову функцію, яка вираховується за формулою (2.3):

$$F(x) = \sum_{(i,j) \in V} \int_0^{x_{ij}} f_{ij}(t) dt \quad (2.3)$$

і систему балансових обмежень, яка визначається за формулою (2.4):

$$\sum_{j: (i,j) \in V} x_{ij} - \sum_{j: (j,i) \in V} x_{ji} = d_i \quad (2.18)$$

Введемо множники Лагранжа p_i для кожного балансового рівняння.

Тоді функція Лагранжа вираховується за формулою (2.5):

$$L(x, p) = \sum_{(i,j) \in V} \int_0^{x_{ij}} f_{ij}(t) dt + \sum_{i \in N} p_i \left(d_i - \sum_{j: (i,j) \in V} x_{ij} + \sum_{j: (j,i) \in V} x_{ji} \right) \quad (2.5)$$

Після перетворення отримаємо рівність, яка вираховується за формулою (2.6):

$$L(x, p) = \sum_{(i,j) \in V} \left(\int_0^{x_{ij}} f_{ij}(t) dt - (p_i - p_j)x_{ij} \right) + \sum_{i \in N} p_i d_i \quad (2.6)$$

Побудуємо двоїсту функцію яка вираховується за формулою (2.7):

$$\Phi(p) = \inf_{x \in [r_{ij}^-, r_{ij}^+]} L(x, p) \quad (2.7)$$

Це означає, що ми зводимо задачу до знаходження такого вектора p , при якому ця функція досягає максимуму вираховується за формулою (2.8):

$$\max_{p \in \mathbb{R}^N} \Phi(p) \quad (2.8)$$

У цьому формулюванні обмеження на x_{ij} враховуються під час обчислення $\inf$, а оптимальні значення x_{ij} можна відновити із правила, яке вираховується за формулою (2.9):

$$x_{ij}(p) = \arg \min_{x \in [r_{ij}^-, r_{ij}^+]} \left(\int_0^x f_{ij}(t) dt - (p_i - p_j)x \right) \quad (2.9)$$

Таким чином, ми маємо компакту задачу лише за змінними p — потенціалами вузлів, яку можна ефективно розв'язувати чисельними методами.

Двоїста задача:

- 1) є опуклою навіть при неопуклості $f_{ij}(t)$;
- 2) добре масштабується на великі графи;
- 3) дозволяє паралельну реалізацію на вузлах.

Цей підхід відкриває шлях до ефективного розв'язання задач розподілу потоків у мережах з багатьма змінними вузлами.

Для ефективного розв'язання задачі використовують підхід, заснований на побудові двоїстої задачі за допомогою множників Лагранжа. Це дозволяє зменшити розмірність та перейти до оптимізації по потенціалах вузлів.

Цільова функція вираховується за формулою (2.10):

$$F(x) = \sum_{(i,j) \in V} \int_0^{x_{ij}} f_{ij}(t) dt \quad (2.10)$$

Обмеження збереження потоку у вузлах вираховується за формулою (2.11):

$$\sum_{j: (i,j) \in V} x_{ij} - \sum_{j: (j,i) \in V} x_{ji} = d_i \quad (2.11)$$

Вводимо множники Лагранжа p_i для кожного обмеження. Функція Лагранжа тоді вираховується за формулою (2.12):

$$L(x, p) = F(x) + \sum_{i \in N} p_i \left(d_i - \sum_{j: (i,j) \in V} x_{ij} + \sum_{j: (j,i) \in V} x_{ji} \right) \quad (2.12)$$

Перегрупуємо члени (див. формулу (2.13)):

$$L(x, p) = \sum_{(i,j) \in V} \left(\int_0^{x_{ij}} f_{ij}(t) dt - (p_i - p_j) x_{ij} \right) + \sum_{i \in N} p_i d_i \quad (2.13)$$

Визначаємо двоїсту функцію (див. формулу (2.14)):

$$Phi(p) = \inf_{x_{ij} \in [r_{ij}^-, r_{ij}^+]} L(x, p) \quad (2.14)$$

Тоді двоїста задача набуває вигляду (див. формулу (2.16)):

$$\max_{p \in \mathbb{R}^N} \Phi(p) \quad (2.15)$$

Це задача максимізації опуклої функції без явних обмежень, що дозволяє застосовувати градієнтні та субградієнтні методи. Знайшовши

потенціали p , можна відновити оптимальні потоки за правилом (див. формулу (2.13)):

$$x_{ij}(p) = \arg \min_{x \in [r_{ij}^-, r_{ij}^+]} \left(\int_0^x f_{ij}(t) dt - (p_i - p_j)x \right) \quad (2.16)$$

Таким чином, двоїстий підхід дозволяє розв'язати вихідну задачу в два етапи:

- 1) знаходження оптимальних потенціалів p_i ;
- 2) обчислення потоку x_{ij} по кожній дузі.

Цей підхід добре масштабується, ефективний для великих мереж і дозволяє працювати з реалістичними обмеженнями.

Для аналізу та розв'язання задачі оптимального розподілу потоків з частково змінними правими частинами використовується підхід на основі побудови двоїстої задачі. Це дозволяє перейти до оптимізації по потенціалах вузлів, що спрощує обчислення в багатьох практичних випадках.

Розглянемо цільову функцію, яка вираховується за формулою (2.17):

$$F(x) = \sum_{(i,j) \in V} \int_0^{x_{ij}} f_{ij}(t) dt \quad (2.17)$$

де $f_{ij}(t)$ — строго зростаючі, неперервні функції витрат.

Накладаються обмеження неперервності у вузлах, які вираховується за формулою (2.18):

$$\sum_{j: (i,j) \in V} x_{ij} - \sum_{j: (j,i) \in V} x_{ji} = d_i \quad (2.18)$$

для всіх $i \in N$,

де d_i — фіксовані або змінні праві частини.

Щоб сформулювати двоїсту задачу, будується функція Лагранжа, яка вираховується за формулою (2.19):

$$L(x, p) = \sum_{(i,j) \in V} \int_0^{x_{ij}} f_{ij}(t) dt + \sum_{i \in N} p_i \left(d_i - \sum_{j: (i,j) \in V} x_{ij} + \sum_{j: (j,i) \in V} x_{ji} \right) \quad (2.19)$$

де p_i — множники Лагранжа (інтерпретуються як потенціали вузлів).

Функція Лагранжа перегрупується в іншу, яка вираховується за формулою (2.20):

$$L(x, p) = \sum_{(i,j) \in V} \left(\int_0^{x_{ij}} f_{ij}(t) dt - (p_i - p_j)x_{ij} \right) + \sum_{i \in N} p_i d_i \quad (2.20)$$

Тепер вводиться двоїста функція, яка вираховується за формулою (2.21):

$$\Phi(p) = \inf_{x_{ij} \in [r_{ij}^-, r_{ij}^+]} L(x, p) \quad (2.21)$$

де обмеження $\forall(i, j) : x_{ij} \in [r_{ij}^-, r_{ij}^+]$.

Таким чином, двоїста задача набуває вигляду, який вираховується за формулою (2.22):

$$\max_{p \in \mathbb{R}^N} \Phi(p). \quad (2.22)$$

Двоїста задача є опуклою незалежно від строгої опуклості вихідної задачі, оскільки $\Phi(p)$ визначається як нижня огибаюча сімейства лінійних функцій по p .

Ключовий момент — можливість відновити оптимальні значення прямих змінних x_{ij} із знайденого p через умовний мінімум, який вираховується за формулою (2.23):

$$x_{ij}(p) = \arg \min_{x_{ij} \in [r_{ij}^-, r_{ij}^+]} \left(\int_0^{x_{ij}} f_{ij}(t) dt - (p_i - p_j)x_{ij} \right). \quad (2.23)$$

Ця формула дозволяє обчислити оптимальний потік по дузі, знаючи лише різницю потенціалів вузлів.

Таким чином, завдяки двоїстому підходу задача розподілу потоків зводиться до неklasичної задачі без обмежень по p , розв'язуваної чисельними методами. Такий підхід масштабований і дозволяє ефективно обробляти великі мережі з десятками тисяч вузлів. опуклих задач нелінійного програмування з частково змінними правими частинами рівнянь неперервності.

У наступних підрозділах буде детально викладено метод побудови двоїстої задачі, формування функції Лагранжа, умови оптимальності, а також конкретні алгоритми чисельного розв'язання, з прикладами.

2.3. Алгоритм чисельного розв'язання задачі

Після побудови двоїстої задачі та вираження цільової функції у термінах потенціалів вузлів p_i наступним етапом є розробка ефективного чисельного алгоритму. Для задач подібного класу доцільно використовувати метод градієнтного спуску або субградієнтного спуску, з урахуванням специфіки функції $\Phi(p)$, яка є опуклою, але не завжди диференційованою.

Загальна ідея алгоритму:

Ініціалізація: обрати початковий вектор потенціалів p^0 , зазвичай $p^0 = 0$.

На кожній ітерації t : а. Для кожної дуги $(i, j) \in V$ обчислити x_{ij}^t за формулою (2.24):

$$x_{ij}^t = \arg \min_{x \in [r_{ij}^-, r_{ij}^+]} \left(\int_0^x f_{ij}(t) dt - (p_i^t - p_j^t)x \right) \quad (2.24)$$

Обчислити субградієнт функції Φ у точці p^t як вектор g^t , (див. (2.25))

$$g_i^t = d_i - \sum_{j: (i,j) \in V} x_{ij}^t + \sum_{j: (j,i) \in V} x_{ji}^t \quad (2.25)$$

Оновити потенціали за правилом: $p^{t+1} = p^t + \alpha_t \cdot g^t$, де t — крок ітерації, який може бути сталим або зменшуваним приклад

$$\alpha_t = \frac{\alpha_0}{\sqrt{t}} \quad (2.26)$$

Повторювати ітерації до виконання умови зупинки: $\|g^t\| < \varepsilon$, де ε — задана точність.

Цей алгоритм ґрунтується на тому, що мінімум функції Лагранжа по x_{ij} визначається поелементно для кожної дуги (i, j) , що дозволяє ефективно паралелізувати обчислення на великих мережах.

Особливості реалізації:

- 1) вибір початкових потенціалів не є критичним — зазвичай береться нульовий вектор.
- 2) крок t повинен зменшуватись достатньо повільно, щоб гарантувати збіжність.
- 3) для строго опуклих $f_{ij}(t)$ алгоритм забезпечує глобальну збіжність.
- 4) за потреби можна застосовувати метод прискореного градієнтного спуску (наприклад, метод Нестерова).

Переваги підходу:

- 1) висока масштабованість: складність одного кроку лінійна від кількості дуг;
- 2) можливість обчислення в розподіленому режимі
- 3) застосовність до задач з частково відомими d_i

У наступному підрозділі буде наведено приклад практичного застосування алгоритму до мережі з кількома джерелами і споживачами.

2.4. Структура тестових графів та постановка обчислювального експерименту

2.4.1. Формальна структура мережі

Для моделювання і тестування алгоритмів оптимізації потоків у мережах використано генератор графів, який дозволяє створювати різні типи мереж із заданими характеристиками. Це дає змогу досліджувати поведінку

алгоритмів в умовах, що максимально наближені до реальних мереж промислового призначення.

Кожна мережа формується у вигляді орієнтованого графа $G = (N, V)$, де:

N — множина вершин, які поділяються на:

джерела (well) — вузли, що подають ресурс у мережу, мають змінні величини подачі;

споживачі (consumer) — вузли, що поглинають потік згідно з фіксованим або випадковим попитом;

проміжні вузли — не мають споживання або генерації, виконують функцію передавання потоку.

V — множина дуг (i, j) , кожна з яких характеризується:

ваговим коефіцієнтом l_{ij} (вартість потоку);

мінімальною та максимальною пропускною здатністю

Для збереження балансу система автоматично визначає частку кожного джерела в загальній подачі залежно від суми споживання та кількості джерел. Орієнтація ребер визначається так, щоб усі споживачі були досяжними від хоча б одного джерела.

2.4.2. Типи графів

В даному пункті розглядаються чотири типи графів. Першим пунктом є Watts-Strogatz Graph (WS-граф)

Цей граф формується з n вершин, з'єднаних у кільце, де кожна вершина з'єднана з k найближчими сусідами. Потім кожне ребро з імовірністю p перемикається на інший випадковий вузол. Такий підхід поєднує локальну зв'язність (низький діаметр) з випадковими переходами, що забезпечує невелику довжину найкоротших шляхів. У реальних задачах це відповідає

мережам регіонального постачання або інфраструктурним системам із резервними маршрутами.

- 1) число вершин: n ;
- 2) кожна вершина з'єднана з $k/2$ сусідами зліва та справа.
- 3) p — ймовірність перемикавання ребра на випадкову вершину

Другим типом є Grid Graph (гратчаста сітка)

Формується як двовимірна решітка з $R \times C$ вузлів, де кожен вузол має до 4 сусідів (вгору, вниз, вліво, вправо). Використовується для моделювання міських систем — водопровідів, вуличної інфраструктури, електричних мереж з регулярною топологією. Характеризується передбачуваністю та можливістю маршрутизації потоків за географічним принципом.

