Задачі оптимального проектування надійних мереж

ЗАДАЧІ ОПТИМАЛЬНОГО ПРОЕКТУВАННЯ НАДІЙНИХ МЕРЕЖ

За загальною редакцією
академіка НАН України Н.З. Шора

КИЇВ НАУКОВА ДУМКА 2005

УДК 519.854

Книгу надруковано завдяки фінансовій підтримці Науково-технологічного центру в Україні

У книзі подано опис математичного і програмного забезпечення для ряду задач оптимального проектування та маршрутизації в мережах із врахуванням можливого виходу з ладу окремих компонент мережі та змін вимог до потоків. Зокрема, розглянуто такі важливі з точки зору практичних потреб задачі на мережах: проектування мережі мінімальної вартості за умови виходу з ладу окремих ланок, знаходження пропускних здатностей дуг надійної орієнтованої мережі, проектування оптимальної логічної структури надійної мережі, модернізація надійної мережі, оптимізація мереж із врахуванням неповної інформації, перспективне планування перевезень, знаходження оптимальної номенклатури рухомого складу. Описано методи недиференційованої оптимізації, методи локального дискретного пошуку та схеми декомпозиції, використані для оптимізації надійних мереж.

Для інженерів і науковців, студентів, аспірантів та викладачів вищих учбових закладів.

Автори:
Н.З. Шор, І.В. Сергієнко, В.П. Шило, П.І. Стецюк, І.М. Парасюк, Т.Т. Лебедєва, Ю.П. Лаптін, М.Г. Журбенко, Т.О. Бардадим, Ф.А. Шаріфов, О.П. Лиховид, О.А. Березовський, В.М. Мірошниченко

Редакція фізико-математичної та технічної літератури
Редактор *Т.С. Мельник*

$3\frac{1602110000}{2005}$

ISBN 966-00-0606-3

Передмова

Книга написана колективом авторів під загальною редакцією академіка НАН України Наума Зуселевича Шора на основі результатів досліджень, виконаних за проектом Науково-технологічного центру в Україні "Математичні та програмні засоби оптимального проектування структур надійних мереж" (№ 1625). Вона містить приклади використання сучасних методів оптимізації для розв'язання складних практичних задач, що постійно виникають на транспорті та комунікаційних мережах. Зокрема, розглянуто такі важливі з точки зору практичних потреб задачі на мережах, як проектування мережі мінімальної вартості за умови виходу з ладу окремих ланок, знаходження пропускних здатностей дуг надійної орієнтованої мережі, проектування оптимальної логічної структури надійної мережі, модернізація надійної мережі, оптимізація мереж із врахуванням неповної інформації, перспективне планування перевезень, знаходження оптимальної номенклатури рухомого складу. Описано методи недиференційованої оптимізації, методи локального дискретного пошуку та схеми декомпозиції, використані для оптимізації надійних мереж.

Всі автори – співробітники Інституту кібернетики імені В.М. Глушкова НАНУ – є спеціалістами в галузі оптимізації. Розділ 1 написали Н.З. Шор, М.Г. Журбенко, О.П. Лиховид та П.І. Стецюк, розділ 2 підготовлено за результатами досліджень Н.З. Шора і Ю.П. Лаптіна, а розділ 3 – І.В. Сергієнка, В.П. Шила, Т.Т. Лебєдєвої і І.М. Парасюка. Розділ 4, що складається з чотирьох підрозділів, написали: 4.1 – П.І. Стецюк, 4.2 – І.В. Сергієнко, В.П. Шило, Т.Т. Лебєдєва і І.М. Парасюк, 4.3 – Ю.П. Лаптін, 4.4 – О.П. Лиховид. Розділ 5 та програмне забезпечення до описаної в ньому задачі перспективного планування перевезень підготували М.Г. Журбенко, Ю.П. Лаптін

і В.М. Мірошниченко. В підготовці тексту розділу 6, що містить результати Ф.А. Шаріфова, взяли участь також Т.О. Бардадим та О.А. Березовський. Опис програмного забезпечення в розділі 7 зробили Ю.П. Лаптін, П.І. Стецюк, М.Г. Журбенко, В.П. Шило, О.П. Лиховид і Т.О. Бардадим.

Програмне забезпечення, створене в рамках цього проекту, розробили М.Г. Журбенко, Ю.П. Лаптін, П.І. Стецюк, О.П. Лиховид і В.П. Шило за допомогою мов програмування ФОРТРАН та С++. Вступ написала Т.О. Бардадим, спираючись на матеріали, надані іншими авторами. У вигляді книги всі матеріали досліджень були впорядковані Т.Т. Лебедєвою, О.П. Лиховидом і Т.О. Бардадим.

Сподіваємося, що ця монографія буде сприяти створенню надійних мереж.

Колектив авторів висловлює подяку колаборатором по проекту, особливо К. Холмбергу, О.П. Бурдакову і П.О. Ліндбергу (університет м. Лінчепінг, Швеція), М. Сміту і М. Додсону (університет м. Йорк, Велика Британія). При створенні книги дуже допомогли консультації та бібліографічні матеріали, які надали А.Р. Маджуб (університет ім. Блеза Паскаля, м. Клермон-Ферран, Франція) та В. Бен-Амер (Національний інститут телекомунікацій, м. Еврі, Франція).

Автори щиро вдячні Науково-технологічному центру в Україні за надану можливість виконати це дослідження, зокрема координатору проекта І.П. Томашевській за постійну доброзичливу підтримку, Інституту кібернетики ім. В.М. Глушкова НАН України за увагу й організаційну допомогу, працівникам видавництв "Наукова думка" та "КИТ" за їх ретельність та професіоналізм.

Київ, червень 2004

Вступ

Математичні методи і програмне забезпечення для ряду задач оптимального проектування та маршрутизації в мережах із врахуванням можливого виходу з ладу окремих компонент мережі та зміни вимог до потоків мають дуже важливе значення для практичного застосування. Подібні оптимізаційні задачі у зв'язку з їх складністю та актуальністю, постійно є предметом досліджень фахівців як в Україні, так і за кордоном. Важливою особливістю сучасних комунікаційних мереж є використання нових технологій для передачі інформації, наприклад оптичноволоконних. Це забезпечує надзвичайно високу передавальну здатність, і, коли окремі елементи виходять з ладу, може статися, що буде втрачена життєво важлива інформація. Тому питання надійності є важливим фактором при проектуванні надійних систем зв'язку. Аварії на трубопроводах завдають значних збитків, бо приводять до забруднення довкілля. Підвищення надійності транспортних мереж сприяє безпеці дорожнього руху. Проблеми, пов'язані із забезпеченням надійності функціонування мереж, становлять складають надзвичайно широку сферу досліджень. Вона включає як вивчення можливостей створення працездатних мереж при можливих аваріях на окремих ланках мережі, так і інженерні дослідження, пов'язані з покращенням якості конструкцій окремих фізичних елементів мереж.

Нагальна потреба розв'язувати практичні задачі на мережах різних типів сприяла розвитку досліджень у багатьох галузях теорії оптимізації та дослідження операцій. Це насамперед теоретичне вивчення мережевих задач та суміжних питань дискретної оптимізації (згадаємо, наприклад, монографії [53, 18, 167, 13, 35], див. також "Енциклопедію оптимізації" [84]). З іншого боку, це розробка ефективних евристичних

алгоритмів та систем автоматичного регулювання дорожнім рухом (див. [80, 55, 154, 73, 152, 123, 124, 157–160, 133]). Окрему групу становлять дослідження, які враховують випадковий характер задач (див., наприклад, [83, 134, 112]). Суттєвий практичний інтерес викликають питання, пов'язані з надійністю, "живучістю" мереж. (З величезної кількості праць із цих питань нагадаємо тут, зокрема, праці [166, 98, 103, 155, 70, 120, 87, 171].) Хотілося б звернути увагу також на цікаві приклади використання лагранжевого підходу, методу гілок та границь і декомпозиції в працях [108–111].

Давно вже теоретично обгрунтовано той факт, що навіть звичайні математичні моделі для задач оптимізації на мережах, особливо в цілочисельному випадку, є надзвичайно складними (див., наприклад, [1, 53]). Більшість із них належить до так званого класу NP–складних задач, і для їх розв'язування часто використовують загальну схему методу гілок і границь. Ефективність цього підходу залежить від ефективності знаходження верхньої і нижньої оцінок по функціоналу. В.С. Михалевич, В.О. Трубін, Н.З. Шор [18] і Ф. Шаріфов [39, 38], враховуючи сфецифіку задач синтезу мереж, розробили ефективні алгоритми розв'язання важливих класів таких задач. Ці алгоритми демонстрували високу ефективність як на експериментальних даних, так і на реальних. Оптимізаційні моделі подібних задач характеризуються великою вимірністю і спеціальною структурою матриці обмежень. Відомі методи для їх розв'язання є надзвичайно трудомісткими і часто не гарантують необхідної точності. Врахування ступеня надійності елементів мережі та можливості зміни її структури суттєво збільшують складність розв'язування розглянутих класів задач оптимального проектування.

Як правило, мережеві задачі формалізуються у вигляді задач математичного програмування (найчастіше, лінійного чи цілочисельного), і ефективність їх розв'язання істотно залежить від того, які методи при цьому використовуються. На сьогодні існують потужні програмні засоби розв'язування задач

математичного програмування (наприклад, CPLEX, MINOS, SOPLEX, PCX, HOPDM тощо). Згадані пакети досить універсальні, розраховані на розв'язання широких класів задач, але вони не враховують специфічні особливості мережевих задач. Тому програмне забезпечення загального призначення часто буває неефективним або навіть неприйнятним для розв'язування розглянутих у цій книзі мережевих задач великої вимірності. Для їх ефективного розв'язування можна запропонувати спеціалізовані методи, що враховують специфіку задачі.

Основною особливістю підходу, що пропонується, є використання алгоритмів негладкої оптимізації, а саме розробленого в Інституті кібернетики сімейства чисельних методів з розтягом простору (*r*-алгоритми) [45, 153] в комбінації з декомпозицією, тобто прийомом, який дає можливість замінити розв'язання складної задачі послідовністю розв'язувань простіших задач (підзадач). Цей підхід був ефективно використаний для розв'язання ряду задач оптимізації мереж без врахування питань їх надійності. Відповідні результати наведені в ряді праць, наприклад [18, 27, 23]. У даній книзі цей підхід застосовано для розв'язування оптимізаційних задач, пов'язаних із забезпеченням надійної роботи мереж.

Поданий у книзі матеріал поділений на дві частини: загально-теоретичну та безпосередній розгляд мережевих задач. Розділ 1 присвячений обчислювальним методам з розтягом простору. Наведено загальну схему *r*-алгоритму (п. 1.1.2), міркування щодо розтягу простору, які лежать у його основі (п. 1.1.1) та опис двох його модифікацій: монотонної та з адаптивним регулюванням довжини кроку (п. 1.1.3 та 1.1.4). Ефективність роботи цих алгоритмів ілюструється рядом обчислювальних експериментів, виконаних на складних для розв'язування оптимізаційних задачах (п. 1.1.5).

Новий алгоритм ε-субградієнтної мінімізації описано в підрозділі 1.2. Він також належить до класу методів із перетворенням простору. На кожній ітерації алгоритму процес перетворення здійснюється при застосуванні операторів

розтягу простору вздовж ортогональних напрямів. Параметри перетворення визначаються побудовою еліпсоїдів, в яких локалізовано ε-розв'язок. На кожній ітерації забезпечується зменшення об'єму області локалізації ε-розв'язку принаймні в задане число раз (яке є одним із параметрів алгоритму). Наведено результати чисельного дослідження ефективності ε-субградієнтного алгоритму.

Завершує розділ 1 опис модифікації методу еліпсоїдів. Замість еліпсоїда мінімального об'єму, описаного навколо напівкулі в n-вимірному евклідовому просторі E^n, пропонується використовувати інший еліпсоїд, параметри якого близькі до тих, що використовуються в методі еліпсоїдів. Показано, що запропонована модифікація методу еліпсоїдів має таку ж швидкість збіжності, як і класичний метод еліпсоїдів. Цією модифікацією, на відміну від методу еліпсоїдів, можна користуватися і в одновимірному випадку.

Розділ 2 присвячено питанням декомпозиції структурованих задач оптимізації. Огляд різних алгоритмів декомпозиції в задачах оптимізації великої розмірності можна знайти, наприклад, в [37, 32, 45, 153]. Ці алгоритми застосовуються переважно до задач лінійного програмування спеціальної структури. В розділі 2 розглянуто дві схеми декомпозиції – за обмеженнями та за змінними. (В зарубіжній літературі першу називають декомпозицією Данціга-Вулфа, а другу – декомпозицією Бендерса [60].) Тут наведено як відомі результати – вони необхідні для подання решти матеріалу, так і нові, що стосуються застосування декомпозиції за змінними в нелінійних задачах опуклого програмування [15, 16]. За допомогою цих результатів можна використовувати наближені розв'язки внутрішніх підзадач декомпозиції замість точних і обчислювати ε-субградієнти функціонала координуючої задачі. Істотна проблема полягає в тому, що підзадачі, які формуються, мають розв'язки не при всіх значеннях зв'язуючих змінних. Це ускладнюється тим, що і функції, які входять у підзадачі, визначені на обмежених множинах.

Спеціальне введення допоміжних змінних і штрафних функцій дає можливість регуляризувати початкову задачу й уникнути вказаних проблем.