- 1) вузли: (i,j) , де $i \in [1,R]$, $j \in [1,C]$;
- 2) зв'язки: між сусідніми координатами.

Третім типом є Tree Graph (ієрархічна структура)

Мережа будується як об'єднання кількох збалансованих дерев з параметрами:

- 1) r — кількість дітей на кожному рівні (ступінь);
- 2) h — висота дерева.

Дерева об'єднуються між собою випадковими додатковими ребрами. Відповідає структурі водопровідних або енергетичних мереж, де є чіткий потік від джерела до споживача. У такій мережі потоки природно рухаються по гілках дерева.

Random Connected Graph

Цей тип формується шляхом створення випадкового дерева з n вершин, яке потім доповнюється m додатковими випадковими ребрами для з'єднання різних компонент. Такий граф моделює мережі з надмірною зв'язністю, резервними маршрутами або неконтрольованим нарощенням інфраструктури.

- 1) базове дерево: гарантує зв'язність;

2) m додаткових ребер додаються випадково без дублікатів.

2.4.3. Узагальнені топологічні характеристики

Watts–Strogatz граф (WS-граф)

Цей тип поєднує регулярність і випадковість: кожен вузол має зв'язок з найближчими сусідами, а деякі ребра випадково перемикаються.

1. Щільність – середня; завдяки регулярній структурі з випадковими з'єднаннями граф не перевантажений, але добре зв'язаний.
2. Діаметр – малий, дозволяє будь-якій парі вершин бути з'єднаними коротким шляхом.
3. Цикли – велика кількість як коротких, так і довших циклів; мережа добре переплетена.
4. Надмірність маршрутів – висока, навіть при втраті кількох ребер залишаються альтернативні шляхи.

Random Connected Graph (випадковий зв'язний граф)

Формується на основі випадкового дерева з додаванням нових випадкових ребер для забезпечення зв'язності.

1. Щільність – середня або помірно висока — залежно від кількості додаткових ребер після побудови дерева.
2. Діаметр – невеликий; наявність випадкових зв'язків суттєво скорочує відстань між вершинами.
3. Цикли – багато; випадкові додаткові ребра утворюють велику кількість циклів довільної довжини.
4. Надмірність маршрутів – висока; існує багато обхідних шляхів, що підвищує стійкість мережі.

Grid Graph (граф-сітка)

Побудований як регулярна двовимірна решітка — кожен вузол з'єднаний з географічно близькими сусідами.

- 1) Щільність – регулярна; кожен вузол (окрім країв) має однакову кількість з'єднань, що створює передбачувану структуру.
- 2) Діаметр – помірний; із зростанням кількості вузлів збільшується рівномірно, але залишає мережу зручною для маршрутизації.
- 3) Цикли – присутні системно — квадратні, прямокутні; сітка дозволяє формувати локальні петлі.
- 4) Надмірність маршрутів – середня до високої; можливі обхідні шляхи, особливо в центральній частині мережі.

Tree Graph (дерево)

Структура без циклів, що моделює ієрархічні або послідовні потоки — від джерел до споживачів.

- 1) Щільність – низька; лише мінімально необхідна кількість ребер для з'єднання всіх вузлів
- 2) Діаметр – високий; у найгіршому випадку шлях між крайніми вузлами проходить через усю мережу.
- 3) Цикли – відсутні повністю, дерева за визначенням є ациклічними структурами.
- 4) Надмірність маршрутів – відсутня; кожен вузол з'єднаний лише одним шляхом, що робить структуру вразливою до збоїв.

Висновки до розділу 2

У другому розділі було сформульовано математичну постановку задачі оптимального розподілу потоків у промисловій мережі з частково змінними правими частинами. На основі цієї постановки обґрунтовано застосування двоїстого підходу, який дозволяє перейти до задачі оптимізації за потенціалами вузлів без явних обмежень. Це, своєю чергою, значно спрощує чисельну реалізацію задачі в умовах великомасштабних графів.

Побудована функція Лагранжа та відповідна двоїста функція дозволяють ефективно використовувати методи градієнтного або

субградієнтного спуску для пошуку оптимальних рішень. Важливо, що ця модель добре масштабується, допускає паралельні обчислення та здатна враховувати як змінні подачі джерел, так і складні обмеження на пропускну здатність дуг.

Крім того, було запропоновано генератор тестових графів, який дає змогу моделювати різні типи топологій (Watts-Strogatz, Grid, Tree, Random Connected) з автоматичним призначенням ролей вузлам, орієнтацією дуг та генерацією обмежень. Це створює реалістичне середовище для тестування алгоритмів оптимізації.

В даному розділі був описаний теоретичний апарат для створення математичної моделі та розв'язку проблеми оптимізації потоків на різних графах за допомогою мови програмування Python.

РОЗДІЛ 3 ПРАКТИЧНА РЕАЛІЗАЦІЯ АЛГОРИТМУ

3.1 Інструменти, які використовувались для розробки програмного застосунку

У процесі реалізації математичної моделі оптимального розподілу потоків у мережах промислового типу було використано мову програмування Python, а також низку спеціалізованих бібліотек для роботи з графовими структурами, обчисленнями та візуалізацією. Такий вибір обумовлений як науковими, так і практичними міркуваннями.

Python є високорівневою мовою програмування з відкритим кодом, що набула широкого поширення у сфері наукових обчислень, моделювання та оптимізації. Її популярність серед дослідників пов'язана з рядом суттєвих переваг:

- 1) простий і читабельний синтаксис, що сприяє швидкій розробці та тестуванню математичних моделей;
- 2) велика кількість бібліотек із відкритим кодом, які охоплюють широкий спектр прикладних задач;
- 3) активна спільнота розробників та наявність документації;
- 4) можливість об'єктно-орієнтованої реалізації, що забезпечує структурованість та розширюваність коду.

Для побудови, аналізу та маніпулювання графовими структурами використано бібліотеку NetworkX. Вона дозволяє ефективно працювати з різними типами графів (орієнтованими, неорієнтованими, зваженими, динамічними тощо), що є важливим у контексті задач транспортного або ресурсного потоку. Переваги використання NetworkX:

- 1) можливість створення складних топологій: дерева, решітки, моделі Ваттса-Строгатца, випадкові зв'язні графи;

- 2) гнучкий механізм додавання атрибутів до вузлів і ребер, що дозволяє зберігати інформацію про статуси вузлів (джерело чи споживач), ваги, обмеження пропускної здатності, напрямки потоку;
- 3) реалізовані базові алгоритми аналізу графів: пошук шляхів, компоненти зв'язності, центральності тощо;
- 4) підтримка збереження графів у форматах GraphML, JSON, GEXF, що спрощує інтеграцію з іншими інструментами та системами.

Для реалізації математичних обчислень, пов'язаних із розв'язанням задач оптимізації, використано бібліотеку NumPy. Вона забезпечує ефективну роботу з векторами та багатовимірними масивами, що є критично важливим при обчисленнях у великих мережах. Серед її ключових переваг:

- 1) висока продуктивність завдяки реалізації ядра на мові C;
- 2) векторизовані операції, що дозволяють виконувати обчислення над усіма елементами масивів без використання циклів, тим самим значно підвищуючи швидкість виконання;
- 3) широкі можливості для реалізації математичних функцій, похідних, чисельного диференціювання та інтегрування, що використовуються в моделі для обчислення функцій вартості та їх градієнтів;
- 4) стабільна робота з великими обсягами даних і сумісність з іншими науковими бібліотеками.

Для візуалізації графів і результатів розрахунків було використано бібліотеку Matplotlib, яка дозволяє створювати високоякісні графічні зображення. У межах проєкту вона застосовується для виведення структури згенерованого графа, розмітки вузлів, напрямків та значень потоку, що наочно ілюструє поведінку алгоритму. Основні переваги цієї бібліотеки:

- 1) підтримка різноманітних типів графіків та візуалізацій;
- 2) можливість збереження графіків у форматах PNG, PDF, SVG, що дозволяє використовувати їх у звітності та публікаціях;

- 3) гнучке налаштування зовнішнього вигляду елементів: кольори, товщина ліній, шрифти тощо;
- 4) інтеграція з іншими бібліотеками Python та підтримка роботи в середовищах на кшталт Jupyter Notebook.

Таким чином, сукупне використання Python як базової мови, а також бібліотек NetworkX, NumPy та Matplotlib забезпечило повноцінну реалізацію програмного модуля для дослідження оптимального розподілу потоків у мережах. Вибрані інструменти не лише спростили процес розробки, але й гарантували масштабованість, продуктивність і можливість глибокого аналізу отриманих результатів. Завдяки цьому стало можливим ефективне тестування алгоритмів на різних типах графів, візуалізація отриманих рішень і подальше використання результатів у дослідницькій або практичній діяльності.

3.2 Генерація графів

Для ефективної реалізації задачі оптимального розподілу потоків у мережі надзвичайно важливою є коректна побудова графової структури, яка відображає як логіку руху ресурсу від джерел до споживачів, так і внутрішні обмеження міжвузлових з'єднань. У рамках цієї роботи генерація графів реалізована за допомогою об'єктно-орієнтованої архітектури, що включає базовий клас та набір спеціалізованих нащадків для різних типів топологій.

Базовий клас BaseFlowGraph виконує ключову функцію — він уніфікує процес створення, розмітки та підготовки графа до подальшого використання в оптимізаційному алгоритмі. На першому етапі ініціалізації всі вузли отримують логічний статус: джерело (well) або споживач (consumer). Розподіл статусів реалізовано стохастично, з перевагою на користь споживачів, що імітує типовий сценарій промислової або інфраструктурної мережі, де кількість точок споживання ресурсів значно перевищує кількість джерел.

Для кожного вузла-споживача додатково задається обсяг необхідного ресурсу у вигляді випадкового значення в межах попередньо визначеного діапазону. Такий підхід дозволяє враховувати варіативність у попиті між різними частинами мережі, що, в свою чергу, впливає на оптимальний розподіл потоків.

Окрема увага приділяється параметрам ребер графа. Для кожної пари зв'язаних вузлів задаються мінімальна та максимальна пропускна здатність, а також ваговий коефіцієнт, який визначає вартість передачі одиниці ресурсу. Ці параметри є ключовими для подальшої оптимізації, оскільки вони входять до цільової функції та обмежень задачі.

Найбільш важливим етапом побудови є орієнтація графа, яка реалізується відповідно до ряду логічних правил. Зокрема, для джерел і споживачів зі ступенем один фіксується напрям потоку, що гарантує коректність трактування ролі вузлів. Також реалізований спеціальний механізм перевірки досяжності всіх споживачів принаймні з одного джерела. Це забезпечує цілісність і функціональність згенерованої мережі, що є обов'язковою умовою для вирішення задачі потокового розподілу.

На основі BaseFlowGraph реалізовано чотири спеціалізовані класи, кожен з яких створює певну топологію, адаптовану до специфіки задачі. В даному пункті розглядаються чотири типи графів.

Першим типом є клас WattsFlowGraph, який реалізує модель малого світу за допомогою генератора графів Ваттса-Строгатца. Ця модель поєднує властивості регулярності та випадковості, що дозволяє змодельовати мережі з високим рівнем кластеризації, але малим середнім діаметром. Такі мережі добре імітують інфраструктури, де є як локальні з'єднання, так і окремі довгі зв'язки — типові, наприклад, для систем передачі енергії або інформації (див. рис. 3.1).

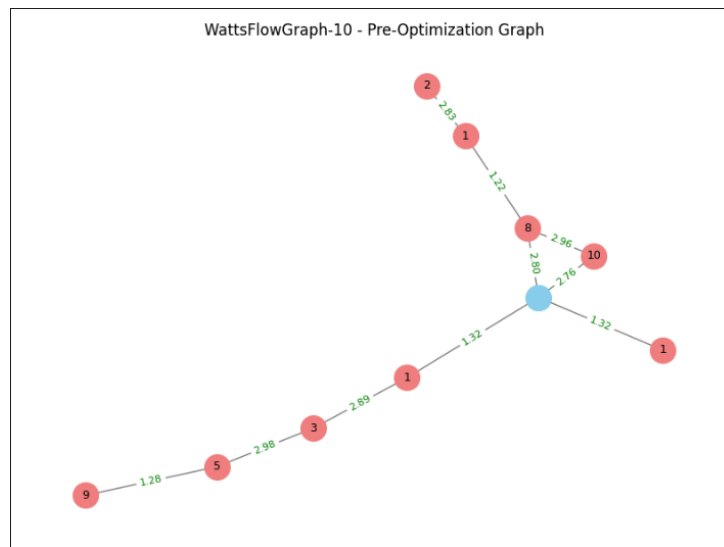


Рисунок 3.1 Граф Ваттса-Строгатца

Другим типом є клас `TreeFlowGraph` — створює структуру з кількох збалансованих дерев, які додатково з'єднуються випадковими ребрами для забезпечення зв'язності. Цей тип графів відповідає ієрархічним мережам, де ресурс передається від центрального вузла до кінцевих споживачів каскадним способом. Його структура наближена до класичних логістичних систем з чітко визначеною маршрутизацією (див. рис. 3.2).

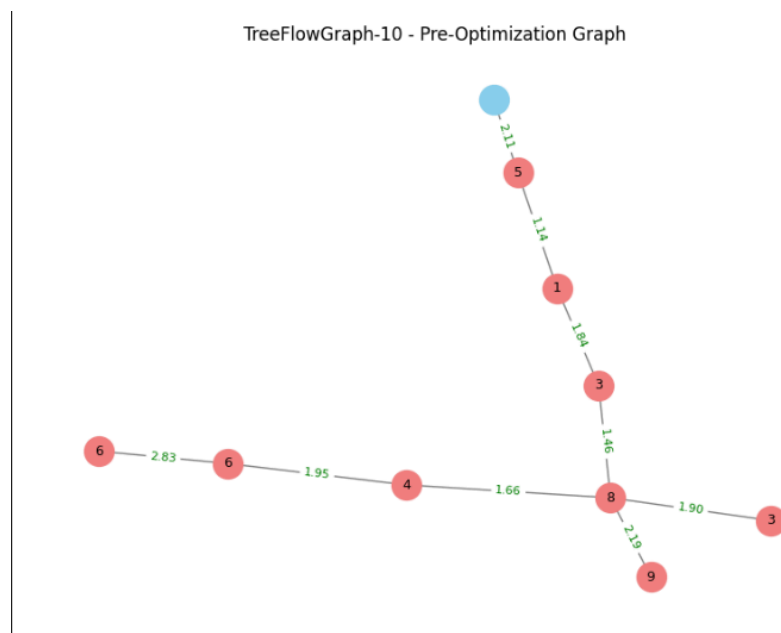


Рисунок 3.2 Дерево

Третій тип графів представлений класом GridFlowGraph, який генерує двовимірну решітку фіксованого розміру. Така структура характерна для урбаністичних або інженерних мереж — водопостачання, енергорозподілу, вуличного освітлення — де зв'язки між вузлами мають геометричну регулярність (див. рис. 3.3).

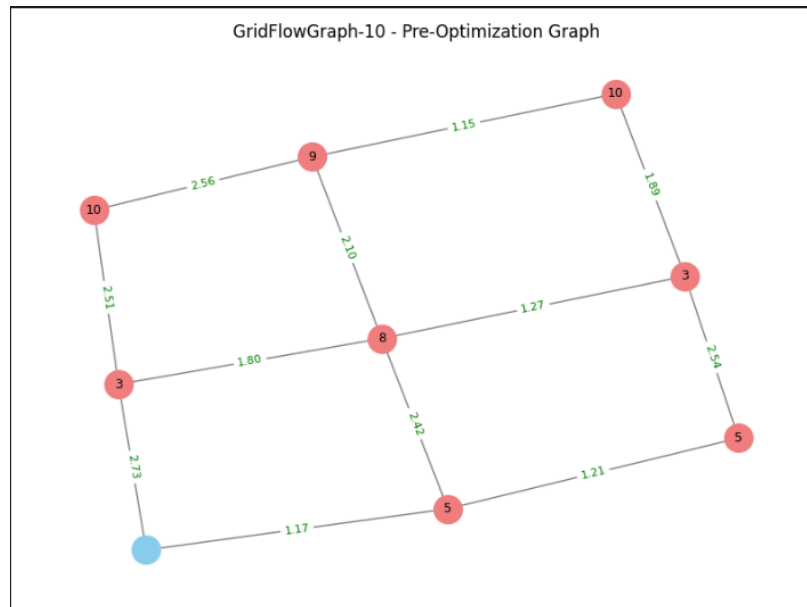


Рисунок 3.3 Сітчастий граф

Четвертий представлений тип — RandomFlowGraph — будується на основі випадкового дерева з подальшим додаванням додаткових ребер. В результаті створюється зв'язний граф із невизначеною кількістю циклів і непередбачуваною структурою. Такий підхід дозволяє тестувати роботу алгоритму на випадкових конфігураціях, які не мають явної симетрії або регулярності.

Кожен з типів графів після побудови автоматично проходить через процедуру орієнтації, маркування, обчислення параметрів ребер, а також зберігається у трьох форматах: PNG для візуального представлення, GraphML для подальшої інтеграції в зовнішні системи, та JSON для відновлення графа без повторної генерації.

Формат JSON у даному контексті є надзвичайно зручним для збереження повної структури графа, з усіма атрибутами вузлів і ребер. Це дозволяє завантажувати попередньо згенеровані графи без потреби повторної ініціалізації, що істотно прискорює процес експериментування та налагодження моделі.

Таким чином, розроблена система генерації графів дозволяє створювати мережі різного рівня складності та топології, що дає змогу гнучко адаптувати структуру до задачі оптимального розподілу потоків і масштабувати експерименти в залежності від обраного сценарію. Ретельно спроектована архітектура дозволяє також без суттєвих зусиль розширювати перелік підтримуваних типів графів шляхом додавання нових нащадків до базового класу.

3.3 Реалізація алгоритму оптимального розподілу потоків

Реалізацію алгоритму оптимального розподілу потоків здійснено у вигляді окремого класу `NonlinearFlowOptimizer`, який працює з уже підготовленим орієнтованим графом. Цей клас відповідає за чисельне розв'язання задачі мінімізації нелінійної функції вартості потоків з дотриманням обмежень на потоки в дугах та балансу вузлів. Основу реалізації становить узагальнення задачі з використанням методів множників Лагранжа.

Клас `NonlinearFlowOptimizer` приймає на вхід об'єкт класу `BaseFlowGraph` або будь-який з його дочірніх графів. Головна ідея полягає у послідовному оновленні потоків по ребрах на основі градієнтів функції Лагранжа та поступовому коригуванні значень множників Лагранжа. Такий підхід дозволяє ефективно враховувати обмеження на пропускні здатності та забезпечити виконання законів збереження потоку у вузлах.

У класі `NonlinearFlowOptimizer` було реалізовано такі етапи:

Ініціалізація структури — з графа витягуються всі ребра, вузли, статуси (джерело/споживач), значення споживання. Створюється словник `lambda_weight`, який зберігає значення множників Лагранжа λ_i для кожного вузла. Функція вартості кожного ребра вираховується за формулою (3.1):

$$F(x) = l \cdot \frac{|x|^{1+\alpha}}{1+\alpha} \quad (3.1)$$

де x — значення потоку, l — вага (вартість) дуги, α — параметр кривизни. Ця функція реалізована в методі `edge_cost`.

Похідна від цієї функції, яка використовується для обчислення оптимального потоку вираховується за формулою (3.2):

$$F'(x) = l \cdot |x|^\alpha \cdot \text{sign}(x) \quad (3.2)$$

Основний метод `optimize` виконує ітеративний процес знаходження оптимального розподілу потоків. Спочатку потоки ініціалізуються як нульові, після чого на кожній ітерації виконується:

1. Оновлення потоку на кожному ребрі відповідно до поточної різниці множників Лагранжа на кінцях ребра $p_j - p_i$, що впливає з рівності, яка вираховується за формулою (3.3):

$$x_{ij}(p) = \arg \min_{x \in [r_{ij}^-, r_{ij}^+]} \left(\int_0^x f_{ij}(t) dt - (p_i - p_j) \cdot x \right) \quad (3.3)$$

2. Якщо похідна функції вартості в межах обмежень менша за $p_j - p_i$, то вибирається нижнє обмеження r_{ij}^- як оптимальне значення потоку, або верхнє — якщо похідна перевищує межу.

3. Усі значення потоків автоматично обмежуються в межах вираховується за формулою (3.4):

$$\forall(i, j) : \quad x_{ij} \in [r_{ij}^-, r_{ij}^+] \quad (3.4)$$

4. Оновлення множників Лагранжа базується на обчисленні градієнтів балансу потоку для кожного вузла вираховується за формулою (3.5):

$$g_i^t = d_i - \sum_{j:(i,j) \in V} x_{ij}^t + \sum_{j:(j,i) \in V} x_{ji}^t \quad (3.5)$$

де d_i — значення споживання або виробництва у вузлі i . Градієнт відповідає за дисбаланс між вхідними та вихідними потоками. Обчислення виконується методом.

Множники оновлюються за градієнтним методом вираховується за формулою (3.6):

$$p_i^{t+1} = p_i^t + \alpha_t \cdot g_i^t \quad (3.6)$$

З параметром α_t , що за замовчуванням фіксований (наприклад, 0.1), але може бути адаптивним вираховується за формулою (3.7):

$$\alpha_t = \frac{\alpha_0}{\sqrt{t}} \quad (3.7)$$

Оновлення здійснюється лише для вузлів-споживачів, оскільки значення λ для джерел фіксовані.

У реалізації використовувались наступні бібліотеки:

1. NumPy: використовується для чисельного обчислення степеневих функцій, похідних, коренів, а також для роботи з абсолютними значеннями і знаками. Завдяки векторизованим обчисленням вона забезпечує високу швидкість обробки навіть при великій кількості ребер.

2. NetworkX: основна бібліотека для зберігання графів, обробки орієнтації дуг, пошуку шляхів, а також для зручного доступу до структур типу "суміжність", "успадковані атрибути", "статуси вузлів". NetworkX надає зручний інтерфейс для орієнтованих графів, що критично важливо для реалізації розподілу потоків із напрямком.

3. Math/Deque: допоміжні засоби для організації обходу графів, черг BFS, генерації випадкових значень тощо.

Кожен крок алгоритму строго відповідає математичній формулі, наведений у теоретичній частині дипломної роботи. Логіка реалізації чітко відокремлює обчислення потоків, градієнтів та оновлення множників, що дозволяє легко масштабувати алгоритм або адаптувати його для інших мереж.

3.5 Збереження та візуалізація результатів

Завершальним етапом реалізації програмної частини стало забезпечення збереження та візуального подання як вихідної структури мережі, так і результатів обчислень після оптимізації. Візуалізація дозволяє ефективно інтерпретувати модель, перевірити її коректність та здійснити якісний аналіз розподілу потоків.

Для збереження згенерованого або імпортованого графа було реалізовано підтримку кількох форматів:

1. PNG (Portable Network Graphics) — статичне зображення графа з кольоровим позначенням типу вузлів (колодязі — зелений, споживачі — блакитний), підписами на вершинах і ребрах. Вивід реалізовано через бібліотеку `matplotlib` методом `plt.savefig()`.

2. GraphML — структурований XML-формат для збереження графових об'єктів із повним набором атрибутів. Дає можливість відкривати результати в зовнішніх візуалізаційних інструментах (наприклад, Gephi або yEd).

3. JSON (node-link формат) — структура сумісна з NetworkX, яка дозволяє повторно завантажити граф у Python-програмі, змінити його або використати для аналізу.

Функціональність реалізована у методі `visualize()` базового класу `BaseFlowGraph`, який одночасно забезпечує і побудову візуалізації, і збереження в усіх зазначених форматах.

Після розв'язання задачі оптимального розподілу потоків (метод `optimize()` класу `NonlinearFlowOptimizer`) користувач отримує словник зі значеннями оптимальних потоків для кожної дуги. Для аналізу цих результатів реалізовано спеціалізований метод `visualize_flows()` у класі оптимізатора.

Візуалізація потоку після оптимізації відбувається за наступною логікою:

- 1) кольорове кодування ребер;
- 2) товщина дуги пропорційна модулю значення потоку — що вища величина потоку, то товстіша лінія ребра;
- 3) цифрове маркування: біля кожного ребра виводиться числове значення потоку з точністю до одного знака після коми.

Ця візуалізація реалізується за допомогою функцій NetworkX (`nx.draw`, `nx.draw_networkx_edge_labels`) та `matplotlib.pyplot`. Вона дозволяє не лише оцінити, як розподілився потік у мережі, а й легко виявити вузькі місця, перенавантажені дуги або недовантажені частини мережі. Вузли залишаються стилізованими згідно зі своїм типом — колодязі залишаються зеленими, а споживачі — синіми.

Окрім візуального аналізу, користувач може вивантажити результати у зручний формат, наприклад, у JSON або GraphML, щоб використовувати ці дані для подальшої інтеграції у зовнішні системи аналітики або симуляцій.

Такий підхід до візуалізації відіграє важливу роль у моделюванні потоків, оскільки дозволяє:

- 1) переконатися у коректності орієнтації графа та напрямку руху потоків;
- 2) виявити вузли, що не обслуговуються або отримують надмірний/недостатній потік;
- 3) інтерпретувати ефективність моделі у розподілі ресурсів між усіма точками споживання.

Таким чином, реалізована система візуалізації є не лише допоміжним засобом, а й інструментом аналітики, який сприяє повноцінному використанню результатів обчислень.

3.6 Тестування результатів роботи програми

Для тестування алгоритму було запущено експеримент, де для кожного з типів графів було запущено алгоритм на 10, 20, 50, 100, 250, 500 вершинах. Експеримент проводився на графах різної топології, згенерованих випадковим чином. Відношення колодязів до споживачів було 1 до 6. В межах експерименту заміряється:

- 1) кількість ітерацій;
- 2) час виконання алгоритму;
- 3) сумарна потреба в споживанні;
- 4) кількість речовини передана на колодці;
- 5) сумарна витрата речовини на ребрах графа.