В розділі 3 наведено кілька підходів до розв'язування дискретних оптимізаційних задач. Зокрема, подано описи метода вектора спаду (підрозділ 3.1), методу відпалу (підрозділ 3.2), методу табу (підрозділ 3.3) та методу глобального рівноважного пошуку (підрозділ 3.4). Всі вони належать до методів локального пошуку, які виявилися ефективними при розв'язуванні дискретних задач оптимізації, в тому числі оптимізації мереж. Докладний опис методів локальної оптимізації можна знайти в працях [27, 23, 26]. Для цих методів загальною проблемою є вибір стратегії пошуку, правильного поєднання розширення і звуження області пошуку з метою отримання глобального оптимуму. Задача полягає в тому, щоб після отримання деякого локального оптимуму перейти в область тяжіння іншого локального оптимуму. Для цього необхідно розширити область пошуку (так звана диверсифікація пошуку). З іншого боку, для поліпшення отриманого розв'язку (локально неоптимального) логічно досліджувати точки, близькі до цього розв'язку, тобто звузити область пошуку (інтенсифікація пошуку). Задача знаходження необхідного балансу між розширенням і звуженням області пошуку розв'язку є найважливішою проблемою для локальних методів.

Схема методу вектора спаду дає можливість створювати цілий спектр алгоритмів розв'язання конкретної задачі дискретної оптимізації та після проведення відповідного аналізу вибирати найкращий із них.

На відміну від звичайного методу локального пошуку, який дозволяє здійснювати перехід тільки в "кращу точку", в методі відпалу допускаються імовірнісні переходи і в "гірші точки". Це дає можливість уникати поганого локального оптимуму і знаходити розв'язки, близькі за значенням цільової функції до глобального оптимуму. Метод відпалу успішно застосовувався для розв'язання багатьох складних оптимізаційних

задач [52]. Але його асимптотична ефективність нижча [82], ніж навіть у тривіального методу повторюваного випадкового локального пошуку (RLMS).

Метод табу був запропонований Гловером [92, 93] як детермінований варіант методу відпалу, в якому пам'ять, а не імовірність грає провідну роль у переходах. Щоб уникнути зациклення, проміжні розв'язки явно або неявно запам'ятовуються, і в алгоритмі на певну кількість ітерацій забороняються переходи в ці табуйовані розв'язки.

Метод глобального рівноважного пошуку зводиться до адаптивного випадкового пошуку розв'язку задач дискретної оптимізації (див. [42]). Ідейно близький до методу відпалу [122], він має всі переваги останнього і при цьому характеризується вищою асимптотичною ефективністю.

Друга частина книги присвячена безпосередньому розгляду мережевих задач оптимізації вказаних типів. Оскільки математичні моделі та аппарат досліджень для різних розглянутих задач істотно відрізняються одне від одного, докладний опис кожної з них наведено у відповідних підрозділах розділу 4.

У підрозділі 4.1 розглянуто підходи до задачі знаходження пропускних здатностей дуг надійної орієнтованої мережі при можливості пошкодження окремих її елементів, а саме – у випадку передачі потоків по довільних шляхах (див. п. 4.1.1) та у випадку передачі потоків по заданій множині допустимих шляхів (див. п. 4.1.2). Надійною буде вважатися така мережа, для якої можна задовольнити всі задані вимоги на передачу обсягів потоків у мережі як при відсутності пошкоджень у мережі, так і в тому випадку, коли станеться тільки одне пошкодження, але довільне з усіх можливих пошкоджень орієнтованої мережі. Пошкодженням мережі будемо називати такий її стан, при якому зменшується пропускна здатність однієї або кількох її дуг. В результаті розв'язання наведених задач отримуємо значення мінімальних за сумарною вартістю пропускних здатностей, що доповнюють вже існуючі та забезпечують надійність мережі або відповідь, що забезпечити її неможливо.

Підрозділ 4.2 присвячено моделям проектування оптимальної логічної структури мережі. Тут розглянуто питання про можливість перебудови структури при аварійній ситуації. Це пов'язано з тим, що, наприклад, у випадку систем зв'язку при виході з ладу деяких їх компонент змінюються не всі інформаційні потоки, а тільки ті, які пов'язані з пошкодженими компонентами. Перенаправлення інформаційних потоків відбувається відповідно до прийнятого в мережі зв'язку алгоритму відновлення, зокрема розглянуто алгоритми, що базуються на відновленні шляху.

У підрозділі 4.3 вивчено модель модернізації надійних мереж. У загальному випадку якість функціонування мережі описується векторним критерієм, в який входять капітальні витрати, експлуатаційні витрати в нормальному режимі, експлуатаційні витрати в кожному з аварійних режимів тощо. Тут зроблено припущення, що як цільова функція використовується деяка згортка всіх критеріїв.

Підрозділ 4.4 присвячено проблемі оптимізації мереж із врахуванням неповної інформації. Математичні моделі є узагальненням задач, розглянутих в п. 4.1.1, 4.1.2, на випадок, коли такі характеристики мережі як вимоги до потоків та ступінь працездатності ребер носять прогнозний характер. Вихід із ладу окремих ланок чи будь-яка інша причина, що призводить до істотних змін стану мережі, є, як правило, подією випадковою. Таку ситуацію можна описати за допомогою двохетапних задач стохастичного програмування. Для їх розв'язання використовується схема декомпозиції за змінними та r-алгоритм.

Математична модель, наведена в розділі 5, призначена для розв'язання задач перспективного планування перевезень та раціонального розподілу коштів на реконструкцію транспортних мереж (залізничних, автомобільних, авіаційних). Реалізація моделі дає можливість одержати чисельну інформацію з питань перспективного планування, що стосуються: критичних за пропускними здатностями вузлів та раціонального розподілу коштів на їх реконструкцію; раціональної схеми потоків

транспортних засобів із врахуванням мінімізації експлуатаційних витрат і обсягів порожніх потоків; знаходження оптимального складу парку транспортних засобів та раціонального використання коштів для його розширення. У розділі 5 наведено характеристику відповідних задач оптимізації й принципи побудови методів їх розв'язання. Для визначеності термінології модель побудовано для задачі перспективного планування вантажних залізничних перевезень та модернізації залізничної транспортної системи, проте її можна використовувати і для інших типів перевезень.

В розділі 6 розглянуто задачу побудови надійної зв'язної мережі (ЗПНЗМ). Задано неорієнтований граф G (його ребрам приписано невід'ємні ваги) та множину термінальних вузлів N. Потрібно знайти мережу мінімальної вартості (підграф графа G) при умові, що вона залишається зв'язною навіть при вилученні з графа G будь-якого підграфа, ізоморфного деякому заданому графу H. Досліджено питання існування розв'язку цієї задачі для випадків, коли граф H є деревом або паросполученням з фіксованими вершинами, але кількість вершин не задано. Показано, що задача Штейнера на графі, задача комівояжера, задача синтезу 2-зв'язної мережі, варіант задачі Штейнера для 2-зв'язної мережі, гра Шеннона другого типу та деякі інші задачі є окремими випадками цієї задачі.

Запропоновано алгоритм розв'язання задачі для випадку, коли вийти з ладу може будь-яке одне ребро, а задані початкові вимоги потрібно виконати навіть у такій ситуації. Проведено експериментальне дослідження його ефективності.

Після того як знайдено мережу мінімальної вартості, розв'язується задача визначення різних шляхів для передачі заданих вимог від одного термінального вузла до іншого таким чином, щоб мінімізувати загальну вартість передачі заданого потоку між цими вузлами. Розв'язок цієї задачі можна знайти за допомогою поліноміального алгоритму, запропонованого Шаріфовим в праці [39].

Завершує книгу розділ 7, де міститься стислий опис програмного забезпечення, створеного в Інституті кібернетики

ім. В.М. Глушкова НАН України для розв'язування розглянутих задач оптимального проектування надійних мереж. Оскільки технічні характеристики розглянутих задач ставлять різні вимоги до інтерфейсу користувача, було створено три програмних блоки з окремими інтерфейсами. Перший об'єднує програмні модулі для розв'язання задач проектування мережі мінімальної вартості за умови виходу з ладу окремих ланок, знаходження пропускних здатностей дуг надійної орієнтованої мережі (для двох випадків: коли для перенаправлення потоків можна використовувати будь-які шляхи та коли лише шляхи із заданого списку дозволених шляхів) та задачі оптимізації мереж із врахуванням неповної інформації. Другий інтерфейс було спеціально розроблено для задач проектування оптимальної логічної структури надійної мережі. Він включає розвинуті засоби для роботи з картою, необхідні для розв'язування практичних задач цього типу. Третій інтерфейс було розроблено для задач перспективного планування перевезень, знаходження оптимальної номенклатури рухомого складу.

Наведені в книзі результати можуть виявитися корисними не тільки для розв'язування задач оптимального проектування надійних мереж, але й для розробки відповідних нових обчислювальних алгоритмів.

Частина I

Методи оптимізації в задачах проектування мереж

Розділ 1. Використання методів недиференційованої оптимізації в задачах проектування мереж

У розділі описано ряд методів мінімізації недиференційовних опуклих функцій. Це – сімейство методів із розтягом простору в напрямі різниці двох послідовних субградієнтів (так звані *r*-алгоритми), сімейство нових алгоритмів ε-субградієнтної мінімізації та нова модифікація відомого методу еліпсоїдів. В останніх також використовується операція розтягу простору з метою прискорення їх збіжності.

1.1. *r*-Алгоритм та його модифікації

У даному підрозділі описано дві модифікації *r*-алгоритмів, які було застосовано до координуючих задач негладкої оптимізації, що виникають на етапі реалізації схем декомпозиції за змінними та обмеженнями, для розв'язання розглянутих задач оптимального проектування надійних мереж.

У першій модифікації (монотонній модифікації *r*-алгоритму) довжина кроку в напрямі антисубградієнта в перетвореному просторі змінних вибирається з умови мінімуму функції в напрямі руху. Це дає можливість досить точно знаходити точку оптимуму негладкої функції, але може привести до значної кількості обчислень значень цієї функції та її субградієнта. У другій модифікації (модифікації *r*-алгоритму з адаптивним

регулюванням кроку) запропоновано адаптивний спосіб регулювання крокового множника, який дозволяє досягти точки мінімуму негладкої опуклої функції за порівняно невелику кількість обчислень значення функції та її субградієнта.

На початку підрозділу подано ідею та загальну схему *r*-алгоритму (п. 1.1.1 і 1.1.2), далі описано способи вибору параметрів регулювання кроку для монотонної модифікації *r*-алгоритму (п. 1.1.3) та модифікації *r*-алгоритму з адаптивним регулюванням кроку (п. 1.1.4). У п. 1.1.5 подано результати тестових експериментів для розроблених програмних реалізацій обох модифікацій *r*-алгоритму.

1.1.1. Розтяг простору в *r*-алгоритмі

Розтяг простору в напрямі, заданому вектором $\xi \in E^n$, $\|\xi\|=1$, з коефіцієнтом α задається оператором

$$R_\alpha(\xi) = I_n + (\alpha - 1)\xi\xi^T,$$

де $\alpha \geq 1$ – фіксована константа; I_n – одинична $n \times n$ -матриця; $(\cdot)^T$ позначає транспонування (так що $\xi\xi^T$ – це $n \times n$ -матриця).

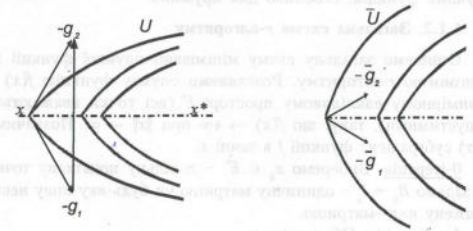


Рис. 1.1. Розтяг простору

Для того щоб проілюструвати ідею розтягу простору, розглянемо кусочно-гладку опуклу функцію f в E^2 та її криву

рівня U в точці $x \in E^2$, де f не є гладкою (див. рис. 1.1, *а*). Напрями найшвидшого спуску функції f в точці x зображено на рис. 1.1, *а* за допомогою антисубградієнтів $-g_1$, $-g_2$ для обох областей гладкості, які відповідають нижній та верхній напівплощинам. Зауважимо, що обидва антисубградієнта спрямовані за межі області зменшення значень функції, обмеженої кривою рівня U. Це означає зростання f замість очікуваного зменшення.

Розтягнемо простір у напрямі різниці двох субградієнтів $g_2 - g_1$ (рис. 1.1, *b*). У трансформованому просторі антисубградієнти $-g_1$, $-g_2$ та направлені в область зменшення функції, обмеженої кривою рівня $\overline{U}$. Таким чином, використання оператору розтягу простору в напрямі різниці двох послідовних субградієнтів (другий з яких знаходиться за допомогою процедури найшвидшого спуску в напрямі першого антисубградієнта) суттєво прискорює збіжність субградієнтного процесу при мінімізації яружних опуклих функцій, тому що розтяг простору на поточній ітерації зменшує кут між субградієнтом та напрямом на точку оптимуму. Це підтверджується швидкою збіжністю r-алгоритму як для гладких, так і для негладких опуклих функцій, особливо для яружних.

1.1.2. Загальна схема r-алгоритму

Опишемо загальну схему мінімізації опуклої функції за допомогою r-алгоритму. Розглянемо опуклу функцію $f(x)$ в n-вимірному евклідовому просторі E^n (всі точки вважаються допустимими), таку, що $f(x) \to +\infty$ при $\|x\| \to \infty$. Позначимо $g_f(x)$ субградієнт функції f в точці x.

0-ітерація. Виберемо $x_0 \in E^n$ – довільну початкову точку та задамо $B_0 = I_n$ – одиничну матрицю чи будь-яку іншу невироджену $n \times n$ -матрицю.

1-а ітерація. Обчислюємо:

$$g_0 = g_f(x_0) \text{ (субградієнт функції } f \text{ в точці } x_0),$$

$$\eta_0 = B_0 \frac{B_0^T g_0}{\left\| B_0^T g_0 \right\|} \text{ (початковий напрям пошуку)},$$

$$x_1 = x_0 - h_0 \eta_0 \quad (h_0 > 0,\ h_0 - \text{довжина кроку}).$$

Запам'ятовуємо x_1, $g_f(x_0)$, B_0. Після k ітерацій маємо: $x_k \in E^n$, $g_f(x_{k-1}) \in E^n$, B_{k-1} – матриця розміру $n \times n$.

$(k+1)$-а ітерація. Обчислюємо

$$g_k = g_f(x_k),$$

$$d_k = g_f(x_k) - g_f(x_{k-1}),$$

$$r_k = B_k^T d_k,$$

$$\xi_k = \frac{r_k}{\|r_k\|},$$

$$B_k = B_{k-1} R_{\beta_k}(\xi_k),\ \beta_k = 1/\alpha_k,\ \alpha_k > 1,$$

$$\eta_k = B_k \frac{B_k^T g_f(x_k)}{\left\| B_k^T g_f(x_k) \right\|} \text{ (напрям пошуку)},$$

$$x_{k+1} = x_k - h_k \eta_k,\ h_k > 0.$$

Запам'ятовуємо x_{k+1}, $g_f(x_k)$, B_k. Якщо виконується критерій зупинки, процедура завершується, в протилежному випадку – переходимо до наступної ітерації.

Як правило, використовуються такі критерії зупинки:

1) за аргументом, тобто завершуємо обчислення, якщо $\|x_{k+1} - x_k\| \le \varepsilon_x$, де $\varepsilon_x > 0$ – достатньо мале число;

2) за градієнтом, тобто завершуємо обчислення, якщо $\|g_f(x_{k+1})\| \le \varepsilon_g$, де $\varepsilon_g > 0$ – достатньо мале число.

Можна використовувати й інші критерії, наприклад значення функції, що мінімізується, кількість ітерацій тощо.

Зауваження. Позначимо $A_{k-1} = B_{k-1}^{-1}$ матрицю трансформування початкового простору E^n після k ітерацій (тобто після k-1 розтягів). Трансформований початковий простір позначимо $Y_k = A_{k-1} E^n$. Функцію f в змінних простору Y_k позна-

чимо $\varphi_k(y) = f(B_{k-1}y) = f(x)$ для всіх $y = A_{k-1}x$, $x \in E^n$. Тоді субградієнт функції $\varphi_k(y)$ в точці $y = A_{k-1}x$ визначається за формулою $g_{\varphi_k}(y) = B_{k-1}^T g_f(x)$ [45].

Таким чином, різниця між субградієнтами на (k–1)-й та k-й ітераціях, розглянута в Y_{k-1}, дорівнює r_k, де ξ_k – нормований напрям цієї різниці. На k-й ітерації Y_{k-1} розтягається вздовж цього напряму. Нове трансформування початкового простору E^n задається такими прямою та оберненою матрицями відповідно:

$$A_k = R_{\alpha_k}(\xi_k)A_{k-1},$$

$$B_k = B_{k-1}R_{\beta_k}(\xi_k),\ \beta_k = 1/\alpha_k.$$

Тоді в трансформованому просторі $Y_{k-1}=A_kE^n$ наступна ітерація, яка визначається функцією

$$\varphi_{k+1}(y) = f(B_k y) = f(x),$$

має вигляд

$$y_{k+1} = y_k - h_k \frac{B_k^T g_f(x_k)}{\left\| B_k^T g_f(x_k) \right\|} = y_k - h_k \frac{g_{\varphi_{k+1}}(y_k)}{\left\| g_{\varphi_{k+1}}(y_k) \right\|},$$

де $g_{\varphi_{k+1}}(y_k)$ – субградієнт функції φ_{k+1} в точці y_k.

Застосуємо лінійне перетворення B_k до наведеної вище формули й одержимо наступну ітерацію в початковому просторі:

$$x_{k+1} = x_k - h_k B_k \frac{B_k^T g_f(x_k)}{\left\| B_k^T g_f(x_k) \right\|}.$$

Сімейство r-алгоритмів характеризується двома послідовностями параметрів: це величини, обернені до коефіцієнтів розтягу $\beta_k=1/\alpha_k$, k = 1, 2, ..., та довжини кроків h_k, k = 1, 2, ... Природно, що ми повинні піклуватися про раціональний вибір цих параметрів, щоб одержати "хорошу" збіжність до оптимальної точки.

Для мінімізації негладкої опуклої функцій в E^n рекомендуємо вибирати значення $\alpha_k = \alpha \in [2; 4]$ для всіх k = 1, 2, ... Такий вибір коефіцієнта розтягу простору буде використовуватись як для монотонної модифікації r-алгоритму, так і для модифікації r-алгоритму з адаптивним регулюванням кроку. Відрізнятися ці модифікації будуть способом регулювання крокового множника в напрямі руху (напрямі антисубградієнта в перетвореному просторі змінних). Монотонна модифікація r-алгоритму пов'язана з регулюванням крокового множника h_k з умови мінімуму функції в напрямі руху. Модифікація r-алгоритму з адаптивним регулюванням кроку пов'язана з "грубим" способом регулювання крокового множника в напрямі руху, мета якого полягає в тому, щоб досягти точки мінімуму негладкої опуклої функції за порівняно невелику кількість обчислень значення функції та її субградієнта.

1.1.3. Монотонна модифікація r-алгоритму

У загальному випадку монотонна модифікація r-алгоритму складається з послідовності ітерацій, на кожній з яких здійснюються наступні дії (далі через x_0, x_1, ..., x_k, ... позначено послідовність точок простору E^n, що мінімізують функцію $f(x)$). Нехай зроблено $\bar{k}$ ітерацій ($\bar{k} > 1$) і, як результат, отримано точку $x_{\bar{k}} \in E^n$, обернену матрицю $B_{\bar{k}}$ перетворення простору відповідно до r-алгоритму і субградієнт $g(x_{\bar{k}})$. Отже, напрям руху з точки $x_{\bar{k}}$ визначається за формулою

$$x_{\bar{k}+1}(h_k) = x_{\bar{k}} - h_{\bar{k}}^{\min} B_{\bar{k}} B_{\bar{k}}^T g_{\bar{k}},$$

де $h_{\bar{k}}^{\min}$ – кроковий множник, що визначається з умови мінімуму за напрямом, тобто

$$h_{\bar{k}}^{\min} = \arg\min f(x_{\bar{k}}(h_{\bar{k}})).$$

Зокрема, $h_{\bar{k}}^{\min}$ може дорівнювати 0, і в цьому випадку $x_{\bar{k}+1} = x_{\bar{k}}$.

В обох випадках серед субградієнтів функції f в точці $x_{\bar{k}+1}$ знаходимо субградієнт, що утворює з напрямом руху кут, який більше або дорівнює $\pi/2$. Якщо шуканий субградієнт можна

вибрати різними способами, то вибирається той, проекція якого на напрям руху мінімальна.

Послідовність $f(x_0), f(x_1), ..., f(x_k)$ є незростаючою, тому що наступна точка отримується з попередньої в результаті операції знаходження мінімуму за напрямом. Звідси ясно, що послідовність значень функції $\{f(x_k)\}_{k=0}^{\infty}$ має границю $f^* \geq f(x^*)$. Обгрунтування запропонованого алгоритму зводиться до доказу того, що при певних обмеженнях на коефіцієнти розтягу простору аргументів $\{\alpha_k\}_{k=0}^{\infty}$ і при ігноруванні помилок округлення при обчисленнях маємо $f^* = f(x^*)$.

При роботі із сталим коефіцієнтом розтягу простору $\alpha > 1$ після кожної ітерації визначник матриці B_k зменшується в α раз, а матриці $H_k = B_k B_k^*$ – в α^2 раз.

При мінімізації так званих істотно яружних функцій, як правило, обумовленість матриць H_k, тобто відношення мінімального власного числа до максимального, прямує до нуля, при цьому помилки округлення починають істотно впливати на вибір напряму спуску, що може значно зменшити швидкість збіжності алгоритму до мінімального значення функції. Тому необхідно застосовувати процедури, що контролюють "близькість" матриць до виродження, і якщо це виявлено, то необхідно "відновити" B_k, замінивши її одиничною матрицею. Крім того, має сенс використовувати спеціальні програми, що дають оцінки знизу для функції, яка мінімізується, з метою оцінки точності визначення мінімуму в поточній точці $\bar{x}$ (для своєчасної зупинки процесу обчислення мінімуму).

Таким чином, реалізація монотонного r-алгоритму в загальному випадку складається із стадій, кожна з яких має такі блоки:

- обчислення послідовних значень $x_k, x_{k+1}, ...$, що відповідають незростаючим значенням функції, яка мінімізується, при сталому $\alpha > 1$ і відповідних матрицях B_k, B_{k+1};
- контроль виродженості матриць B_k та їх "відновлення" в разі потреби;
- розв'язання спеціальних квадратичних задач для одержання оцінок знизу функції, яка мінімізується.

1.1.4. Адаптивне регулювання довжини кроку

Для адаптивного регулювання довжини кроку пропонується така послідовність дій.

1. Фіксуються:

h_0^0 (початковий кроковий множник на 0-ітерації);

$n_h > 1$ (максимальна кількість спроб для збільшення довжини кроку);

$q_1 < 1$ (коефіцієнт зменшення довжини кроку в послідовних спробах; для негладких цільових функцій найкращі результати одержано при $q_1 = 1$, а для гладких – при $q_1 < 1$);

$q_2 > 1$ (коефіцієнт збільшення довжини кроку при послідовних спробах).

2. Позначимо h_k^0 початкову довжину кроку на k-ітерації.

3. На $(k+1)$-ітерації вибираємо напрям спуску згідно з r-алгоритмом. Робимо в цьому напрямі кроки h_k^0 до тих пір, поки в розтягнутому просторі напрям спуску становить гострий кут з якимсь із субградієнтів.

Якщо зроблено n_h кроків, продовжуємо спуск у тому ж напрямі, але зі збільшеною довжиною кроків $h_k^1 = q_2 h_k^0$. Якщо після n_h кроків умови зупинки не виконано, знову збільшуємо довжину кроку: $h_k^2 = q_2 h_k^1$, і так далі. Оскільки вважається, що функція f є опуклою і $\lim_{\|x\|\to\infty} f(x) = +\infty$, то зупинимося після скінченної кількості ітерацій.

Якщо зупинка сталася відразу після першої ітерації, слід зменшити довжину кроку: $h_{k+1}^0 = q_1 h_k^0$, та скінчити спуск вздовж цього напряму.

4. Довжина кроку $h_{k+1}^{p_k} = q_1^{p_k} h_1^0$, яка одержана на k-ітерації після p_k проб, використовується як початкова довжина кроку на $(k+1)$-ітерації.

Можна сподіватися на швидшу збіжність при застосуванні r-алгоритму для гладких задач оптимізації. В цих випадках можна брати більші значення α_k в комбінації із залученням більш точних методів знаходження мінімуму вздовж напряму спуску. Обчислювальна ефективність r-алгоритму залежить від

коефіцієнта розтягу простору та параметрів адаптивного регулювання кроку (див. також [47]). Для негладких функцій рекомендуємо такі значення параметрів: $\alpha \in [2; 4]$; $h_0 = 1{,}0$; $q_1 = 1{,}0$; $q_2 \in [1{,}1; 1{,}2]$; $n_h = [2; 3]$.

Якщо вдається оцінити відстань між початковою точкою x_0 та точкою мінімуму x^*, то початковий кроковий множник h_0 може відповідати умові

$$h_0 \approx \|x_0 - x^*\|.$$

Для гладких функцій рекомендовані ті самі параметри, лише $q_1 \in [0{,}8; 1{,}2]$. Це пояснюється тим фактом, що менша довжина кроку зумовлює точніше знаходження мінімуму функції, що, в свою чергу, означає швидшу збіжність для гладких функцій.

При використанні рекомендованих параметрів r-алгоритм звичайно дає такі результати:

- середня кількість кроків у напряму спуску на одній ітерації не перевищує 2–3;
- для поточної k-ітерації за n додаткових ітерацій (n — вимірність простору) значення функції f задовольняє нерівності

$$\frac{1}{5} \le \frac{f(x_{k+n}) - f^*}{f(x_k) - f^*} \le \frac{1}{2},$$

тобто воно стає у 2–5 разів ближче до оптимального значення f^*.

При мінімізації опуклої функції (навіть суттєво яружної) з критерієм зупинки, що випливає з нерівності $\|x_{k+1} - x_k\| \le \varepsilon_x$, $\varepsilon_x \in [10^{-6}; 10^{-5}]$ можна досягти точки x_k^*, для якої величина $\left|f(x_k^*) - f(x^*)\right| / \left|f(x^*) + 1\right|$ буде ~10^{-6}–10^{-5} для негладких функцій і ~10^{-12}–10^{-10} для гладких функцій. Ці експериментально одержані оцінки підтверджуються великою кількістю тестових і прикладних обчислень.