Результати експерименту задовольнили усі вимоги до алгоритму (див. табл. 3.1):

- 1) кожен споживач отримав необхідну для себе кількість речовини;
- 2) алгоритм відпрацьовує за адекватний час;
- 3) зроблена карта переходів для оптимального розподілу потоків.

З таблиці видно, що усі споживачі задоволені, для кожного графу порахована загальна вартість переданої речовини, втрати на ребрах, кількість ітерацій і час.

Нижче наведені результати оптимізації графів окремо для кожного типу графу з 10 вузлами, при більшій кількості вузлів візуалізації незручно демонструвати результати у вигляді малюнків. Результат роботи це граф до оптимізації, граф після, таблиця з балансом вузлів. На рисунку 3.4 можемо

побачити графи до оптимізації, з розміткою споживач-колодязь. На рисунку 3.5 бачимо вже оптимізований граф.

Таблиця 3.1 Результати експерименту

graph	nodes	iterations	time	cost	demand	actual	losses
TreeFlowGraph	10	473	4,79	525,98	14	14	511,983
TreeFlowGraph	20	325	13,74	537,72	37	37	500,718
TreeFlowGraph	100	1456	354,92	8949,08	179	179	8770,07
TreeFlowGraph	500	1990	888,58	29740,9	817	817	28923,9
GridFlowGraph	10	153	2,07	151,17	17	17	134,174
GridFlowGraph	20	187	4,89	387,4	30	30	357,404
GridFlowGraph	100	147	19,28	1093,31	177	177	916,313
GridFlowGraph	500	215	121,34	5312,84	820	820	4492,83
WattsFlowGraph	10	243	1,65	433,3	20	20	413,295
WattsFlowGraph	20	335	4,56	407,2	31	31	376,199
WattsFlowGraph	100	911	58,94	5356,09	167	167	5189,08

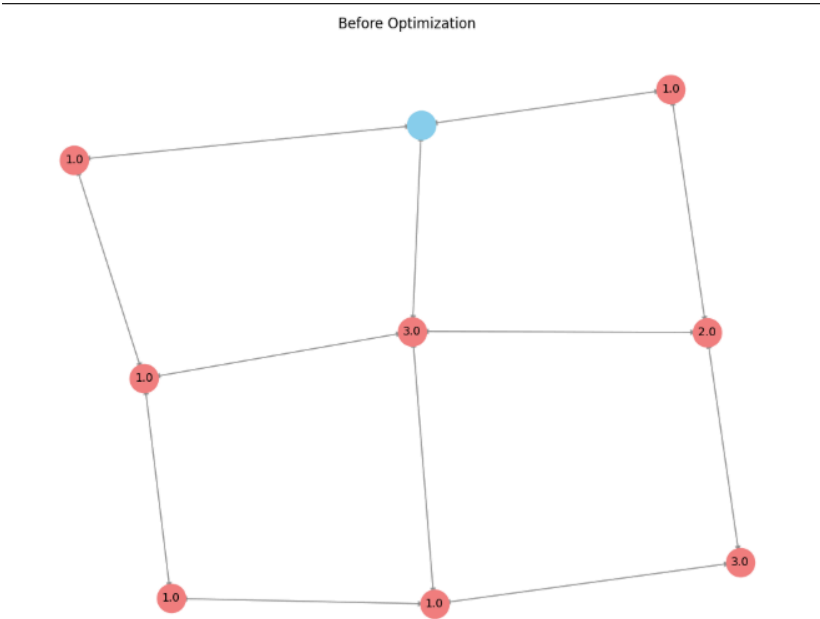


Рисунок 3.4 Решітчастий граф з 9 вузлами до оптимізації

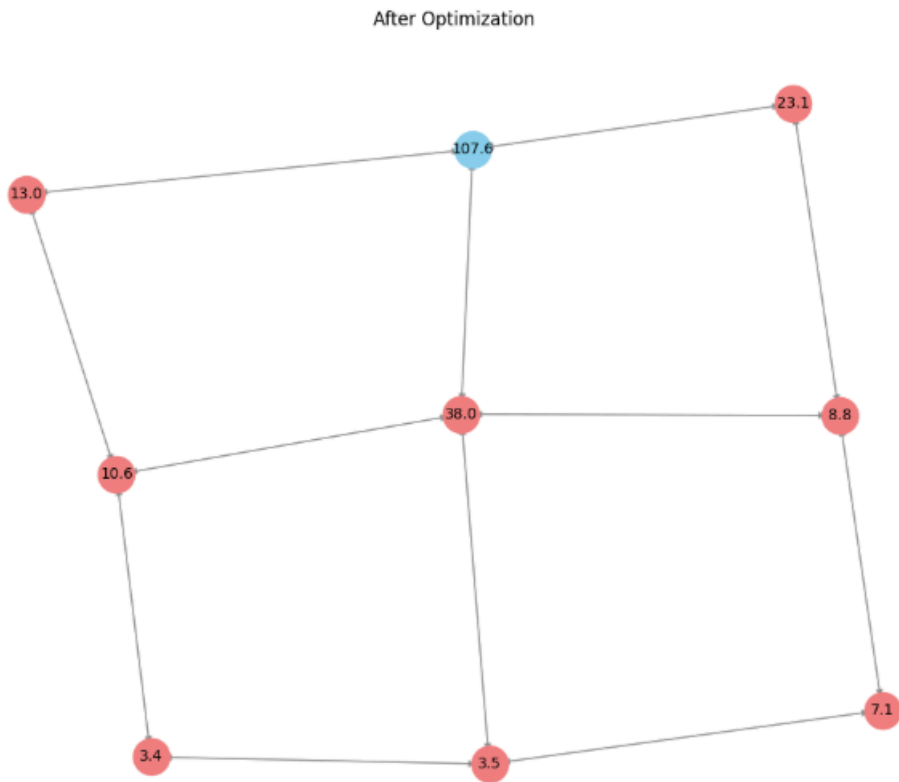


Рисунок 3.5 Граф після оптимізації

На наступних таблицях видно баланс постачання, потреб і витрат для всіх вузлів графа (див. табл. 3.2) і розподіл потоків на оптимізованому графі (див. табл. 3.3)

Таблиця 3.2 Таблиця контролю балансу в вузлах

n	type	in	out	net	losses	need	supply
0	consumer	3,86	2,86	1	22,11	1	23,11
1	well	0	13	-13	0	0	107,55
2	consumer	4,21	3,21	1	12,01	1	13,01
3	consumer	3,61	1,61	2	6,81	2	8,81
4	consumer	5,67	2,67	3	34,99	3	37,99
5	consumer	3,21	2,21	1	9,59	1	10,59
6	consumer	3	0	3	4,14	3	7,14
7	consumer	2,39	1,39	1	2,54	1	3,54
8	consumer	1,47	0,47	1	2,37	1	3,37

Таблиця 3.3 Розподіл потоків при оптимізації графу

i\j	0	1	2	3	4	5	6	7	8
0				8,97					
1	25,97		16,22		39,1				
2						12,8			
3							4		
4				1,45				4,16	
5					1,55				3,84
6									
7							3,13		
8								0,77	

Висновки до розділу 3

У межах третього розділу було реалізовано повноцінний програмний модуль для розв’язання задачі оптимального розподілу потоків у мережах з довільною топологією. У процесі розробки було сформовано комплексний підхід, який охоплює всі ключові етапи — від генерації графової моделі до оптимізаційних обчислень і візуалізації результатів. Вибір інструментів і архітектура реалізації були обґрунтовані як з позиції математичної коректності, так і з точки зору ефективності та зручності подальшого використання. Зокрема, як базову мову програмування було обрано Python — потужний інструмент для чисельного моделювання, який, у поєднанні з бібліотеками NetworkX, NumPy та Matplotlib, дозволив забезпечити високу швидкість розробки, гнучкість коду та ефективне управління графовими структурами.

На етапі генерації графів було реалізовано універсальну архітектуру з базовим класом BaseFlowGraph і чотирма спеціалізованими нащадками, які відповідають різним типам топологій: Ваттса-Строгатца, дерево, решітка та випадковий граф. Це дозволило моделювати широкий спектр можливих конфігурацій мереж: від ієрархічних структур до нерегульованих, з

неоднорідними характеристиками. Такий підхід значно підвищує універсальність розробленого інструменту та дає змогу адаптувати модель до реальних прикладних задач. Орієнтація графа та маркування вузлів (колодязі й споживачі) здійснювалась автоматично з використанням логічних правил, що гарантувало цілісність структури та її придатність до подальшої оптимізації. Було впроваджено механізм перевірки досяжності споживачів із джерел, а також забезпечено контроль за параметрами дуг, зокрема, їх пропускною здатністю та вартістю передачі ресурсу.

Оптимізаційна частина реалізована через клас `NonlinearFlowOptimizer`, що базується на методі множників Лагранжа. Ключовими перевагами даної реалізації є чіткий поділ обчислювальних процедур (розрахунків потоків, оновлення множників, градієнтні кроки) та їх точна відповідність теоретичним положенням, викладеним у попередніх розділах. Алгоритм довів свою ефективність під час численних ітерацій, демонструючи здатність до забезпечення балансу потоків у вузлах та дотримання обмежень на дугах.

Підсумковим етапом стала побудова та збереження візуалізацій. Завдяки продуманій кольоровій схемі та маркуванню значень потоків реалізована система наочно відображає результати оптимізації, дозволяючи легко виявити проблемні місця у структурі, перевірити коректність напрямків потоків і провести аналіз розподілу ресурсу. Крім того, підтримка різних форматів збереження (PNG, JSON, GraphML) забезпечила інтеграційні можливості з іншими інструментами.

Результати тестування, проведені на графах з кількістю вершин від 10 до 500, підтвердили працездатність та масштабованість алгоритму. Для всіх експериментальних сценаріїв було досягнуто коректного розподілу потоків, з дотриманням вимог споживання, реалістичним часом виконання та прийнятними обчислювальними витратами. Алгоритм показав стабільність навіть при збільшенні розмірів мережі, що дозволяє його застосовувати у задачах реального масштабу.

Таким чином, у цьому розділі було здійснено не лише імплементацію математичної моделі, але й створено гнучкий, масштабований та інтегровний інструмент для аналізу задач оптимального розподілу потоків. Отримані результати свідчать про високу ефективність обраного підходу, а розроблена програмна реалізація може бути використана як у дослідницьких, так і в прикладних проєктах у сфері логістики, енергетики, телекомунікацій та інших суміжних галузях.

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ РЕАЛІЗАЦІЇ ТА ПОСТІЙНОЇ РОБОТИ АЛГОРИТМУ ОПТИМІЗАЦІЇ ПОТОКІВ

В цьому розділі буде проведено оцінювання основних характеристик для майбутнього вбудованого програмного продукту.

Також в даному дослідженні показано різні варіанти реалізації для забезпечення найбільш коректної та оптимальної стратегії вибору, що має вплив на економічні фактори та сумісність з майбутнім програмним продуктом. Для цього застосовувався апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) передбачає собою технологію, що дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. ФВА проводиться з метою виявлення резервів зниження витрат за рахунок ефективніших варіантів виробництва, кращого співвідношення між споживчою вартістю виробу та витратами на його виготовлення. Для проведення аналізу використовується економічна, технічна та конструкторська інформація.

Алгоритм функціонально-вартісного аналізу включає в себе визначення послідовності етапів розробки продукту, визначення повних витрат (річних) та кількості робочих часів, визначення джерел витрат та кінцевий розрахунок вартості програмного продукту.

4.1 Постановка задачі проектування

У роботі застосовується метод функціонально-вартісного аналізу (ФВА) для проведення техніко-економічного обґрунтування розробки програмної системи оптимального розподілу потоків у мережі. Оскільки рішення щодо структури графа, вибору методу оптимізації та способу реалізації напряду впливають на загальну ефективність системи, кожен компонент повинен відповідати заданим функціональним і технічним вимогам. З цієї причини

аналіз охоплює детальне вивчення можливостей програмного продукту, який забезпечує коректне балансування потоків між джерелами і споживачами при дотриманні обмежень на пропускну здатність ребер та інших умов мережі.

Технічні вимоги до програмного продукту є наступні:

- a) проведення оптимізації потоків за прийнятний час;
- b) зручність та зрозумілість для користувача;
- c) зручна візуалізація оптимізацій;
- d) підтримка одночасної обробки кількох графів;
- e) можливість зручного масштабування та обслуговування, додавання нового функціоналу;
- f) мінімальні витрати на впровадження програмного продукту.

4.2 Обґрунтування функцій програмного продукту

Головна функція F_0 відповідає за створення програмного забезпечення для побудови, моделювання та оптимізації мереж з розподілом потоків. Вона включає в себе реалізацію інтерфейсу для користувача, ядра обчислень для розв'язання задачі оптимізації, а також візуалізаційних засобів для представлення графів і результатів розрахунків.