1.1.5. Програмна реалізація модифікацій r-алгоритму й обчислювальні експерименти

Для оцінки ефективності розроблених модифікацій r-алгоритму при розв'язанні різних класів задач негладкої опуклої оптимізації були розроблені різні версії дослідницьких програм і проведені обчислювальні експерименти.

Основними блоками дослідницьких програм є:

- обчислення значень опуклої функції $f(x)$, що мінімізується, і її субградієнтів у визначених точках;
- обчислення нових значень результуючої оберненої матриці перетворення B_k на кожній ітерації;
- обчислення параметрів напряму руху і пошук мінімуму за напрямком в основному просторі;
- підготовка даних і розв'язання спеціальної задачі квадратичного програмування для знаходження напряму ε-субградієнтного спуску.

При розробці дослідницької програми для монотонної модифікації r-алгоритму (мова програмування C++) використовувались всі чотири блоки, а при розробці програм для модифікації r-алгоритму з адаптивним регулюванням кроку (мови програмування C++ та FORTRAN) використовувались лише перші три блоки. Дослідницькі програми можуть працювати в різних режимах, що контролюються рядом параметрів, настроювання яких здійснюється перед початком запуску програми.

При реалізації монотонної модифікації r-алгоритму для методу пошуку мінімуму за напрямом використовувався алгоритм дихотомії.

Нижче наводяться результати обчислювальних експериментів розв'язання задач за допомогою обох модифікацій r-алгоритму. Експерименти проводилися на персональному комп'ютері IBM PC/Pentium III/ 750МГц/256Мгб/Windows98.

Спочатку наведемо результати розв'язання таких спеціальних квадратичних задач:

знайти

$$\min_{\{\alpha\}} \left\| \sum_{i=1}^{m} \alpha_i g_i \right\|^2, \quad \alpha = (\alpha_1, \ldots, \alpha_m), \quad g_i \in E^n \qquad (1.1)$$

при обмеженнях

$$\alpha_i \geq 0 \qquad \forall i, \qquad (1.2)$$

$$\sum_{i=1}^{m} \alpha_i = 1 . \qquad (1.3)$$

Таблиця 1.1. Порівняльні результати часу розв'язання задач (1.1)–(1.3) різними алгоритмами

Задача	r-алгоритм з адаптивним регулюванням кроку	Монотонний r-алгоритм	LOQO
(n, m)	t (с)	t (с)	t (с)
(14, 28)	0,01	0,05	0,33
(16, 32)	0,01	0,11	0,49
(20, 40)	0,06	0,22	0,54
(40, 80)	0,28	1,1	1,27
(80, 160)	2,47	3,7	3,79
(100, 200)	3,79	5,9	6,11

У табл. 1.1 наведені порівняльні результати часу розв'язання задач (1.1)–(1.3) обома модифікаціями r-алгоритму і програмою LOQO [156].

Розмірності задач наведені в першому стовпці таблиці. Вихідні дані для тестових задач генерувалася за допомогою пакета AMPL [88]. Значення параметрів для монотонного r-алгоритму були такими: коефіцієнт розтягу простору α вибирався рівним 2, штрафні коефіцієнти дорівнювали 10000, умовою припинення обчислень було виконання нерівності $\| B_{k-1}^T g_f(x^k) \| \leq \text{eps}$, де $\text{eps} \approx 10^{-6}$. Значення параметрів модифіка-

ції r-алгоритму з адаптивним регулюванням кроку вибиралися такими: $\alpha = 2{,}0$; $h_0 = 1{,}0$; $q_1 = 1{,}0$; $q_2 = 1{,}1$; $n_h = 3$; $\varepsilon_x = 10^{-6}$; $\varepsilon_g = 10^{-6}$.

Далі наведені результати, що стосуються задачі про максимальний розріз графа для випадку тріангуляції, що відповідає ікосаедру (див. рис. 1.2). Вказана задача зводилась до задачі мінімізації такої негладкої опуклої функції від змінних y_{ij}, кожна з яких відповідає ребру ікосаедра:

$$\Phi_{\lambda,\mu}(y) = -\sum_{(i,j)\in E} c_{ij} y_{ij} + \sum_{(i,j)\in E} \lambda_k \{\max\{0, y_{ij}^2 - y_{ij}\}\} + \\ + \sum \mu_T \max\{0, y_{ij} + y_{ik} + y_{kj} - 2\}. \qquad (1.4)$$

Тут μ_T – штрафний коефіцієнт, який відповідає обмеженню $y_{ij} + y_{ik} + y_{kj} - 2 \leq 0$ для кожного трикутника $T \leq \{i, j, k\}$. Значення параметрів для кожного ребра показані на рис. 1.2.

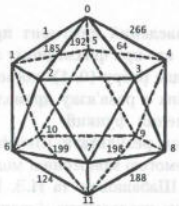


Рис. 1.2. Приклад тріангуляції, що відповідає ікосаедру

Мінімізація функції (1.4) проводилась за допомогою монотонної модифікації r-алгоритму, коли значення штрафних коефіцієнтів λ_k, μ_T дорівнювали 2000. У табл. 1.2 наведений фрагмент процесу мінімізації цієї функції.

Таблиця 1.2. Результати обчислень для задачі про максимальний розріз графа

Ітерація	f	$\|x_k - x_{k-1}\|$
0	-1	1.73222
100	-3446,9588208	0,000371058
200	-347706612665	$8{,}59048 \cdot 10^{-6}$
300	-3477,97230801	$1{,}64329 \cdot 10^{-6}$
400	-3477,99847902	$2{,}76847 \cdot 10^{-7}$
500	-3477,99994801	$2{,}49984 \cdot 10^{-8}$
600	-3477,99999793	$2{,}59161 \cdot 10^{-10}$
700	-3477,99999993	$4{,}48987 \cdot 10^{-12}$
800	-3478	$9{,}07441 \cdot 10^{-12}$

У табл. 1.3 наведений фрагмент процесу розв'язування попередньої задачі, але при фіксованому значенні 1 для змінної $x(0, 1)$, що відповідає ребру (0, 1) ікосаедра. Цей приклад показує, що мала зміна в розв'язку приводить до істотної зміни оптимального значення функції.

Далі наводяться результати мінімізації відомої функції Шабашової за допомогою монотонної модифікації r-алгоритму. В праці [48] Л.П. Шабашовою та Н.З. Шором була запропонована така кусочно-квадратична функція. Треба знайти

$$\min_{x \in E^n} f(x),$$

де $f(x) = \max_{i=1,m} \varphi_i(x)$; $\varphi_i(x) = a_i \sum_{j=1}^{n} (x^{(j)} - a_{ij})^2$; $\{a_i\}$ – m-вимірний вектор; $x = \{x^{(1)}, ..., x^{(n)}\}$; $\{a_{ij}\}$ – матриця розміром m × n. У прикладі використовувалися такі значення параметрів: $n = 5$, $m = 10$, $\{a_i\} = \{1; 5; 10; 2; 4; 3; 1{,}7; 2{,}5; 6; 3{,}5\}$. Величини $\{a_{ij}\}$ задаються матрицею (в транспонованому вигляді)

$$\begin{bmatrix} 0 & 2 & 1 & 1 & 3 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 2 & 4 & 2 & 2 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 \\ 0 & 1 & 1 & 2 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 3 & 2 & 2 & 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Початковою точкою була взята точка $x_0 = (0, 0, 0, 0, 1)$. Точне значення x^*, при якому досягається мінімум, становить $x^* = (1{,}12434; 0{,}97945; 1{,}47770; 0{,}92023; 1{,}12429)$, а оптимальне значення функції $f(x^*) = 22{,}60016$.

Для наведених значень параметрів ця функція має складну яружну структуру, так що її чисельна мінімізація викликає труднощі для багатьох алгоритмів.

В табл. 1.4 наведено фрагмент процесу мінімізації функції Шабашової монотонним r-алгоритмом. Відзначимо, що на 6-й ітерації $h_k^{\min} \approx 0$, що характеризує той факт, що напрям антисубградієнта не дає зменшення функції. Для негладких

Таблиця 1.3 Фрагмент процесу розв'язування задачі про максимальний розріз графа при фіксованому значенні 1 для змінної $x(0, 1)$, що відповідає ребру (0, 1) ікосаедра

Ітерація	f	$\|x_k - x_{k-1}\|$
0	-1	1,73222
100	-3352,72739482	0,0435266
200	-3378,5249768	0,000168383
300	-3378,98617354	$2{,}99322 \cdot 10^{-7}$
400	-3378,99948758	$6{,}19661 \cdot 10^{-7}$
500	-3378,99998512	$2{,}84686 \cdot 10^{-9}$
600	-3378,99999939	$6{,}25072 \cdot 10^{-11}$
700	-3378,99999999	$1{,}52584 \cdot 10^{-12}$
800	-3379	$3{,}54075 \cdot 10^{-15}$

функцій саме така ситуація затрудняє регулювання кроку з умови мінімуму функції.

Таблиця 1.4 Результати мінімізації функції Шабашової

Ітерація	f	$\|x_k - x_{k-1}\|$
0	10350110	1,73222
1	36,4316909873	1018,66
2	35,4183668678	0,410959
3	24,8335672661	0,835904
4	24,4928006163	0,0289056
5	24,3124641069	0,125827
6	24,3124641071	$5{,}19203 \cdot 10^{-10}$
7	22,9025749786	0,256012
...	...	...
90	22,6001620958	$1{,}14558 \cdot 10^{-11}$

І, нарешті, в табл. 1.5 наведені порівняльні результати часу розв'язання наведених вище задач обома модифікаціями r-алгоритму.

Таблиця 1.5 Порівняльні результати часу розв'язання обома модифікаціями r-алгоритму.

Задача	r-алгоритм з адаптивним регулюванням кроку	Монотонний r-алгоритм
	t (с)	t (с)
Max-cut (Ікосаедр)	0,02	0,09
Функція Шабашової	0,01	0,01

Результати експериментів показують, що розроблені модифікації r-алгоритмів є досить ефективними саме для класу

задач мінімізації негладких опуклих функцій. Час, який затрачено на пошук оптимальних значень для 100 змінних, вимірюється секундами, що дає можливість при використанні схем декомпозиції для задач проектування надійних мереж розв'язувати координуючі задачі негладкої оптимізації за час порядку кількох секунд.

1.2. Про один алгоритм ε-субградієнтної мінімізації

Розглядається задача мінімізації обмеженої знизу опуклої функції $f(x)$ в n-вимірному евклідовому просторі R^n:

$$f^* = \min\{f(x) \mid x \in R^n\}.$$

Для заданого числа $\bar{\varepsilon} \geq 0$ введемо $\bar{\varepsilon}$-оптимальну множину $X^*(\bar{\varepsilon})$

$$X^*(\bar{\varepsilon}) = \{x \in R^n \mid f(x) \leq f^* + \bar{\varepsilon}\}. \qquad (1.5)$$

Довільну точку $\tilde{x} \in X^*(\bar{\varepsilon})$ та відповідне значення функції $\tilde{f} = f(\tilde{x})$ будемо називати розв'язком задачі $\bar{\varepsilon}$-оптимізації ($\bar{\varepsilon}$-розв'язком). Задамо кулю $D(z, R)$ початкової локалізації $\bar{\varepsilon}$-розв'язку: $z \in R$ центр кулі, $R > 0$ її радіус. Припустимо, що куля $D(z, R)$ містить $X^*(\bar{\varepsilon})$ (припущення $X^*(\bar{\varepsilon}) \subset D(z, R)$ робиться для стислості викладок, хоча наведені далі твердження справедливі і за умови $D(z, R) \cap X^*(\bar{\varepsilon}) \neq \varnothing$).

Будемо використовувати трохи відмінне від класичного визначення ε-субградієнта [7]. Вектор $g \in R^n$ називається умовним $(\varepsilon, \tilde{f})$-субградієнтом функції $f(x)$ в точці z, якщо для $\forall x \in D(z, R)$ виконується нерівність

$$f(x) \geq \tilde{f} + (g, x - z) - \varepsilon, \qquad (1.6)$$

де $f(z) \geq \tilde{f} \geq f^*$; $\varepsilon \in R^1$. Значення $\tilde{f}$, одержане на деякій ітерації алгоритму розв'язання задачі $\bar{\varepsilon}$-оптимізації, звичайно дорівнює отриманому рекордному значенню функції $f(x)$. Якщо значення f^* відоме апріорі, то $\tilde{f} = f^*$. Звичайне класичне визначення ε-субградієнта [126] відповідає означенню умовного $(\varepsilon, \tilde{f})$-субградієнта (1.6) при $\tilde{f} = f(z)$; $\varepsilon \geq 0$.

Узагальнений градієнт функції $f(x)$ у точці z в класичному розумінні [45] є умовним $(0, f(z))$-субградієнтом.