F_1 – вибір найбільш підходящої мови програмування на якій базуються бібліотеки для чисельних обрахунків;

F_2 – вибір найбільш підходящої мови програмування чи програмного продукту для написання обгортки для оптимізації та візуалізації;

F_3 – вибір середовища розробки.

Функція F_1 :

- a) Rust;
- b) C++.

Функція F_2 :

- a) Python;
- b) R;
- c) використання готової моделі.

Функція F_3 :

- a) VS code;
- b) JetBrains IDE.

Варіанти реалізації основних функцій представлено у морфологічній карті системи (рис. 4.1).

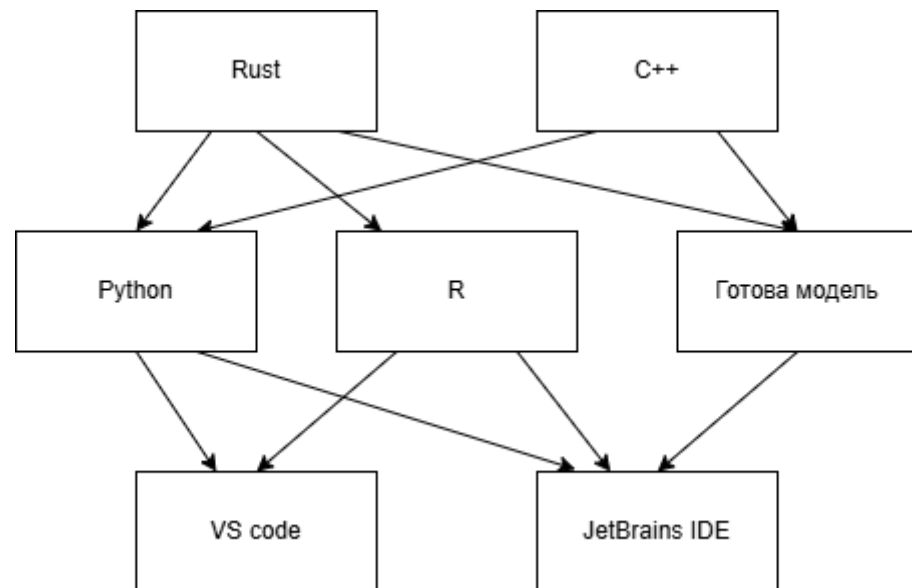


Рисунок 4.1 – Морфологічна карта

Морфологічна карта відображає множину всіх можливих варіантів основних функцій. Позитивно-негативна матриця показана в табл. 4.1.

Таблиця 4.1 – Позитивно-негативна матриця

Функція	Варіант реалізації	Переваги	Недоліки
F₁	(A) Rust	Висока продуктивність. Безпечне управління пам'яттю.	Крута крива навчання. Менша кількість готових бібліотек обрахунків.
	(Б) C++	Висока швидкість виконання. Широкий вибір бібліотек для низькорівневого доступу.	Застарілість бібліотек.
F₂	(A) Python	Величезна кількість бібліотек для машинного навчання. Добра документація. Легка інтеграція з моделями.	Нижча швидкодія у порівнянні з іншими мовами. Може потребувати окремого середовища виконання.
	(Б) R	Сильні сторони в статистичному аналізі. Зручна візуалізація даних.	Обмежена підтримка для NLP. Менша кількість сучасних бібліотек для глибокого навчання.
	(B) Готова модель (API або сервіс)	Швидка інтеграція. Мінімальні витрати на тренування. Висока точність (у випадку використання перевірених рішень).	Обмежена кастомізація. Залежність від зовнішнього сервісу або ліцензії.
F₃	(A) Visual Studio Code	Легкий, кросплатформенний редактор. Велика кількість розширень для Rust, Python, C++.	Менше вбудованих інструментів порівняно з повноцінними IDE.
	(Б) JetBrains IDE (CLion, PyCharm)	Розширені можливості для рефакторингу, автодоповнення, налагодження. Інтеграція з Git, Docker, тощо.	Високе споживання ресурсів. Частина функцій доступна тільки у платній версії.

Згідно з аналізом позитивно-негативної матриці, деякі варіанти реалізації функцій програмного продукту не відповідають поставленим перед ним задачам. Ці варіанти слід виключити з розробки.

Функція F_1 .

Для реалізації системи оптимізації потоків було обрано мову програмування Python у поєднанні з бібліотеками NetworkX та NumPy, які базуються на високопродуктивному коді, написаному мовою C++. Такий підхід дозволяє поєднати зручність розробки на Python з ефективністю обчислень, необхідною для роботи з великими графами та проведення чисельної оптимізації.

Функція F_2 .

У процесі вибору засобу реалізації модуля штучного інтелекту для аналізу повідомлень було проаналізовано варіанти Python, R та використання готової моделі. Найбільш доцільним є використання мови Python, оскільки вона має найширшу підтримку бібліотек для обробки даних і візуалізацій. Крім того, вона є стандартом де-факто в галузі NLP, що забезпечує легку інтеграцію сучасних моделей.

Функція F_3 .

При виборі середовища розробки розглядалися Visual Studio Code та середовище JetBrains. Обидва інструменти є функціональними та підтримують усі необхідні мови. Проте пріоритетним є використання Visual Studio Code через його кросплатформенність, швидкодію та широкий набір розширень. Це дозволяє пришвидшити розробку та забезпечити гнучкість при роботі з різними компонентами системи.

Маємо наступні варіанти реалізації ПП:

$$F_1 a - F_2 a - F_3 a;$$

$$F_1 b - F_2 a - F_3 a.$$

Оцінювання якості згаданих функцій виконується з допомогою параметрів нижче.

4.3 Обґрунтування системи параметрів програмного продукту

Враховуючи доступні технології та інструменти, було проаналізовано ключові фактори для розрахунку коефіцієнта технічного рівня, які впливають на вибір рішення для реалізації системи оптимального розподілу потоків у мережі.

Оцінюючи можливість використання мов програмування та бібліотек, зокрема Python у поєднанні з високопродуктивними модулями на базі C++ (такими як NetworkX і NumPy), було визначено такі критерії для порівняння кожного з варіантів реалізації:

- 1) $X1$ – гіпотетичний об'єм коду програми;
- 2) $X2$ – час на навчання моделі;
- 3) $X3$ – час на виконання поставленого завдання;
- 4) $X4$ – точність моделі.

Параметри поділяються на гірші, середні і кращі та обираються замовником та диктуються умовами використання програмного продукту. (табл. 4.2).

Таблиця 4.2 – Основні параметри програмного продукту

Назва Параметра	Умовні позначенн я	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Гіпотетичний об'єм коду програми	$X1$	кількість рядків коду	6000	4500	3500
Час на навчання моделі	$X2$	хвилини	65	31	10
Час витрачений на виконання поставленого завдання	$X3$	дні	80	40	20
Точність моделі	$X4$	відсотки	80	90	95

Дані в таблиці 4.2 дозволяють побудувати графічні характеристики для оцінки та порівняння параметрів (рис. 4.2 – рис. 4.5).

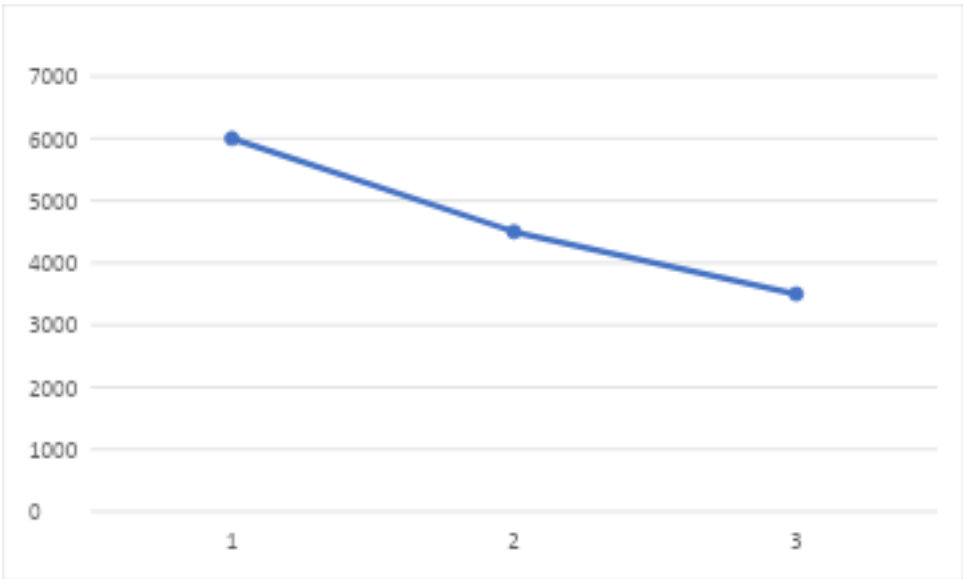


Рисунок 4.2 – X1, Гіпотетичний об’єм коду програми

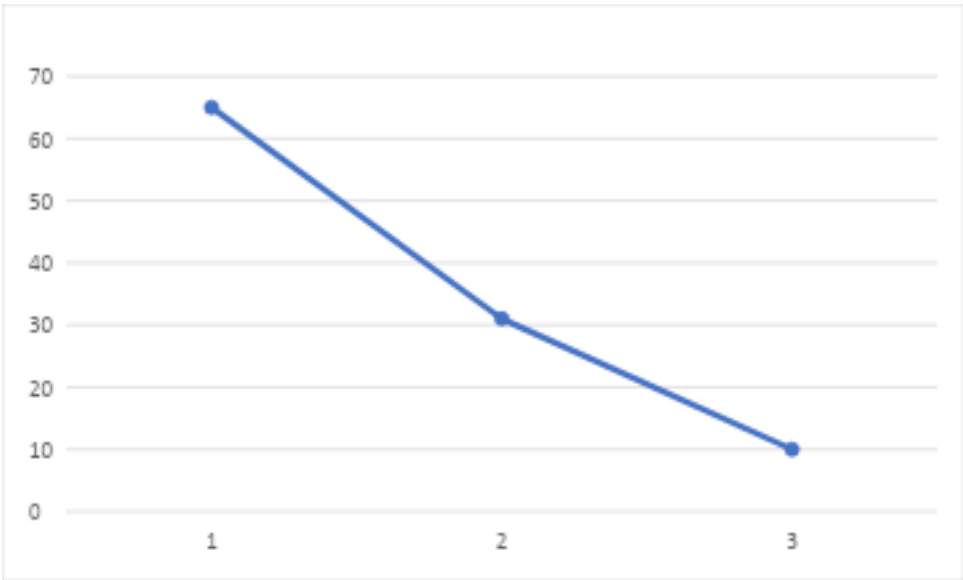


Рисунок 4.3 – X2, Час на навчання моделі

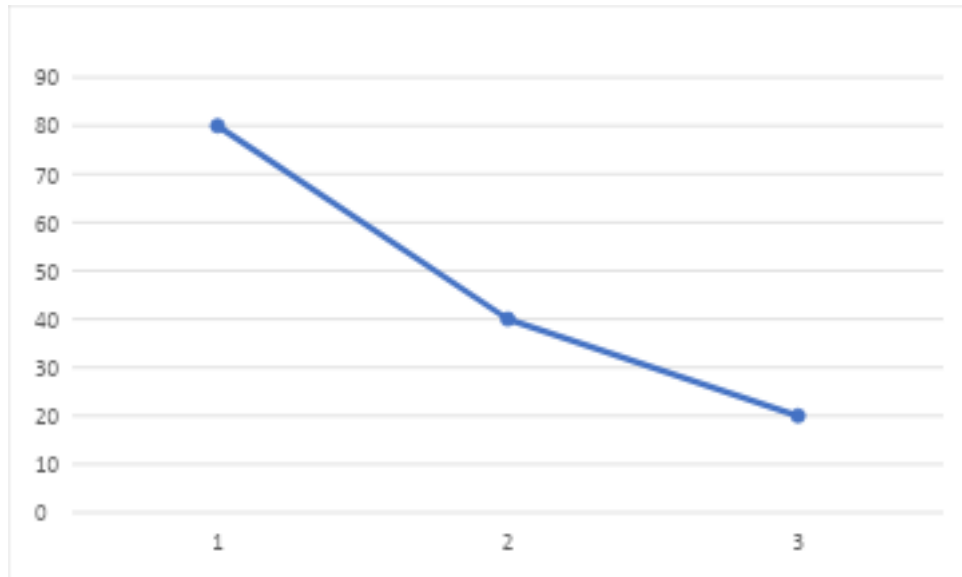


Рисунок 4.4 – ХЗ, Час витрачений на виконання поставленого завдання

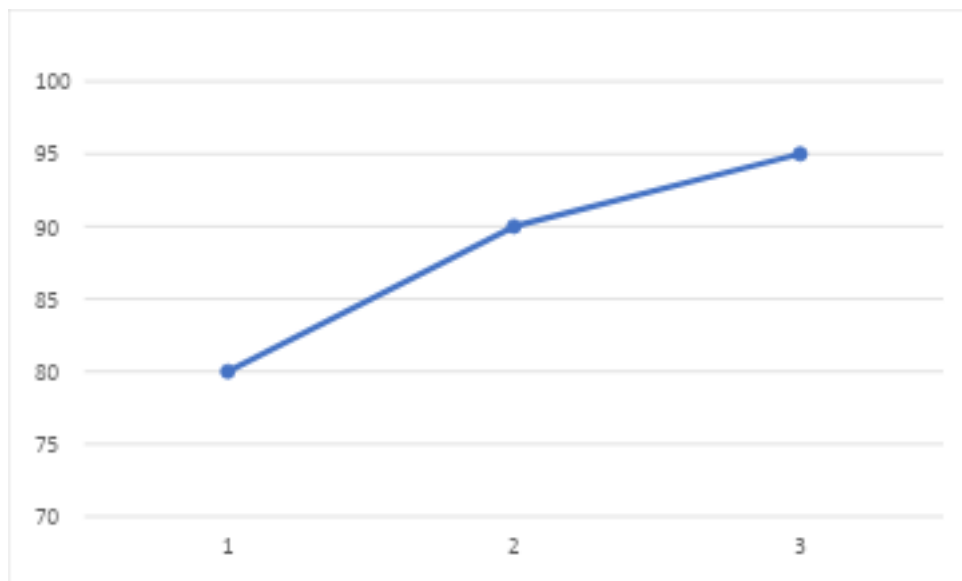


Рисунок 4.5 – Х4, Точність моделі

4.4 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангі в R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
$X1$	Гіпотетичний об'єм коду програми	кількість рядків коду	1	2	2	1	1	1	2	10	-7,5	56,25
$X2$	Час на навчання моделі	хвилини	3	4	3	3	4	3	4	24	6,5	42,25
$X3$	Час витрачений на виконання поставленого завдання	дні	2	1	1	2	2	2	1	11	-6,5	42,25
$X4$	Точність моделі	відсотки	4	3	4	4	3	4	3	25	7,5	56,25
	Разом		10	10	10	10	10	10	10	70	0	197