Позначимо $G(\varepsilon, \tilde{f}, z)$, $\partial f(z)$ множини $(\varepsilon, \tilde{f})$-субградієнтів і узагальнених градієнтів функції $f(x)$ в точці z відповідно. Очевидно, $\partial f(z) = G(0, f(z), z)$ Вважаємо, що алгоритми обчислення $f(x)$ і $g(x) \in \partial f(z)$ у довільній точці відомі. Нехай

$g \in G(\varepsilon_1, \tilde{f}_1, z_1)$; $f^* \le \tilde{f}_2 \le f(z_2)$, тоді $g \in G(\varepsilon_2, \tilde{f}_2, z_2)$, де

$$\varepsilon_2 = \varepsilon_1 + \tilde{f}_2 - \tilde{f}_1 - (g, z_2 - z_1). \qquad (1.7)$$

Таким чином, $(\varepsilon, \tilde{f})$-субградієнт у деякій точці $z_1 \in (\varepsilon, \tilde{f})$-субградієнтом у будь-якій іншій точці z_2. Формула (1.7) визначає правило переобчислення параметрів $(\varepsilon, \tilde{f})$-субградієнта. Множина $G(\varepsilon, \tilde{f}, z)$ за своїми властивостями аналогічна множині $\partial f(z)$ (щодо опуклості, замкнутості тощо). Наприклад, $\{0 \in G(\varepsilon, \tilde{f}, z),\ \varepsilon \le \bar{\varepsilon}\} \Leftrightarrow \{\tilde{f} \le f^* + \bar{\varepsilon}\}$.

Нехай $\tilde{X}(\bar{\varepsilon}, \tilde{f}) = \{x \in R^n \mid f(x) \le \tilde{f} - \bar{\varepsilon}\}$.

Відзначимо, що:

$$\tilde{X}(\bar{\varepsilon}, \tilde{f}) = \varnothing \Rightarrow \{\tilde{f} \in \bar{\varepsilon}\text{-розв'язком}\};$$

$$\tilde{f} \ge f^* + 2\bar{\varepsilon} \Rightarrow X^*(\bar{\varepsilon}) \subset \tilde{X}(\bar{\varepsilon}, \tilde{f}).$$

Введемо позначення: $P(z, \eta) = \{x \in R^n \mid (x - z, \eta) = 0\}$ – площина, що проходить через точку z з нормаллю $\eta \in R^n$; $|\eta| = 1$; $P^+(z, \eta) = \{x \in R^n \mid (x - z, \eta) \ge 0\}$ – напівпростір, що відповідає площині $P(z, \eta)$. Припустимо, що

$$g \in G(\varepsilon, \tilde{f}, z);\ h = (\bar{\varepsilon} - \varepsilon) / |g|;\ \tilde{z} = z - hg / |g|. \qquad (1.8)$$

Тоді $\tilde{X}(\bar{\varepsilon}, \tilde{f}) \subset P^+(z - \tilde{z}, -g)$.

Твердження 1.1. *Для заданої точки z та напряму спуску η ($|\eta| = 1$) і $\bar{\varepsilon} > 0$ можна гарантувати обчислення такого ε- субградієнта $g \in G(\varepsilon, \tilde{f}, z)$, для якого $\varepsilon \le \bar{\varepsilon}$ і $(g, \eta) \ge 0$.*

Тут і далі ми не наводимо доведення тверджень. Вони досить прості, а деякі з них (для тверджень 1.2 і 1.3) є в літературі.

Два наступних твердження стосуються побудови описаних

еліпсоїдів мінімального об'єму навколо підмножин кулі, що мають просту структуру.

Додатково позначимо: $W = D(z, R) \cap P^+(\tilde{z}, \eta)$ –частина кулі $D(z, R)$, утворена січною площиною $P(\tilde{z}, R)$ де $\tilde{z} = z + h\eta$; $0 \le h < R$; $|\eta| = 1$; W^*– еліпсоїд мінімального об'єму з центром у точці $\tilde{z}$, який містить множину W; $\tilde{D}^*(z, \tilde{R})$ – куля мінімального об'єму, що містить еліпсоїд W^*; $V(K)$– об'єм тіла K; $R_\alpha(\eta) = (\alpha - 1)\eta^T\eta + 1$ оператор розтягу простору [45] в напрямі η з коефіцієнтом α.

Твердження 1.2. *Справедливі такі формули:*

$$\tilde{R} = \sqrt{R^2 - h^2}\ ;\ \tilde{D}^*(z, \tilde{R}) = \Omega W^* = R_{\alpha^*}(\eta) W^*,$$

де $\alpha^* = \sqrt{(R + h)/(R - h)}$;

$q = V(W^*)/V(D(z, \tilde{R})) = \sqrt{(R + h)/(R - h)}\,((R - h)/(R + h))^{n/2}$.

Нехай $W = D(z, R) \cap P^+(z, \eta_1) \cap P^+(z, \eta_2)$ – частина кулі $D(z, R)$, що утворена січними площинами $P(z, \eta_1)$, $P(z, \eta_2)$; W^* – еліпсоїд мінімального об'єму з центром у точці z, що містить множину W; $\tilde{D}^*(z, \tilde{R})$ – куля мінімального об'єму, що містить еліпсоїд W^*; $\eta^* = (\eta_2 - \eta_1)/|\eta_2 - \eta_1|$; $\xi^* = (\eta_2 + \eta_1)/|\eta_2 + \eta_1|$; $\{\bar{\eta}_i\}$ – ортонормований базис підпростору, ортогонального векторам η^*, ξ^*; $\cos(\varphi) = (\eta_1, \eta_2) \le 0$; $\psi = \pi - \varphi$ (ψ – кут між площинами $P(z, \eta_1)$ і $P(z, \eta_2)$).

Твердження 1.3. *Справедливі такі формули:*

$$\tilde{R} = \sqrt{2}\cos(\psi/2) R; \qquad (1.9)$$

$$\tilde{D}^*(z, \tilde{R}) = \Omega W^* = \left\{R_{\alpha^*}(\eta^*) \prod_{i=1}^{n-2} R_{\tilde{\alpha}}(\eta_i)\right\} W^* \qquad (1.10)$$

де

$$\alpha^* = \operatorname{ctg}(\psi/2),\ \tilde{\alpha} = \sqrt{2\cos(\psi/2)}, \qquad (1.11)$$

$$q = V(W^*)/V(D(z, R)) = \sin(\psi). \qquad (1.12)$$

Відзначимо, що оператор перетворення Ω з формули (1.10) можна подати у компактнішому вигляді, а саме

$$\Omega = \tilde{\alpha} R_{\alpha^*/\tilde{\alpha}}(\eta^*) R_{1/\tilde{\alpha}}(\xi^*).$$

З твердження 1.2 випливає, що об'єм еліпсоїда W^* менше об'єму кулі $\tilde{D}^*(z, \tilde{R})$ тільки для $\psi < \pi/2$ ($\varphi > \pi/2$).

Нехай задане число $\delta > 0$ Наведемо ітеративний алгоритм генерації (принаймні) двох площин $P(z, \eta_1)$, $P(z, \eta_2)$, що $(\eta_1, \eta_2) = \cos(\varphi) \le -1 + \delta$.

Позначимо $E = \{e_1, e_2, ..., e_m\}$ множину векторів з R^n, $\mathrm{Nr}\{E\}$ – вектор з опуклої оболонки множини E, що має мінімальну довжину.

Перша ітерація.

1. Обчислюємо $g_1 \in \partial f(z)$. Якщо $g_1 = 0$, то робота завершується (задача мінімізації $f(x)$ розв'язана), інакше покладемо $e_1 = -g_1/|g_1|$.

2. Використовуючи процедуру одновимірної мінімізації вздовж напряму e_1, обчислюємо $\bar{\varepsilon}$-субградієнт $g_2 \in G(\varepsilon, \tilde{f}, z)$, для якого $\varepsilon \le \bar{\varepsilon}$, $(g_2, e_1) \ge 0$ (див. твердження 1.1). Якщо $g_2 = 0$ то робота завершується (задача $\bar{\varepsilon}$-мінімізації $f(x)$ розв'язана).

3. Нехай $G_1 = \{g_1, g_2\}$; $e_2 = -g_2/|g_2|$; $E_1 = \{e_1, e_2\}$; $p_1 = \mathrm{Nr}\{E_1\} = 1/2(e_1 + e_2)$; $\eta_1 = e_1$; $\eta_2 = e_2$; $\cos(\varphi) = (\eta_1, \eta_2) \le 0$.

Якщо $p_1 = 0$, то робота завершується (задача $\bar{\varepsilon}$-мінімізації $f(x)$ розв'язана).

Якщо $\cos(\varphi_1) \le -1 + \delta$, то робота завершується, інакше – перехід на наступну ітерацію.

Нехай на ітерації з номером k ($k = 1, 2, ...$) отримано множину E_k ($m_k = |E_k|$; $2 \le m_k \le n$) і вектор $p_k = \mathrm{Nr}\{E_k\} \ne 0$ (для скорочення запису індекс k в позначенні m_k будемо опускати).

Ітерація з номером $(k + 1)$.

1. Використовуючи процедуру одновимірної мінімізації вздовж напряму p_k, обчислюємо ε-субградієнт $g_{m+1} \in G(\varepsilon, \tilde{f}, z)$, для якого $\varepsilon \le \bar{\varepsilon}$, $(g_{m+1}, e_1) \ge 0$ (див. твердження 1.1). Якщо $g_{m+1} = 0$ то робота завершується (задача $\bar{\varepsilon}$-мінімізації $f(x)$ розв'язана). Відзначимо, що в результаті одновимірної мінімізації $\tilde{f}$ значення може зменшитися. У цьому випадку переобчислюють $(\varepsilon, \tilde{f})$ параметрів векторів із множини G_k згідно з (1.5).

2. Нехай $\tilde{G}_{k+1} = G_k \cup g_{m+1}$; $e_{m+1} = -g_{m+1}/|g_{m+1}|$; $\tilde{E}_{k+1} = E_k \cup e_{m+1}$; $p_{k+1} = \mathrm{Nr}\{\tilde{E}_{k+1}\} = \sum_{i \in I} \lambda_i e_i$, де I – множина індексів i, для яких $\lambda_i > 0$.

Якщо $p_{k+1} = 0$ то робота завершується (задача мінімізації $f(x)$ розв'язана). Зауважимо, що $\lambda_{m+1} > 0$ (внаслідок $(g, \eta) \ge 0$), і тому $n \ge |I| \ge 2$, якщо $p_k \ne 0$.

3. Нехай $G_{k+1} = \{g_i \in \tilde{G}_{k+1} \mid \lambda_i > 0\}$; $E_{k+1} = \{E_i \in \tilde{E}_{k+1} \mid \lambda_i > 0\}$ (E_{k+1} – множина "активних" векторів з $\tilde{E}_{k+1}$ відносно операції $\mathrm{Nr}\{\}$, $n \ge |E_{k+1}| \ge 2$). Перенумеруємо вектори множини $E_{k+1} = \{e_i, i = 1, 2, ..., m\}$ таким чином, щоб $\lambda_1 \ge \lambda_2 \ge ... \lambda_m > 0$ (вектори G_{k+1} перенумеровуються відповідно до E_{k+1}).

4. Нехай $\eta_1 = e_1$, $\eta_2 = \xi/|\xi|$, де $\xi = \sum_{i=2}^{m} \frac{\lambda_i}{1 - \lambda_1} e_i$; $\cos(\varphi_{k+1}) = (\eta_1, \eta_2)$.

Якщо $\cos(\varphi_{k+1}) \le -1 + \delta$, то робота завершується, інакше переходимо до наступної ітерації.

Твердження 1.4. *Алгоритм 1–4 є скінченним. При завершенні роботи алгоритму виконується одна з таких умов: а) задача $\bar{\varepsilon}$-мінімізації $f(x)$ розв'язана; б) отримані такі площини $P(z, \eta_1)$, $P(z, \eta_2)$ що $(\eta_1, \eta_2) = \cos(\varphi_{k+1}) \le -1 + \delta$.*

Справедливі такі оцінки значень для $|p_{k+1}|$ та $\sin(\psi_{k+1}) = \sin(\pi - \varphi_{k+1})$:

$$|p_{k+1}| \le 1/\sqrt{2 + k}, \qquad (1.13)$$

$$\sin^2(\psi_{k+1}) \le \frac{p_{k+1}^2(1 - p_{k+1}^2)}{p_{k+1}^2(1 - 2/n) + (1/n)^2} = \Phi(p_{k+1}^2). \qquad (1.14)$$

Можна показати, що функція $\Phi(p_{k+1}^2)$ правої частини (1.14) монотонно зростає для $p_{k+1}^2 \in [0, 1/n]$ причому $\Phi(0) = 0$, $\Phi(1/n) = 1$. Тому оцінка (1.14) є змістовною лише при $p_{k+1}^2 < 1/n$. Наприклад, $\Phi(1/n^2) = \frac{1}{2}(1 + 1/n)$. Тому, якщо $p_{k+1}^2 \le 1/n^2$ то $\sin(\psi_{k+1}) \le \frac{\sqrt{2}}{2}(1 + 1/n)$, ($\sim \frac{\sqrt{2}}{2}$ для $n \gg 2$). Відзначимо, що при цьому, відповідно до (1.11), (1.12), $q = \sin(\psi) \approx \sqrt{2}/2 \approx 0,7$;

$\alpha^* = \operatorname{ctg}(\psi/2) \approx 1+\sqrt{2} \approx 2{,}4$; $\tilde{\alpha} = \sqrt{2}\cos(\psi/2) \approx \sqrt{1+\sqrt{2}/2} \approx 1{,}3$.

В основі описаного нижче ε-субградієнтного алгоритму мінімізації лежить досить проста ідея. Задано число q, $0 < q < 1$. Нехай на ітерації k отримано: точку x_k; A_k – невироджене лінійне перетворення ($B_k = A_k^{-1}$); $\tilde{f}_k$ – рекордне значення функції $f(x)$; R_k – параметр еліпсоїда локалізації E_k множини $\tilde{X}(\bar{\varepsilon}, \tilde{f}_k)$:

$$E_k = \{x \in R^n \mid (A_k(x - x_k), A_k(x - x_k)) \le R_k\}; \ \tilde{X}(\bar{\varepsilon}, \tilde{f}_k) \subset E_k.$$

Ітерація з номером $k+1$ виконується в перетвореному просторі $Y_k = A_k X$, де X – початковий простір змінних. Образом еліпсоїда E_k у просторі Y_k є куля $D(y_k, R_k)$ де $y_k = A_k x_k$. На ітерації з номером $k+1$ встановлюється факт розв'язання задачі $2\bar{\varepsilon}$-оптимізації або будується еліпсоїд $\tilde{E}_{k+1}$ локалізації множини $\tilde{Y}(\bar{\varepsilon}, \tilde{f}_{k+1})$:

$$\tilde{Y}(\bar{\varepsilon}, \tilde{f}_{k+1}) \subset \tilde{E}_{k+1} = \{y \in Y_k \mid (\Omega_{k+1}(y - y_{k+1}), \Omega_{k+1}(y - y_{k+1})) \le R_{k+1}\}.$$

При цьому $V(\tilde{E}_{k+1}) \le qV(D(y_k, R_k))$. Побудова цього еліпсоїда і визначення параметра $\tilde{f}_{k+1}$ проводиться з використанням процедур одновимірної мінімізації відповідно до алгоритму 1–4 з твердження 1.4. При цьому на кожній ітерації цього алгоритму обчислюються значення коефіцієнтів q_1 і q_2 зменшення об'ємів еліпсоїдів локалізації множини $\tilde{X}(\bar{\varepsilon}, \tilde{f}_k)$ відповідно до тверджень 1.3 і 1.2. Нехай $q_{k+1} = \min\{q_1, q_2\}$. Якщо $q_{k+1} \le q$, то робота за алгоритмом 1–4 припиняється. Можливі такі два випадки.

1. $q_{k+1} = q_1$. У цьому випадку оператор Ω_{k+1} визначається оператором Ω з твердження 1.3. Значення параметра (кроково-го множника) h обчислюється за формулами (відповідно до (4) і з врахуванням перетворення простору):

$$h = (\bar{\varepsilon} - \varepsilon) / |B_k^* g(x_k)|,$$

де $g(x_k) \in \partial f(x_k)$; $\varepsilon = \tilde{f}_{k+1} - f(x_k)$. Якщо $h > R_k$, то робота зупиняється – задачу розв'язано. Інакше –

$$x_{k+1} = x_k - hB_k B_k^* g(x_k) / |B_k^* g(x_k)|.$$

2. $q_{k+1} = q_2 > q_1$. У цьому випадку Ω_{k+1} оператор визначається оператором Ω з твердження 1.2. Перехід у нову точку не відбувається: $x_{k+1} = x_k$.

Перетворення простору на ітерації $k+1$ визначається оператором Ω_{k+1}: $A_{k+1} = \Omega_{k+1} A_k$.

На підставі [8] можна довести таке твердження про ефективність наведеного алгоритму.

Твердження 1.5. *Для числа k ітерацій алгоритму, за які він забезпечує розв'язання задачі 2ε-оптимізації, справедлива така оцінка: $k \le n \dfrac{\ln(1/\gamma)}{\ln(1/q)}$, де γ – відносна точність розв'язання задачі, $\gamma = \bar{\varepsilon}/(RC)$; – С оцінка зверху норм субградієнтів у початковій кулі $D(z, R)$.*

Якісна інтерпретація описаного алгоритму, який будемо називати $r(\varepsilon)$-алгоритмом, полягає в наступному. Алгоритм належить до класу методів із перетворенням простору. На кожній ітерації алгоритму процес перетворення здійснюється при застосуванні операторів розтягу простору вздовж ортогональних напрямів. Параметри перетворення визначаються на основі побудови еліпсоїдів, в яких локалізовано $\bar{\varepsilon}$-розв'язок. Еліпсоїди локалізації будуються на основі інформації, одержаної в результаті застосування процедури одновимірної мінімізації вздовж вибраного напряму. Якщо в результаті одновимірної мінімізації відбувається істотне поліпшення рекордного значення функції, то перетворення простору можна розглядати як розтяг у напряму субградієнта. У протилежному випадку – як розтяг уздовж напрямку, ортогональному "ярові", що утворюють поверхні рівня функції. На кожній ітерації забезпечується зменшення об'єму області локалізації $\bar{\varepsilon}$-розв'язання принаймні в задане число раз (яке є одним із параметрів алгоритму). При оцінці ефективності алгоритму згідно з твердженням 1.5 необхідно враховувати також трудомісткість побудови множин локалізації $\bar{\varepsilon}$-розв'язку. Про цю трудомісткість можна судити за коментарями до твердження 1.4. Відповідно до них для забезпечення істотного зменшення об'єму області

локалізації (~0,7 раз) досить застосувати n^2 процедур одновимірної мінімізації. Таким чином, за вказаною оцінкою, для такого коефіцієнта зменшення об'єму області локалізації трудомісткість однієї ітерації алгоритму може бути досить великою. Однак ця оцінка досить груба. Крім того, відзначимо, що процес застосування алгоритму (на відміну від методу еліпсоїдів [50]) істотно залежить від конкретних характеристик функції, яка мінімізується.

Наведемо результати чисельного дослідження ефективності розробленого ε-субградієнтного алгоритму мінімізації з перетворенням простору – $r(\varepsilon)$-алгоритму.

В розробленій програмній реалізації $r(\varepsilon)$-алгоритму використовуються такі процедури: генерація множин ε-субградієнтів, побудова агрегованого ε-субградієнта, ортогонального перетворення простору змінних. Параметри ортогональних перетворень (коефіцієнти й напрями розтягу) визначаються на основі побудови множин локалізації ε-розв'язків. Алгоритм використовує процедуру одновимірного спуску та ε в певному розумінні монотонним. $r(\varepsilon)$-Алгоритм забезпечує розв'язання задачі мінімізації опуклих функцій із заданою точністю. Ця характеристика алгоритму особливо важлива при його використанні для розв'язання мережевих задач оптимізації. Програмна реалізація $r(\varepsilon)$-алгоритму оформлена як клас CRalg_Eps.

Як тестові задачі для дослідження чисельної ефективності класу CRalg_Eps розглядалися задачі мінімізації таких двох функцій:

$$f_1(x) = \sum_{i=1}^{n} \rho_n^{i-1} x_i^2,$$

$$f_2(x) = \sum_{i=1}^{n} \rho_n^{i-1} |x_i|,$$

де ρ_n параметр вибирався залежно від вимірності задачі n за формулою $\rho_n = 10^{6/(n-1)}$. При цьому $\rho_n^{n-1} = 10^6$. Початкова точка $x_i = 1{,}0$; $i = 1, 2, ..., n$. Параметр точності розв'язку за функціоналом становить $\bar{\varepsilon} = 10^{-6}$. Критерій зупинки: $f_k \le 10^{-6}$ де f_k – значення функції на ітерації зупинки k.

Результати розв'язування тестових задач мінімізації функцій $f_1(x), f_2(x)$ наведені в табл. 1.6 та 1.7 відповідно, де прийнято такі позначення: NVarbl – кількість змінних; QVolum – значення параметра q (коефіцієнт зменшення об'єму); nIter – кількість ітерацій; NLStep_Avrg – середнє число застосувань алгоритму одновимірної мінімізації на одній ітерації; Alph_Avrg – середнє значення коефіцієнта розтягу простору.

Таблиця 1.6. Результати розв'язування задачі мінімізації функції $f_1(x)$

NVarbl	QVolum	nIter	NLStep_Avrg	Alph_Avrg
5	0,7	36	2,25	3,624
10	0,99	107	1,364	1,661
10	0,7	56	3,214	3,035
20	0,99	195	1,297	1,621
20	0,7	86	3,686	2,724
30	0,99	283	1,254	1,606
30	0,7	109	3,872	2,848
40	0,99	360	1,236	1,611
40	0,7	134	4,127	2,761
50	0,99	435	1,205	1,6
50	0,7	153	4,255	2,759
100	0,99	711	1,136	1,592
100	0,7	243	4,407	2,744

Результати обчислювальних експериментів показують досить високу ефективність $r(\varepsilon)$-алгоритму і виявляють такі особливості. Середнє число застосування алгоритму одновимірної мінімізації на одній ітерації виявляється малим порівняно з

Таблиця 1.7. Результати розв'язування задачі мінімізації функції $f_2(x)$

NVarbl	QVolum	nIter	NLStep_Avrg	Alph_Avrg
5	0,99	142	1,204	2,524
5	0,7	67	2,179	4,748
10	0,99	413	1,165	1,855
10	0,7	133	3,015	3,38
20	0,99	1274	1,095	1,552
20	0,7	289	4,173	2,842
30	0,99	2164	1,081	1,52
30	0,7	445	5,231	2,69
40	0,99	1930	1,09	1,508
40	0,7	374	6,035	2,607
50	0,99	2594	1,076	1,465
50	0,7	455	6,868	2,601
100	0,9	4062	2,597	1,755
100	0,7	1559	9,201	2,547

наведеною гарантованою його оцінкою n^2. Навіть коли параметр зменшення об'єму множини локалізації ε-розв'язку на ітерації задається малим (0,99), середнє значення коефіцієнта розтягу простору виявляється істотно більшим за одиницю.

1.3. Про модифікацію методу еліпсоїдів

У даному підрозділі описана нова модифікація методу еліпсоїдів, збіжність якої гарантується на основі монотонного зменшення об'єму області локалізації точки мінімуму опуклої функції.

1.3.1. Метод еліпсоїдів

Метод еліпсоїдів (МЕ) був запропонований Д.Б. Юдіним і А.С. Немировським [50] та Н.З. Шором [44]. У ньому використовується еліпсоїд мінімального об'єму, описаний навколо напівкулі в n-вимірному евклідовому просторі E^n (рис. 1.3). Цей еліпсоїд має менший об'єм, ніж об'єм початкової кулі, і коефіцієнт зменшення об'єму визначається за формулою

$$q_n = \left(\frac{a}{r}\right)\left(\frac{b}{r}\right)^{n-1} = \sqrt{\frac{n-1}{n+1}}\left(\frac{n}{\sqrt{n^2-1}}\right)^n < \exp\left\{-\frac{1}{2n}\right\} < 1.$$

Для великих n значення коефіцієнта добре апроксимується асимптотичною формулою

$$q_n \approx 1 - \frac{1}{2n}.$$

На кожній ітерації МЕ будується наступний еліпсоїд із сталим коефіцієнтом зменшення об'єму q_n.

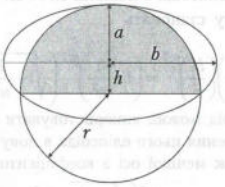


Рис. 1.3. Еліпсоїд мінімального об'єму, що містить у собі напівкулю з E^n

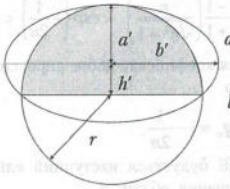
Недоліком МЕ є те, що він не спрацьовує в одновимірному випадку (тобто при $n = 1$). Причиною є те, що більша напіввісь еліпсоїда мінімального об'єму (див. рис. 1.3) обчислюється за формулою

$$b = r\frac{n}{\sqrt{n^2-1}},$$

яка не визначена, коли $n = 1$.

1.3.2. Модифікація методу еліпсоїдів

Головна ідея модифікації методу еліпсоїдів (ММЕ) полягає у використанні замість еліпсоїда найменшого об'єму іншого еліпсоїда, параметри якого зображені на рис. 1.4. Вони близькі до тих, що використовуються в методі еліпсоїдів.



$$a' = r\left(1-\frac{1}{n}\left(\sqrt{1+\frac{1}{n^2}}-\frac{1}{n}\right)\right)$$

$$b' = r\sqrt{1+\frac{1}{n^2}}$$

$$h' = \frac{r}{n}\left(\sqrt{1+\frac{1}{n^2}}-\frac{1}{n}\right)$$

Рис. 1.4. Новий еліпсоїд, що містить у собі напівкулю

Об'єм цього еліпсоїда є меншим за об'єм кулі, а коефіцієнт зменшення об'єму становить

$$Q_n = \left(\frac{a'}{r}\right)\left(\frac{b'}{r}\right)^{n-1} = \left(1+\frac{1}{n^2}\right)^{n/2}\left(\sqrt{1+\frac{1}{n^2}}-\frac{1}{n}\right) < 1.$$

Новий еліпсоїд можна використовувати також при $n = 1$.

Для перетворення цього еліпсоїда в нову кулю досить розтягти його вздовж меншої осі з коефіцієнтом розтягу

$$\alpha = \frac{a'}{b'}\left(\sqrt{1+\frac{1}{n^2}}+\frac{1}{n}\right).$$

Це можна зробити за допомогою оператора розтягу простору (див. [45])

$$R_\alpha(\xi) = I_n + (1-\alpha)\xi\xi^T, \quad \xi \in E^n, \quad \|\xi\| = 1,$$

де $(\cdot)^T$ означає транспонування; $\|\cdot\|$ – евклідова норма;

I_n – одинична матриця розміром $n \times n$; ξ – напрям розтягу простору; α – коефіцієнт розтягу простору.