Щоб оцінити достовірність експертних рейтингів, визначаємо такі параметри:

а) сумарний ранг кожного показника та загальний ранг усіх показників:

$$R_i = \sum_{j=1}^N r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 70$$

де N – кількість експертів;

n – число параметрів.

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 17,5$$

с) різниця між рангами кожного параметра та середньої суми рангів:

$$\Delta_i = R_i - T$$

Сума, що визначає відхилення кожного параметру повинна дорівнювати 0;

д) загальна сума квадратів відхилення:

$$S = \sum_{i=1}^N \Delta_i^2 = 197.$$

Коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3-n)} = \frac{12 \cdot 197}{7^2(4^3-4)} = 0,804 > W_k = 0,56.$$

Отримані результати ранжування можна вважати надійними, адже розрахований коефіцієнт узгодженості перевищує нормативний показник, значення якого дорівнює 0,56. Таким чином утворимо таблицю 4.4, де попарно порівнюємо всі параметри.

Таблиця 4.4 – Попарне порівняння параметрів

Параметр и	Експерти							Кінцев а оцінка	Числове значенн я
	1	2	3	4	5	6	7		
X1 і X2	<	<	<	<	<	<	<	<	0,5
X1 і X3	<	>	>	<	<	<	>	<	0,5
X1 і X4	<	<	<	<	<	<	<	<	0,5
X2 і X3	>	>	>	>	>	>	>	>	1,5
X2 і X4	<	>	<	<	>	<	<	<	0,5
X3 і X4	<	<	<	<	<	<	<	<	0,5

Ступінь переваги i -го параметра над j -тим визначається числовим параметром a_{ij} і визначається по формулі:

$$a_{ij} = \begin{cases} 1.5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases}$$

З отриманих числових оцінок переваги складемо матрицю $A = \| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості K_{ei} за наступними формулами:

$$K_{bi} = \frac{b_i}{\sum_{i=1}^n b_i}$$

$$b_i = \sum_{j=1}^N a_{ij}$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{Bi} = \frac{b_i}{\sum_{i=1}^n b_i}.$$

$$b_i = \sum_{j=1}^N a_{ij} b_j.$$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 4.5 – Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j				Перша ітер.		Друга ітер.		Третя ітер.	
	X1	X2	X3	X4	b_i	K_{Bi}	b_i^1	K_{Bi}^1	b_i^2	K_{Bi}^2
X1	1	0,5	0,5	0,5	2,5	0,16	9,25	0,16	34,125	0,16
X2	1,5	1	1,5	0,5	4,5	0,28	16,25	0,28	59,125	0,28
X3	1,5	0,5	1	0,5	3,5	0,22	12,25	0,21	41,875	0,2
X4	1,5	1,5	1,5	1	5,5	0,34	21,25	0,35	77,875	0,36
Всього:					16	1	59	1	213	1

4.5 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів $X1$ (Гіпотетичний об'єм коду програми), $X2$ (Час на навчання моделей) та $X4$ (Точність моделі) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра $X3$ (Час витрачений на виконання поставленого завдання) обрано не найгіршим.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (табл. 4.6):

$$K_K(j) = \sum_{i=1}^n K_{bi,j} B_{i,j}$$

де n – кількість параметрів;

K_{bi} – коефіцієнт вагомості i -го параметра;

B_i – оцінка i -го параметра в балах.

Таблиця 4.6 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіанти реалізації функцій	Параметри	Абсолютні значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1	А	X1	4000	29	0,16	4,64
	Б	X1	4700	24	0,16	3,84
F2	А	X3	28	19	0,2	3,8
F3	А	X4	97	23	0,36	8,28

За даними з табл. 4.6 та за формулою:

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \dots + K_{\text{ТУ}}[F_{zk}]$$

визначаємо рівень якості кожного з варіантів:

$$K_{K1} = 4,64 + 3,8 + 8,28 = 16,72.$$

$$K_{K2} = 3,84 + 3,8 + 8,28 = 15,92.$$

Як видно з розрахунків, кращим є 1 варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.6 Економічний аналіз варіантів розробки ПП

Всі варіанти включають в себе два окремих завдання.

1. Розробка проекту програмного продукту.
2. Розробка допоміжної програми для візуалізації.

Завдання 1 за ступенем новизни відноситься до групи Б, завдання 2 до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 2.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Загальна трудомісткість обчислюється як:

$$T_o = T_p \cdot K_{\Pi} \cdot K_{СК} \cdot K_M \cdot K_{СТ} \cdot K_{СТ.М}$$

де T_p – трудомісткість розробки ПП;

K_{Π} – поправочний коефіцієнт;

$K_{СК}$ – коефіцієнт на складність вхідної інформації;

K_M – коефіцієнт рівня мови програмування;

$K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;

$K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення.

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $T_p = 64$ людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $K_{\Pi} = 1.021$. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх завдань рівний 1: $K_{СК} = 1$. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $K_{СТ} = 0.6$. Тоді загальна трудомісткість програмування першого завдання дорівнює:

$$T_1 = 64 \cdot 1.021 \cdot 1 \cdot 0.6 = 65.94 \text{ людино-днів.}$$

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $T_p = 27$ людино-днів, $K_{\Pi} = 1.08$, $K_{СК} = 1$, $K_{СТ} = 0.7$:

$$T_2 = 27 \cdot 1.08 \cdot 1 \cdot 0.7 = 20.41 \text{ людино-днів.}$$

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$$T_0 = (65.94 + 20.41) \cdot 8 = 690.8 \text{ людино-годин.}$$

В розробці беруть участь один програміст з окладом 15000 грн.
Визначимо середню зарплату за годину за формулою:

$$CЧ = \frac{M}{T_m \cdot t} \text{ грн}$$

де M – місячний оклад працівників;

T_m – кількість робочих днів на місяць;

t – кількість робочих годин в день.

$$CЧ = \frac{15000}{18 \cdot 8} = 104,17 \text{ грн}$$

Тоді, розрахуємо заробітну плату за формулою:

$$CЗП = C_{\text{ч}} \cdot T_i \cdot КД$$

де $C_{\text{ч}}$ – величина погодинної оплати праці програміста;

T_i – трудомісткість відповідного завдання;

$КД$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

$$C_{ЗП} = 104,17 \cdot 690.8 \cdot 1.125 = 80955,71 \text{ грн.}$$

Відрахування на єдиний соціальний внесок становить 22%:

$$C_{\text{ВІД}} = C_{ЗП} \cdot 0.22 = 80955,71 \cdot 0.22 = 17810,26 \text{ грн.}$$

Тепер визначимо витрати на оплату однієї машино-години. (C_M)

Так як одна ЕОМ обслуговує одного програміста з окладом 15000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$C_M = 12 \cdot M \cdot K_3 = 12 \cdot 15000 \cdot 0,2 = 36000 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$C_{3П} = C_M \cdot (1 + K_3) = 36000 \cdot (1 + 0.2) = 43200 \text{ грн.}$$

Відрахування на соціальний внесок:

$$C_{ВІД} = C_{3П} \cdot 0.22 = 43200 \cdot 0.22 = 9504 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 49000 грн.

$$C_A = K_{TM} \cdot K_A \cdot Ц_{ПР} = 1.23 \cdot 0.25 \cdot 49000 = 15067,5 \text{ грн,}$$

де K_{TM} – коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

K_A – річна норма амортизації;

$Ц_{ПР}$ – договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$C_P = K_{TM} \cdot Ц_{ПР} \cdot K_P = 1.23 \cdot 49000 \cdot 0.03 = 1808,1 \text{ грн,}$$

де K_P – відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$T_{\text{ЕФ}} = (D_{\text{К}} - D_{\text{В}} - D_{\text{С}} - D_{\text{Р}}) \cdot t_3 \cdot K_{\text{В}} = (365 - 105 - 21 - 16) \cdot 8 \cdot 0.87 = \\ = 1552,08 \text{ години,}$$

де $D_{\text{К}}$ – календарна кількість днів у році;

$D_{\text{В}}, D_{\text{С}}$ – відповідно кількість вихідних та святкових днів;

$D_{\text{Р}}$ – кількість днів планових ремонтів устаткування;

t – кількість робочих годин в день;

$K_{\text{В}}$ – коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$C_{\text{ЕЛ}} = T_{\text{ЕФ}} \cdot N_{\text{С}} \cdot K_3 \cdot C_{\text{ЕН}} = 1552,08 \cdot 0,3 \cdot 0,5 \cdot 9,43 = 2195,42 \text{ грн,}$$

де $N_{\text{С}}$ – середньо-споживча потужність приладу;

K_3 – коефіцієнтом зайнятості приладу;

$C_{\text{ЕН}}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$C_{\text{Н}} = C_{\text{ПР}} \cdot 0.67 = 20928,45 \cdot 0,67 = 14022, 06 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$C_{\text{ЕКС}} = C_{\text{ЗП}} + C_{\text{ВІД}} + C_{\text{А}} + C_{\text{Р}} + C_{\text{ЕЛ}} + C_{\text{Н}}. \#(4.17)$$

$$C_{\text{ЕКС}} = 80955,71 + 22809,6 + 15067,5 + 1808,1 + 2195,42 + 14022, 06 = \\ = 227623,14 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$C_{M-Г} = C_{EКС} / T_{ЕФ} = 227623,14 / 1552,08 = 146,65 \text{ грн/год.}$$

Оскільки в даному випадку всі роботи, які пов'язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$C_M = C_{M-Г} \cdot T$$

$$C_M = 146,65 \cdot 690,8 = 101188,5 \text{ грн.}$$

Накладні витрати складають 67% від заробітної плати:

$$C_H = C_{ЗП} \cdot 0,67$$

$$I. \quad C_H = 172698,27 \cdot 0,67 = 115707,84 \text{ грн.}$$

Отже, вартість розробки ПП за варіантами становить:

$$C_{ПП} = C_{ЗП} + C_{ВІД} + C_M + C_H$$

$$I. \quad C_{ПП} = 80955,71 + 17810,26 + 101188,5 + 115707,84 = 315662,31 \text{ грн.}$$

4.7 Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$K_{\text{ТЕР}j} = K_{Kj} / C_{\Phi j}$$

$$K_{\text{ТЕР}1} = 19,5 / 414697,55 = 4,7 \cdot 10^{-5}.$$

На основі проведеного аналізу вартості розробки програмного продукту було визначено трудомісткість для двох ключових завдань: розробки проекту програмного продукту та програмної оболонки. Для першого завдання трудомісткість склала 65.94 людино-дні, з урахуванням використання довідкової інформації та стандартних модулів, що відображено поправочним коефіцієнтом 1.021 та коефіцієнтом стандартних модулів 0.6. Друге завдання, з трудомісткістю в 20.41 людино-дні, застосовує алгоритми третьої групи складності з відповідним коефіцієнтом 1.08. Сумарна трудомісткість обох завдань у людино-годинах дорівнює 690.8, що дозволяє оцінити необхідні ресурси та час для реалізації проекту. Загальна вартість розробки програмного продукту становить 41,469,755 гривень, що включає заробітну плату, витрати на оплату машинного часу, соціальні внески та накладні витрати. Значення коефіцієнта техніко-економічного рівня (4.7×10^{-5}) вказує на високу вартість проекту у порівнянні з його технічними параметрами та комерційною цінністю, що вимагає додаткового аналізу ефективності інвестицій та оптимізації проектних витрат.