1.3.3. Опис ММЕ

Нехай в E^n задано векторне поле $g(x)$ (не обов'язково неперервне), $n \geq 1$. Треба знайти таку точку x^*, що $(g(x), x - x^*) > 0$ для будь-яких $x \in E^n$. Вважається, що $g(x) \neq 0$ при $x \neq x^*$.

ММЕ для даної задачі реалізується за допомогою такої ітеративної процедури.

Ініціалізація. Фіксуємо початкову точку $x_0 \in E^n$ та початковий радіус r_0, що задовольняють умову

$$\|x_0 - x^*\| \leq r_0.$$

Обчислюємо величину за формулою

$$\beta = \frac{1}{\alpha} = \left(\sqrt{1+\frac{1}{n^2}}-\frac{1}{n}\right).$$

Нехай B_k, $k = 1, 2, \ldots$, – матриця розміром $n \times n$. Покладемо $B_0 := I^n$.

Першу ітерацію починаємо із значень x_0, r_0, B_0.

Нехай на k-й ітерації було знайдено величини $x_k \in E^n$, r_k, B_k. Тоді $(k+1)$-а ітерація полягає у виконанні такої послідовності дій.

1. Обчислюємо $g(x_k)$. Якщо $g(x_k) = 0$, то стоп: "$x^* = x_k$". У протилежному випадку переходимо до кроку 2.

2. Визначаємо

$$\xi_k = \frac{B_k^T g(x_k)}{\|B_k^T g(x_k)\|}.$$

3. Обчислюємо чергову точку:

$$x_{k+1} = x_k - h_k B_k \xi_k, \text{ де } h_k = \frac{r_k}{n}\beta.$$

4. Обчислюємо матрицю

$$B_{k+1} = B_k R_\beta(\xi_k) \text{ та радіус } r_{k+1} = r_k\sqrt{1+\frac{1}{n^2}}.$$

5. Переходимо до $(k+1)$-ї ітерації із значеннями x_{k+1}, r_{k+1} та B_{k+1}.

1.3.4. Аналіз збіжності ММЕ

Для ММЕ справедлива така теорема.

Теорема 1.1. *Нехай* $A_k = B_k^{-1}$. *Тоді послідовність точок* $\{x_k\}_{k=0}^{\infty}$, *породжена ММЕ, задовольняє нерівність*

$$\| A_k(x_k - x^*) \| \le r_k,\ k = 0, 1, 2, \ldots$$

Доведення теореми 1.1 аналогічне доведенню теореми 3.14 з книги [45].

Множина точок x, для яких виконується нерівність $\|A_k(x_k - x)\| \le r_k$, – це еліпсоїд Φ_k, що містить точку x^* та має об'єм

$$\mathrm{vol}(\Phi_k) = \frac{v_0 r_k^n}{\det A_k}.$$

Тут v_0 – об'єм n-вимірної одиничної кулі.

Швидкість збіжності ММЕ встановлюється такою теоремою.

Теорема 1.2. *Коефіцієнт зменшення об'єму* Q_n *є сталим*

$$Q_n = \frac{\mathrm{vol}(\Phi_{k+1})}{\mathrm{vol}(\Phi_k)} = \left(1 + \frac{1}{n^2}\right)^{n/2} \left(\sqrt{1 + \frac{1}{n^2}} - \frac{1}{n}\right) < \exp\left\{-\frac{1}{2n} + \frac{1}{2n^2}\right\} < 1$$

і не залежить від номера ітерації k.

Наведемо два наслідки з теореми 1.2.

1. Для великих n коефіцієнт Q_n апроксимується асимптотичною формулою

$$Q_n \approx 1 - \frac{1}{2n}.$$

Отже, запропонована модифікація методу еліпсоїдів має таку ж асимптотичну швидкість збіжності, як і відомий метод еліпсоїдів Юдіна–Немировського–Шора.

2. Якщо $n = 1$, то коефіцієнт $Q_1 = 2 - \sqrt{2} \approx 0.5858$.

Таким чином, цією модифікацією можна успішно користуватися і в одновимірному випадку.

Ці наслідки характеризують граничні випадки МЕМ. У загальному випадку для довільного $n \ge 2$ маємо $Q_n > q_n$.

Проаналізуємо швидкість збіжності МЕ і ММЕ при n в межах першої десятки, де МЕ є досить ефективним.

Розглянемо таблицю, де наведено дані експериментальних розрахунків для $n = 2, 3, \ldots, 10$: значення q_n, Q_n, їх відношення Q_n/q_n, кількість ітерацій K_1 для МЕ та K_2 для ММЕ, що необхідні для розв'язання задачі з відносною точністю 10^{-10}.

Таблиця 1.8. Порівняльний аналіз МЕ та ММЕ

n	q_n	Q_n	Q_n / q_n	K_1	K_2
2	0,7698004	0,7725425	1,0035621	177	179
3	0,8437500	0,8441633	1,0004898	407	408
4	0,8813189	0,8814234	1,0001186	730	730
5	0,9042245	0,9042600	1,0000392	1144	1144
6	0,9196855	0,9197001	1,0000159	1651	1651
7	0,9308347	0,9308416	1,0000074	2249	2250
8	0,9392592	0,9392628	1,0000038	2940	2940
9	0,9458508	0,9458528	1,0000021	3723	3723
10	0,9511498	0,9511510	1,0000012	4598	4598

Кількості ітерацій K_1 і K_2 обчислювалися за формулами

$$K_1 = \frac{-10n \ln 10}{\ln q_n} + 1,\quad K_2 = \frac{-10n \ln 10}{\ln Q_n} + 1$$

для досягнення точності за об'ємом

$$(q_n)^{K_1} \le 10^{-10n},\ (Q_n)^{K_2} \le 10^{-10n}.$$

З таблиці випливає, що для всіх наведених там значень n кількості ітерацій для МЕ (K_1) та ММЕ (K_2) фактично збігаються. Це означає, що обидва методи однаково ефективно спрацьовують при розв'язанні задач для $n \le 10$.

Розділ 2. Схеми декомпозиції в задачах опуклого програмування

Під декомпозицією розуміють підхід, який дає можливість замінити розв'язок однієї складної задачі послідовністю розв'язків простіших задач (підзадач). Огляд різних алгоритмів декомпозиції в задачах оптимізації великої розмірності можна знайти, наприклад, в [32, 37, 45, 153]. Головним чином ці алгоритми застосовуються до задач лінійного програмування спеціальної структури.

У даному розділі розглядаються дві схеми декомпозиції – декомпозиція за обмеженнями і декомпозиція за змінними, наводяться відомі і нові результати. При викладі матеріалу будемо спиратися на праці [45, 153]. Нові результати стосуються застосування декомпозиції за змінними в нелінійних задачах опуклого програмування [15, 16].

Традиційно в схемах декомпозиції при розв'язанні кожної підзадачі знаходяться точні розв'язки, за допомогою яких формуються субградієнти координуючої задачі. Для нелінійних задач існуючі алгоритми формують тільки наближення до оптимального розв'язку, тобто точні розв'язки не можуть бути знайдені. Щоб подолати цю проблему, пропонується перехід до обчислення ε-субградієнтів на підставі наближених розв'язків підзадач. Наводяться результати, потрібні для розробки обчислювальних алгоритмів.

Інша проблема при застосуванні декомпозиції за змінними полягає в тому, що підзадачі, які формуються, мають розв'язки не при всіх значеннях зв'язуючих змінних. Ця проблема ускладнюється тим, що і функції, які входять в підзадачі, можуть бути визначені на обмежених множинах. Спеціальне введення допоміжних змінних та штрафних функцій дозволяє уникнути вказаних проблем.

Результати, що наводяться, дають можливість значно розширити коло задач, до яких можуть застосовуватися схеми декомпозиції. Найбільш ефективне таке застосування у випадках, коли нелінійних підзадач небагато, їх розмірність невелика, а всі інші підзадачі мають спеціальну структуру і спеціальні швидкі алгоритми їх розв'язання. У зв'язку з тим, що мережеві задачі оптимізації є задачами великої і надвеликої розмірності і, як правило, мають блочну структуру, використання декомпозиції – єдиний підхід, який дає змогу розв'язувати ці задачі.

2.1. Декомпозиція за обмеженнями

Нехай $x^1, \ldots, x^Q$ – вектори розміром $l_1, \ldots, l_Q$, відповідно. Розглянемо задачу опуклого програмування такого виду:

знайти

$$F^* = \min \sum_{q=1}^{Q} f_0^q(x^q) \tag{2.1}$$

при обмеженнях

$$\sum_{q=1}^{Q} f_i^q(x^q) \le 0, \quad i = 1,\ldots,n, \tag{2.2}$$

$$h_j^q(x^q) \le 0, \quad j = 1,\ldots,m_q, \quad q = 1,\ldots,Q, \tag{2.3}$$

де всі функції f і h з відповідними індексами є опуклими. Будемо говорити, що задача (2.1)–(2.3) має **квазіблочну структуру** із зв'язуючими обмеженнями (2.2).

Розглянемо функцію Лагранжа

$$L(x,u) = \sum_{q=1}^{Q} [f_0^q(x^q) + \sum_{i=1}^{n} u_i f_i^q(x^q)],$$

де $u = (u_1, \ldots, u_n)$ – вектор множників Лагранжа, і задачу:

знайти

$$\Psi(u) = \inf_x L(x,u) \tag{2.4}$$

при обмеженнях

$$h_j^q(x^q) \le 0, \quad j = 1,...,m_q, \quad q = 1,...,Q. \tag{2.5}$$

Розглянемо також задачу:

знайти

$$\Psi^* = \sup\{\Psi(u): \ u \ge 0\}. \tag{2.6}$$

Задачу (2.4), (2.5) будемо називати внутрішньою, задачу (2.6) – координуючою.

Неважко бачити, що внутрішню задачу можна розбити на Q менших підзадач:

знайти

$$\Psi_q(u) = f_0^q(x^q) + \sum_{i=1}^{n} u_i f_i^q(x^q) \to \min, \tag{2.7}$$

при обмеженнях

$$h_j^q(x^q) \le 0, \quad j = 1,...,m_q, \quad x^q \in E^{l_q}, \tag{2.8}$$

для всіх $q = 1, ..., Q$. Тоді

$$\Psi(u) = \sum_{q=1}^{Q} \Psi_q(u). \tag{2.9}$$

Схема декомпозиції за обмеженнями полягає в розв'язанні координуючої задачі (2.6). Для обчислення значень функції $\Psi(u)$ для кожного набору значень множників Лагранжа u розв'язуються Q підзадач (2.7)–(2.8).

Відомо [45], що $\Psi(u)$ і $\Psi_q(u)$ увігнуті по u, а значення Ψ^* дорівнює оптимальному значенню цільової функції F^* початкової задачі.

Позначимо $L^q(x^q,u) = f_0^q(x^q) + \sum_{i=1}^{n} u_i f_i^q(x^q)$, $x^q(u)$ – оптимальний розв'язок задачі (2.7)–(2.8), тобто $\Psi_q(u) = L^q(x^q(u), u)$. Будемо говорити, що вектор $x_\varepsilon^q(u)$ – ε-оптимальний розв'язок задачі (2.7)–(2.8), якщо вектор задовольняє обмеження (2.8) і $L^q(x_\varepsilon^q(u), u) \le L^q(x^q(u), u) + \varepsilon$.

Правила обчислювання суперградієнтів і ε-суперградієнтів функцій $\Psi_q(u)$ визначаються в такій лемі.

Лема 2.1. Нехай $x_\varepsilon^q(u)$ – ε-оптимальний розв'язок задачі (2.7)–(2.8). Тоді ε-суперградієнт $g_{\Psi_q}^\varepsilon(u)$ функції $\Psi_q(u)$ має вигляд

$$g_{\Psi_q}^\varepsilon(u) = (f_1^q(x_\varepsilon^q(u)), \ ..., \ f_n^q(x_\varepsilon^q(u))). \tag{2.10}$$

Доведення. Оскільки $L^q(x^q(u), u) \ge L^q(x_\varepsilon^q(u), u) - \varepsilon$, легко перевірити, що

$$\Psi_q(u') - \Psi_q(u) =$$

$$= L^q(x^q(u'),u') - L^q(x^q(u),u) \le L^q(x_\varepsilon^q(u),u') - L^q(x^q(u),u) \le$$

$$\le L^q(x_\varepsilon^q(u),u') - L^q(x_\varepsilon^q(u),u) + \varepsilon = \sum_{i=1}^{n} (u_i' - u_i) f_i^q(x_\varepsilon^q(u)) + \varepsilon,$$

звідки випливає твердження леми.

Таким чином, координуюча задача (2.6) є задачею опуклого програмування, формула (2.10) дає змогу нам обчислювати суперградієнти функцій $\Psi_q(u)$ і для розв'язання задачі (2.6) можна скористатися субградієнтними алгоритмами (мінімізується функція $\Phi(u) = -\Psi(u)$).

У багатьох випадках підзадачі (2.7), (2.8) мають спеціальну структуру, для їх розв'язання розробляються відповідні ефективні методи. Такі підходи розвиваються в наступних розділах.

Питання визначення оптимального розв'язку початкової задачі (2.1)–(2.3), якщо відомий розв'язок u^* координуючої задачі (2.6), розглядаються в працях [45, 153].