Висновки до розділу 4

У цьому розділі дипломної роботи проведено докладний аналіз функціональних та вартісних аспектів програмного продукту. Також оцінено основні функції цього програмного продукту.

В результаті аналізу функціональності та вартості програмного комплексу, який розробляється, визначено та проаналізовано його ключові функції, а також ідентифіковано параметри, що характеризують його. На

основі проведеного аналізу був обраний варіант реалізації програмного продукту.

ВИСНОВКИ

У дипломній роботі було розглянуто теоретичні та практичні аспекти задачі оптимального розподілу потоків у мережах, що моделюють промислові, логістичні або інфраструктурні системи. Результатом проведеного дослідження стала побудова математичної моделі, розробка ефективного алгоритму її розв'язання, а також програмна реалізація з можливістю генерації, візуалізації та тестування різних типів мережевих структур.

У першому розділі було здійснено теоретичний огляд існуючих підходів до моделювання потоків у мережах. Детально проаналізовано поняття вузлового балансу, пропускних обмежень на ребрах та функцій витрат. Особливу увагу приділено варіантам побудови цільової функції — зокрема, опуклим нелінійним функціям вартості, які враховують залежність витрат від величини потоку. Було показано, що така модель є більш реалістичною для промислових мереж, де витрати не зростають лінійно.

У другому розділі було сформульовано математичну постановку задачі з використанням методу множників Лагранжа. Запропоновано розв'язання у вигляді двоїстої задачі з урахуванням фіксованих і вільних вузлових параметрів. Теоретично обґрунтовано метод покоординатного градієнтного спуску для оновлення множників Лагранжа, а також формули для обчислення оптимального потоку в кожній дузі графа. Особливої уваги було надано умовам допустимості та збіжності алгоритму, а також ролі параметра кривизни α у формуванні функції витрат.

У третьому розділі здійснено повну програмну реалізацію описаного підходу. У результаті проведених експериментів було підтверджено ефективність та коректність реалізованого алгоритму. Алгоритм успішно масштабувався до графів з сотнями вершин, забезпечував дотримання балансу у вузлах, враховував обмеження пропускної здатності та видавав

зрозумілий візуальний результат. Сумарне споживання повністю задовольнялось, а обчислення виконувались за прийнятний час.

Загалом, результати роботи підтверджують, що розроблений підхід є універсальним, гнучким і придатним до практичного використання. Він може застосовуватись у широкому спектрі задач — від моделювання енергетичних мереж до логістичних і телекомунікаційних систем. Запропонована реалізація дозволяє легко адаптувати модель до нових умов, що відкриває перспективи для подальших досліджень, включно з задачами динамічного перерахунку потоків, стохастичного попиту або багатокритеріальної оптимізації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Conejo A. J., Carrión M., Morales J. M. Decision making under uncertainty in electricity markets. Berlin : Springer, 2010. 296 p.
2. Hitchcock F. L. The distribution of a product from several sources to numerous localities. Journal of Mathematics and Physics. 1941. Vol. 20, No. 3. P. 224–230.
3. Bertsekas D. P., Gallager R. G. Data networks. 2nd ed. Upper Saddle River : Prentice Hall, 1992. 512 p.
4. Wood A. J., Wollenberg B. F. Power generation, operation, and control. 2nd ed. New York : Wiley-Interscience, 1996. 592 p.
5. Ahuja R. K., Magnanti T. L., Orlin J. B. Network flows: theory, algorithms, and applications. Upper Saddle River : Prentice Hall, 1993. 846 p.
6. Kirschen D., Strbac G. Fundamentals of power system ecoNomics. Chichester : Wiley, 2004. 296 p.
7. Ford L. R., Fulkerson D. R. Flows in networks. Princeton : Princeton University Press, 1962. 194 p.
8. Medhi D., Ramasamy K. Network routing: algorithms, protocols, and architectures. Amsterdam : Morgan Kaufmann, 2007. 880 p.
9. Ormsbee L., Lansey K. Optimal control of water distribution systems. Journal of Water Resources Planning and Management. 1994. Vol. 120, No. 2. P. 237–252.
10. Minoux M. Network synthesis and optimum network design problems: models, solution methods and applications. Networks. 1989. Vol. 19, No. 3. P. 313–360.
11. Kennington J. L., Helgason R. V. Algorithms for network programming. New York : Wiley-Interscience, 1980. 360 p.
12. Tassiulas L., Ephremides A. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop

radio networks. IEEE Transactions on Automatic Control. 1992. Vol. 37, No. 12. P. 1936–1948.

13. КП «Міськводоканал» Суми. Проєкти модернізації водовідведення. URL: <https://vodokanal.zt.ua/news/pro-ekologicni-proekti-vodnogo-sektoru-ta-ih-realizaciu-10-uspisnih-prikladiv> (last accessed: 10.05.2025).

14. Децентралізація: сільське водопостачання в рамках проєкту DESPRO. URL: <https://decentralization.ua/news/10175> (last accessed: 10.05.2025).

15. Воловецький В. Б., Щирба О. М. Оптимізація експлуатації газових свердловин. Науково-технічний вісник НУ «Львівська політехніка». 2019 С. 19-28 URL: <https://science.lpnu.ua/sites/default/files/journal-paper/2019/nov/19644/191030budiv-19-28.pdf> (last accessed: 10.05.2025).

16. Sidhant Misra, Michael W. Fisher, Scott Backhaus, Russell Bent, Michael Chertkov, Feng Pan. Geometric optimization of gas pipeline operations. 2014 P. 10 URL: <https://arxiv.org/abs/1312.2668> (last accessed: 10.05.2025).

17. Ye Wang, Kevin Too Yok, Wenyan Wu, Angus R. Simpson, Erik Weyer, Chris Manzie. Economic model predictive control of pumping stations. 2021. P. 20. URL: <https://arxiv.org/abs/2010.07477> (last accessed: 10.05.2025).

18. Кірік О.Є. Нелінійні задачі розподілу потоків у мережах з фіксованими та вільними вузловими параметрами *Системні дослідження та інформаційні технології*. 2002. № 4. С. 106–119.

ДОДАТОК А ЛІСТИНГ КОДУ ПРОГРАМНОГО ПРОДУКТУ

```

import random
import time
from typing import List, Type

import networkx as nx
import pandas as pd
import matplotlib.pyplot as plt

def generate_random_tree(n):
    prufer = [random.randint(0, n - 1) for _ in range(n - 2)]
    degree = [1] * n
    for node in prufer:
        degree[node] += 1

    edges = []
    ptr = 0
    while ptr < len(prufer):
        for node in range(n):
            if degree[node] == 1:
                degree[node] = 0
                nxt = prufer[ptr]
                edges.append((node, nxt))
                degree[nxt] -= 1
                ptr += 1
                break

    last_two = [i for i in range(n) if degree[i] == 1]
    edges.append((last_two[0], last_two[1]))

    G = nx.Graph()
    G.add_nodes_from(range(n))
    G.add_edges_from(edges)
    return G

class BaseFlowGraph:
    def __init__(self, graph, filename_prefix='flow_graph'):
        self.graph = graph

```

```

self.filename_prefix = filename_prefix
self.node_status = {}
self.consumption = {}

self.assign_node_status()
self.ensure_reachability()
self.assign_consumption()
self.balance_consumption()
self.assign_edge_weights()

def assign_node_status(self):
    nodes = list(self.graph.nodes())
    wells = set(random.sample(nodes, max(1, len(nodes) // 6)))
    for node in nodes:
        self.node_status[node] = 'well' if node in wells else 'consumer'

def ensure_reachability(self):
    pass

def assign_consumption(self):
    for v in self.graph.nodes():
        if self.node_status[v] == 'well':
            self.consumption[v] = -random.randint(500, 1000)
        else:
            self.consumption[v] = random.randint(1, 3)

def balance_consumption(self):
    total = sum(self.consumption.values())
    if abs(total) > 1e-12:
        some_well = next(v for v, st in self.node_status.items() if st == 'well')
        self.consumption[some_well] -= total

def assign_edge_weights(self):
    for u, v in self.graph.edges():
        self.graph[u][v]['weight'] = random.uniform(1.0, 3.0)
        self.graph[u][v]['capacity_min'] = 0.0
        self.graph[u][v]['capacity_max'] = random.uniform(900, 1000.0)

def save_graph_image(self, name: str):
    if self.graph.number_of_nodes() > 100:
        return

```

```

pos = nx.spring_layout(self.graph, seed=42)
plt.figure(figsize=(8, 6))

wells = [n for n, st in self.node_status.items() if st == 'well']
consumers = [n for n, st in self.node_status.items() if st != 'well']

nx.draw_networkx_nodes(self.graph, pos, nodelist=wells, node_color='skyblue', node_size=400,
label='Wells')
nx.draw_networkx_nodes(self.graph, pos, nodelist=consumers, node_color='lightcoral',
node_size=400, label='Consumers')
nx.draw_networkx_edges(self.graph, pos, edge_color='gray')

labels = {
    node: f"{self.consumption[node]}" if self.node_status[node] != 'well' else "
    for node in self.graph.nodes()
}
nx.draw_networkx_labels(self.graph, pos, labels=labels, font_size=9, font_color='black')

edge_labels = {
    (u, v): f"{self.graph[u][v]['weight']:.2f}"
    for u, v in self.graph.edges()
}
nx.draw_networkx_edge_labels(self.graph, pos, edge_labels=edge_labels, font_color='green',
font_size=8)

plt.axis('off')
plt.title(f"{name} - Pre-Optimization Graph")
plt.tight_layout()
plt.savefig(f"{name}_graph.png")
plt.close()

class TreeFlowGraph(BaseFlowGraph):
    def __init__(self, n):
        g = generate_random_tree(n)
        super().__init__(g)

class GridFlowGraph(BaseFlowGraph):
    def __init__(self, n):
        size = int(n**0.5)
        g = nx.grid_2d_graph(size, size)

```

```

g = nx.convert_node_labels_to_integers(g)
super().__init__(g)

```

```

class WattsFlowGraph(BaseFlowGraph):
    def __init__(self, n):
        g = nx.watts_strogatz_graph(n, k=3, p=0.3)
        super().__init__(g)

```

```

class RandomFlowGraph(BaseFlowGraph):
    def __init__(self, n):
        g = generate_random_tree(n)
        for _ in range(n // 4):
            u, v = random.sample(list(g.nodes()), 2)
            if not g.has_edge(u, v):
                g.add_edge(u, v)
        super().__init__(g)

```

```

class NonlinearOptimizer:
    def __init__(self, flow_graph: BaseFlowGraph):
        self.graph = flow_graph.graph
        self.consumption = flow_graph.consumption.copy()
        self.node_status = flow_graph.node_status
        self.lambdas = {v: 0.0 for v in self.graph.nodes()}

        self.r_min = {
            (u, v): self.graph[u][v]['capacity_min'] for u, v in self.graph.edges()
        }
        self.r_max = {
            (u, v): self.graph[u][v]['capacity_max'] for u, v in self.graph.edges()
        }
        self.l_coeff = {
            (u, v): self.graph[u][v]['weight'] for u, v in self.graph.edges()
        }

    def edge_flow(self, u, v):
        lam_u = self.lambdas[u]
        lam_v = self.lambdas[v]
        l_ij = self.l_coeff[(u, v)]
        rmin = self.r_min[(u, v)]

```

```

rmax = self.r_max[(u, v)]

x = (lam_u - lam_v) / l_ij
if x < rmin:
    return rmin
if x > rmax:
    return rmax
return x

def solve_lambda_for_node(self, v):
    demand = self.consumption[v]
    low, high = -1e6, 1e6
    for _ in range(500):
        mid = 0.5 * (low + high)
        self.lambdas[v] = mid
        net = 0.0
        for u in self.graph.neighbors(v):
            if (u, v) in self.l_coeff:
                net += self.edge_flow(u, v)
            if (v, u) in self.l_coeff:
                net -= self.edge_flow(v, u)
        if net > demand:
            low = mid
        else:
            high = mid
    self.lambdas[v] = 0.5 * (low + high)

def optimize(self, max_iter=2000, tol=1e-7):
    start_time = time.time()
    iter_count = 0
    for _ in range(max_iter):
        max_err = 0.0
        for v in self.graph.nodes():
            if self.node_status[v] != 'well':
                old = self.lambdas[v]
                self.solve_lambda_for_node(v)
                max_err = max(max_err, abs(self.lambdas[v] - old))
        iter_count += 1
        if max_err < tol:
            break

    flows = {}

```

```

for u, v in self.graph.edges():
    flows[(u, v)] = self.edge_flow(u, v)

total_cost = 0.0
for (u, v), x in flows.items():
    total_cost += 0.5 * self.l_coeff[(u, v)] * x * x

total_time = time.time() - start_time
return flows, total_cost, iter_count, total_time

```

```

def visualize_flow(graph: nx.DiGraph, flows: dict, node_status: dict):
    pos = nx.spring_layout(graph, seed=42)
    plt.figure(figsize=(8, 6))

    wells = [n for n, st in node_status.items() if st == 'well']
    consumers = [n for n, st in node_status.items() if st != 'well']

    nx.draw_networkx_nodes(graph, pos, nodelist=wells, node_color='skyblue', node_size=400,
label='Wells')
    nx.draw_networkx_nodes(graph, pos, nodelist=consumers, node_color='lightcoral', node_size=400,
label='Consumers')

    all_edges = list(flows.keys())
    flow_values = [flows[edge] for edge in all_edges]
    max_flow = max(flow_values) if flow_values else 1.0
    widths = [1.0 + 8.0 * (fv / max_flow) for fv in flow_values]

    nx.draw_networkx_edges(graph, pos, edgelist=all_edges, width=widths, edge_color='grey', alpha=0.7)
    nx.draw_networkx_labels(graph, pos, font_size=9, font_color='black')

    edge_labels = {edge: f'{flows[edge]:.2f}' for edge in all_edges}
    nx.draw_networkx_edge_labels(graph, pos, edge_labels=edge_labels, font_color='green', font_size=7)

    plt.axis('off')
    plt.legend(scatterpoints=1)
    plt.title("Optimized Flow")
    plt.show()

```

```

class FlowOptimizerEvaluator:
    def __init__(self, graph_classes: List[Type[BaseFlowGraph]], sizes: List[int]):

```

```

self.graph_classes = graph_classes
self.sizes = sizes
self.results = []

def test_all(self):
    for graph_cls in self.graph_classes:
        for size in self.sizes:
            graph = graph_cls(size)
            name = f"{graph_cls.__name__}-{size}"
            # graph.save_graph_image(name)

            DG = nx.DiGraph()
            DG.add_nodes_from(graph.graph.nodes(data=True))
            for u, v in graph.graph.edges():
                DG.add_edge(u, v, **graph.graph[u][v])
                DG.add_edge(v, u, **graph.graph[u][v])
            graph.graph = DG

            optimizer = NonlinearOptimizer(graph)
            flows, cost, iters, duration = optimizer.optimize()

            sum_demand = sum(v for v in graph.consumption.values() if v > 0)

            delivered = sum(
                val for (u, v), val in flows.items() if graph.node_status[v] != 'well'
            )

            flow_matrix = pd.DataFrame(0.0, index=graph.graph.nodes(), columns=graph.graph.nodes())

            for (u, v), f in flows.items():
                flow_matrix.at[u, v] = round(f, 6)
            flow_matrix.to_excel(f"{name}_flows.xlsx")

            total_cost = cost + sum_demand

            self.results.append({
                'graph': name,
                'nodes': size,
                'iterations': iters,
                'time_sec': round(duration, 2),
                'total_cost': round(total_cost, 2),
                'sum_demand': round(sum_demand, 2),
            })

```

```
        'delivered': round(delivered, 2),
        'estimated_losses': cost
    })

    if size < 5:
        print(f"\nVisualization for {graph_cls.__name__} n={size}")
        visualize_flow(graph.graph, flows, graph.node_status)

def get_results(self):
    return pd.DataFrame(self.results)

def save_results(self):
    pd.DataFrame(self.results).to_excel('experiment.xlsx')
```

ДОДАТОК Б ПРЕЗЕНТАЦІЯ

«Розподіл потоків в великих мережах промислового призначення з оптимальним перерозподілом джерел постачання»

Виконав студент IV курсу,
Групи КА-12 Рудий Дмитро Ігорович

Науковий керівник
Касьянов Павло Олегович

Об'єкт, предмет, мета дослідження

Об'єкт дослідження — потоки ресурсів у мережах промислового типу з урахуванням структури графа, обмежень на потоки та динамічних характеристик джерел і споживачів.

Предмет дослідження — математичні методи оптимізації потоків у мережах, зокрема побудова двоїстої задачі на основі методу множників Лагранжа та алгоритми чисельного розв'язання з використанням потенціалів вузлів.

Мета роботи — розробка та реалізація математичної моделі оптимального розподілу потоків у мережах із частково змінними вузловими подачами, побудова відповідної двоїстої задачі та апробація ефективного алгоритму її розв'язання на тестових графах різної топології.



Постановка задачі

- Дослідити математичну модель для оптимізації розподілу потоків на графах
- Створити механізм генерації графів різної топології для тестування роботи алгоритму
- Реалізувати алгоритм оптимізації
- Створити модуль для зручного використання методу та гнучкого експорту результатів

3



Актуальність теми

У сучасних технічних, логістичних та енергетичних системах дедалі гострішою стає потреба в ефективному управлінні потоками ресурсів у складних розгалужених мережах. Водопостачання, електромережі, газотранспортні системи, транспортна логістика, телекомунікаційні інфраструктури — всі ці сфери функціонують на основі мережових структур, у межах яких здійснюється передача ресурсів між джерелами та споживачами. Зростання масштабу систем, складності конфігурацій і динамічності змін вимагає застосування нових математичних методів, що здатні забезпечити оптимальне планування розподілу потоків у таких мережах. Особливої актуальності задача набуває в умовах обмежень — як технічних (наприклад, пропускна здатність трубопроводів чи кабелів), так і економічних (витрати на транспортування, енергоспоживання, втрати).

4

Теоретичні основи: Постановка задачі

При формалізації задач оптимального розподілу потоків у мережах важливо надати точне абстрактне представлення технічної системи, яка складається з джерел (колодязів), споживачів, каналів передачі ресурсу та обмежень, які регулюють ці потоки. Одним із найбільш зручних і загальноприйнятих підходів є подання системи у вигляді орієнтованого графа $G = (N, V)$, де:

- N — множина вершин, що представляють елементи інфраструктури: джерела, споживачі, проміжні вузли;
- V — множина дуг (ребер), що моделюють можливі напрямки передачі потоку між вузлами.

У цій моделі джерела (колодязі) — це вузли, які мають надлишок ресурсу, який потрібно передати в систему. Наприклад, у водопостачанні це насосні станції чи водозабори; у газопостачанні — газові свердловини або компресорні станції. Для кожного джерела і визначено параметр r_i , який означає обсяг ресурсу, що постачається.

Споживачі — це вузли, що поглинають ресурс. Вони моделюються вершинами з параметром d_i , який визначає потребу у ресурсі. Наприклад, це житлові квартали, промислові підприємства або об'єкти критичної інфраструктури

5

Теоретичні основи: Формалізація умов

Для аналізу та розв'язання задачі оптимального розподілу потоків з частково змінними правими частинами використовується підхід на основі побудови двоїстої задачі. Це дозволяє перейти до оптимізації по потенціалах вузлів, що спрощує обчислення в багатьох практичних випадках.

Розглянемо цільову функцію, яка вираховується за формулою (2.17):

$$F(x) = \sum_{(i,j) \in V} \int_0^{x_{ij}} f_{ij}(t) dt \quad (2.17)$$

де $f_{ij}(t)$ — строго зростаючі, неперервні функції витрат.

Накладаються обмеження неперервності у вузлах, які вираховується за формулою (2.18):

$$\sum_{j: (i,j) \in V} x_{ij} - \sum_{j: (j,i) \in V} x_{ji} = d_i \quad (2.18)$$

для всіх $i \in N$,

де d_i — фіксовані або змінні праві частини.

Щоб сформулювати двоїсту задачу, будеться функція Лагранжа, яка вираховується за формулою (2.19):

$$L(x, p) = \sum_{(i,j) \in V} \int_0^{x_{ij}} f_{ij}(t) dt + \sum_{i \in N} p_i \left(d_i - \sum_{j: (i,j) \in V} x_{ij} + \sum_{j: (j,i) \in V} x_{ji} \right) \quad (2.19)$$

де p_i — множники Лагранжа (інтерпретуються як потенціали вузлів).

6

Практична реалізація

Структура програми

1. Генерація графів:

Реалізовано універсальну архітектуру з базовим класом BaseFlowGraph та чотирма нащадками, що відповідають топологіям Ваттса-Строгатца, дерева, решітки та випадкового графа. Це забезпечило гнучке моделювання як ієрархічних, так і хаотичних структур. Автоматична орієнтація графа та маркування вузлів (коподязі й споживачі) виконувалися за логічними правилами, що гарантувало цілісність структури. Вбудовано перевірку досяжності споживачів із джерел і контроль параметрів дуг (пропускна здатність, вартість).

2. Оптимізація потоків:

Оптимізаційний модуль `NonlinearFlowOptimizer` реалізований на основі методу множників Лагранжа. Передбачено чіткий поділ етапів: обчислення потоків, оновлення множників, градієнтні кроки. Алгоритм узгоджений з теоретичною базою й ефективно забезпечує баланс у вузлах та дотримання обмежень на дугах протягом численних ітерацій.

3. Виведення результатів:

Результати оптимізації візуалізовано за допомогою кольорової схеми з маркуванням значень потоків, що дозволяє легко аналізувати структуру, напрямки потоків і виявляти проблемні місця. Реалізовано збереження графів у форматах PNG, JSON і GraphML для подальшої інтеграції з іншими системами.

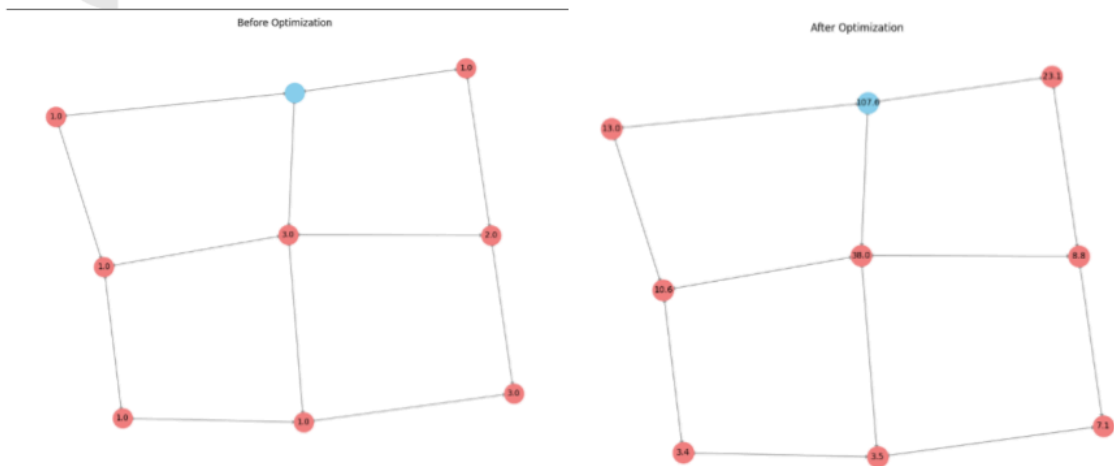
7

Результати: Згенеровані графи

Зображення графів до і після оптимізації

Синя вершина - колодязь

Червоні вершини - споживачі



8

Результати: Баланс речовини в вузлах

n	type	in	out	net	losses	need	supply
0	consumer	3,86	2,86	1	22,11	1	23,11
1	well	0	13	-13	0	0	107,55
2	consumer	4,21	3,21	1	12,01	1	13,01
3	consumer	3,61	1,61	2	6,81	2	8,81
4	consumer	5,67	2,67	3	34,99	3	37,99
5	consumer	3,21	2,21	1	9,59	1	10,59
6	consumer	3	0	3	4,14	3	7,14
7	consumer	2,39	1,39	1	2,54	1	3,54
8	consumer	1,47	0,47	1	2,37	1	3,37

9

Результати: Розподіл потоків

	0	1	2	3	4	5	6	7	8
0				8,97					
1	25,97		16,22		39,1				
2						12,8			
3							4		
4				1,45				4,16	
5					1,55				3,84
6									
7							3,13		
8								0,77	

10

Результати: Підсумок експерименту

graph	nodes	iterations	time	cost	demand	actual	losses
TreeFlowGraph-10	10	473	4,79	525,98	14	14	511,9839741
TreeFlowGraph-20	20	325	13,74	537,72	37	37	500,718083
TreeFlowGraph-100	100	1456	354,92	8949,08	179	179	8770,079666
TreeFlowGraph-500	500	1990	888,58	29740,93	817	817	28923,93217
GridFlowGraph-10	10	153	2,07	151,17	17	17	134,1747544
GridFlowGraph-20	20	187	4,89	387,4	30	30	357,4043438
GridFlowGraph-100	100	147	19,28	1093,31	177	177	916,3137654
GridFlowGraph-500	500	215	121,34	5312,84	820	820	4492,836247

11

Висновки

У дипломі розглянуто задачу оптимального розподілу потоків у мережах, які моделюють промислові та логістичні системи.

Досліджено предметну область, популярні шляхи вирішення поставленої задачі, вибрано оптимальний.

Розроблено математичну модель з урахуванням вузлового балансу, обмежень та втрат. Побудовано двоїсту задачу з використанням методу Лагранжа. Запропоновано ефективний алгоритм покоординатного спуску для оновлення потенціалів вузлів.

Алгоритм реалізовано програмно з підтримкою генерації, візуалізації та масштабування на великі графи. Результати показали правильність та гнучкість методу. Підхід придатний до застосування у різних типах мереж.

12