2.2. Декомпозиція за змінними

2.2.1. Загальна схема та основні властивості

Розглянемо задачу опуклого програмування, змінні в якій згруповані у дві підмножини. В задачі треба знайти

$$\inf_{x,y} f_0(x,y) \tag{2.11}$$

при обмеженнях

$$f_i(x,y) \le 0, \quad i = 1,...,n, \tag{2.12}$$

де x та y – вектори змінних: $x = (x_1, ..., x_l)$, $y = (y_1, ..., y_m)$, а f_i, $i = 0, 1,...,n$, – опуклі функції $(l + m)$-вимірного вектора $(x, y) \in E^l \times E^m$.

Введемо позначення $D = \{(x,y) : f_i(x,y) \le 0, i = 1, ..., n\}$. Вважається, що D – замкнута множина, для якої виконується умова Слейтера, $D \subset \text{int dom } f_i$, $i = 0, 1, ..., n$. Зафіксуємо $x = \overline{x}$ та розглянемо таку задачу:

знайти

$$\inf_y f_0(\overline{x}, y) \tag{2.13}$$

при обмеженнях

$$f_i(\overline{x}, y) \le 0, \ i = 1,...,n. \tag{2.14}$$

Неважко помітити, що задача (2.13)–(2.14) є задачею опуклого програмування. Позначимо W множину тих значень $\overline{x}$, для яких розв'язок задачі (2.13)–(2.14) існує, $y(\overline{x})$ – розв'язок задачі (2.13)–(2.14). Введемо функцію Ф:

$$\Phi(\overline{x}) = \inf_{y \in D(x)} f_0(\overline{x}, y), \tag{2.15}$$

де $\overline{x} \in W$; $D(\overline{x})$ – множина всіх y, які задовольняють (2.14).

Розглянемо задачу: знайти

$$\inf \{\Phi(x) : \ x \in W\} \tag{2.16}$$

Нехай x^* – розв'язок задачі (2.16). Легко бачити, що вектор $(x^*, y(x^*))$ є розв'язком задачі (2.11)–(2.12).

Задачу (2.13)–(2.14) будемо називати внутрішньою, задачу (2.16) – координуючою.

Схему розв'язання задачі (2.11)–(2.12), при якій розв'язується задача (2.16), а для обчислення значень функції Ф розв'язується задача (2.13)–(2.14), будемо називати схемою декомпозиції за змінними.

Властивості функції Ф формулюються в такій теоремі.

Теорема 2.1. [45]. *Якщо $f_i(x, y)$, $i = 0, 1, ..., n$, – опуклі функції і задача (2.11)–(2.12) має оптимальний розв'язок (x^*, y^*), то функція Ф, введена згідно (2.15), є власною опуклою функцією (тобто вона скінченна на деякій опуклій множині W з E^l). Якщо для деякого $\overline{x} \in \text{int } W$ виконується умова Слейтера для (2.14), то субградієнт функції Ф в точці $x = \overline{x}$ можна знайти за формулою*

$$g_\Phi(\overline{x}) = g_{L_u^*}(\overline{x}, y(\overline{x})), \tag{2.17}$$

де

$$L_u(x,y) = f_0(x,y) + \sum_{i=1}^{n} u_i f_i(x,y),$$

$y(\overline{x})$ є одним з оптимальних значень в задачі (2.13)–(2.14); $u = \{u_i\}$ – множники Лагранжа для (2.13)–(2.14), одержані згідно з теоремою Куна-Такера, а $g_{L_u^*}(\overline{x}, y(\overline{x}))$ – проекція субградієнта функції $L_u(x, y)$ на підпростір $E^l_{(x)}$, який має нульову проекцію на підпростір $E^m_{(y)}$ (субградієнт береться в точці $(\overline{x}, y(\overline{x}))$.

Наслідок 2.1. Якщо в наведеній вище теоремі припустити, що функції $f_i(x, y)$, $i = 0, 1, ..., n$, неперервно диференційовані по y, то формулу (2.17) можна спростити:

$$g_\Phi(\overline{x}) = g^x_{f_0}(\overline{x}, y(\overline{x})) + \sum_{i=1}^{n} u_i(\overline{x}) g^x_{f_i}(\overline{x}, y(\overline{x})), \tag{2.18}$$

де $g^x_{f_i}(\overline{x}, y(\overline{x}))$ – проекції довільних субградієнтів функції f_i в точці $(\overline{x}, y(\overline{x}))$ на $E^l_{(x)}$, $i = 0, 1, ..., n$.

Таким чином, координуюча задача (2.16) є задачею опуклого програмування, формула (2.18) дає можливість нам обчислювати субградієнти функції Ф і для розв'язання задачі (2.16) можуть використовуватися субградієнтні алгоритми.

Результати, які сформовані, застосовуються у випадку, коли внутрішню задачу (2.13)–(2.14) можна розв'язати за скінченну кількість кроків, а також одержати множники Лагранжа та обчислити "часткові" субградієнти $g^x_{f_i}$, $i = 0, 1, ..., n$. Це при-

пущення виконується, наприклад, якщо (2.13)–(2.14) є задачею лінійного чи квадратичного програмування.

Додаткові труднощі можуть виникати, якщо задача (2.13)–(2.14) не має розв'язків для деякого $\bar{x}$. Стандартний спосіб перебороти ці труднощі у випадку, коли функції $f_i(x, y)$, $i = 0, 1, ..., n$ визначені на всьому просторі $E^l \times E^m$, полягає у використанні штрафної функції [45]. При цьому задача (2.11)–(2.12) замінюється на таку:

знайти

$$\min_{x,y,v} f_0(x,y) + M\sum_{i=1}^{n} v_i \tag{2.19}$$

при обмеженнях

$$f_i(x,y) - v_i \le 0, \qquad v_i \ge 0, \qquad i \in \{1,...,n\}, \tag{2.20}$$

де M досить велике додатне число. Задача (2.13)–(2.14) замінюється на таку:

знайти

$$\min_{y,v} f_0(\bar{x},y) + M\sum_{i=1}^{n} v_i \tag{2.21}$$

при обмеженнях

$$f_i(\bar{x},y) - v_i \le 0, \qquad v_i \ge 0, \qquad i \in \{1,...,n\}. \tag{2.22}$$

Ясно, що для будь-якого $\bar{x}$ задача (2.21)–(2.22) має непусту допустиму множину, яка задовольняє умову Слейтера.

При застосуванні схеми декомпозиції за змінними будемо вважати, що задачу (2.11)–(2.12) вже зведено до вигляду, за яким функція $\Phi(\bar{x})$ визначена для всіх $\bar{x}$. Тоді координуюча задача (2.16) є задачею опуклого програмування без обмежень.

Опишемо детальніше правило обчислення субградієнта $g_\Phi(\bar{x})$ для випадку, коли (2.11)–(2.12) є задачею лінійного програмування. Нехай треба знайти

$$\max[(c,x) + (d,y)] \tag{2.23}$$

при обмеженнях

$$Ax + By \le e, \tag{2.24}$$

$$x \ge 0,\ y \ge 0, \tag{2.25}$$

де $x \in E^l$ та $y \in E^m$ – вектори змінних; A – матриця розміром $n \times l$; B – матриця розміром $n \times m$; $c \in E^l$, $d \in E^m$ та $e \in E^n$ – фіксовані вектори. Зафіксуємо $x = \bar{x}$ та розглянемо задачу:

знайти

$$\max_{y}[(d,y) + (c,\bar{x})] \tag{2.26}$$

при обмеженнях

$$By \le e - A\bar{x}, \tag{2.27}$$

$$y \ge 0, \tag{2.28}$$

та двоїсту задачу:

знайти

$$\min_{v}(e - A\bar{x}, v), \tag{2.29}$$

при обмеженнях

$$B^T v \ge d, \tag{2.30}$$

$$v \ge 0, \tag{2.31}$$

де $v \in E^n$ – вектор змінних; B^T – транспонована матриця B.

Нехай $v(\bar{x})$ – розв'язок задачі (2.29)–(2.31). Тоді для обчислення субградієнта $g_\Phi(\bar{x})$ маємо просте співвідношення

$$g_\Phi(\bar{x}) = c + A^T v(\bar{x}). \tag{2.32}$$

2.2.2. Особливості застосування для нелінійних задач

Розглянемо ситуацію, коли задача (2.13)–(2.14) є задачею опуклого програмування загального виду. Існуючі алгоритми сходяться до оптимального розв'язку таких задач як до гранич-

ної точки, тобто задачу (2.13)–(2.14) не можна розв'язати за скінченну кількість кроків. Оскільки функції $f_i(x, y)$, $i = 0, 1, ..., n$, визначені на обмежених множинах, регуляризація (2.19)–(2.20), (2.21)–(2.22) також не може бути застосована.

Для подолання першої проблеми доводиться переходити до обчислення ε-субградієнтів функції $\Phi(x)$. За допомогою наведених нижче результатів можна обчислювати ε-субградієнти функції $\Phi(x)$, базуючись на наближених розв'язках задачі (2.13)–(2.14). Процес розв'язування задачі (2.13)–(2.14) закінчується при досягненні заданого значення ε.

Регуляризації задачі (2.11)–(2.12) буде розглянута в п. 2.2.3.

Теорема 2.2. *Нехай* $(\overline{x}, \overline{y}) \in \operatorname{int} \operatorname{dom} f_i$, $i = 0, ..., n$, $\overline{x} \in \operatorname{int} W$, *для допустимої множини* $D(\overline{x})$ *задачі* (2.13)–(2.14) *виконується умова Слейтера, задано числа* $\varepsilon_i \geq 0$, $i = 0, ..., n$, *для яких обчислено* ε_i*-субградієнти* $g^i = (g^i_x, g^i_y)$ *функцій* f_i *у точці* $(\overline{x}, \overline{y})$ *такі, що для деяких чисел* $\overline{u}_i \geq 0$, $i = 1, ..., n$, *виконуються співвідношення*

$$g^0_y + \sum_{i=1}^{n} \overline{u}_i g^i_y = 0, \tag{2.33}$$

$$\overline{u}_i \geq 0, \quad i = 1, ..., n. \tag{2.34}$$

Тоді:

1) $\Phi(\overline{x}) \geq f_0(\overline{x}, \overline{y}) - \overline{\varepsilon}$;

2) *якщо* $\overline{y} \in D(\overline{x})$, *то* $\overline{\varepsilon}$*-субградієнт функції* $\Phi(x)$ *у точці* $x = \overline{x}$ *можна обчислити за формулою*

$$g^{\overline{\varepsilon}}_{\Phi}(\overline{x}) = g^0_x + \sum_{i=1}^{n} \overline{u}_i g^i_x, \tag{2.35}$$

де

$$\overline{\varepsilon} = \varepsilon_0 + \sum_{i=1}^{n} \overline{u}_i (\varepsilon_i - f_i(\overline{x}, \overline{y})). \tag{2.36}$$

Доведення. Розглянемо перше твердження. Використовуючи теорему Куна-Такера, маємо

$$\Phi(\overline{x}) = \max_{u \geq 0} \min_{y} \left\{ f_0(\overline{x}, y) + \sum_{i=1}^{n} u_i f_i(\overline{x}, y) \right\} \geq$$

$$\geq \min_{y} \left\{ f_0(\overline{x}, \overline{y}) + (g^0_y, y - \overline{y}) - \varepsilon_0 + \sum_{i=1}^{n} \overline{u}_i (f_i(\overline{x}, \overline{y}) + (g^i_y, y - \overline{y}) - \varepsilon_i) \right\}.$$

З цього з урахуванням (2.33) одержуємо твердження теореми.

Доведемо друге твердження. Нехай у деякій точці $x \in \operatorname{int} W$ для допустимої множини $D(x)$ задачі (2.13)–(2.14) виконується умова Слейтера. Позначимо $y(x)$ розв'язок задачі (2.13)–(2.14), $u_i(x) \geq 0$, $i = 1, ..., n$, – значення множників Лагранжа такі, що

$$\Phi(x) = L_{u(x)}(x, y(x)) = f_0(x, y(x)) + \sum_{i=1}^{n} u_i(x) f_i(x, y(x)) =$$

$$= \min_{y} \left\{ f_0(x, y) + \sum_{i=1}^{n} u_i(x) f_i(x, y) \right\} =$$

$$= \max_{u \geq 0} \min_{y} \left\{ f_0(x, y) + \sum_{i=1}^{n} u_i f_i(x, y) \right\}.$$

Згідно з теоремою Куна-Такера такі множники існують. За умовою теореми в точці $\overline{x}$ для допустимої множини $D(\overline{x})$ умова Слейтера виконується. Розглянемо точку $x' \in \operatorname{int} W$, досить близьку до $\overline{x}$, в якій для множини $D(x')$ також виконується умова Слейтера (така точка існує через неперервність функцій $f_i(x, y)$, $i = 1, ..., n$).

Позначимо $y' = y(x')$. Очевидно,

$$\Phi(x') = L_{u(x')}(x', y') \geq L_{\overline{u}}(x', y').$$

Оскільки $\overline{y}$ – допустима точка задачі (2.13)–(2.14), а $\Phi(\overline{x})$ – значення цільової функції в оптимальній точці, то $\Phi(\overline{x}) \leq f_0(\overline{x}, \overline{y})$. Звідси одержуємо

$$\Phi(x') - \Phi(\overline{x}) \geq L_{\overline{u}}(x', y') - f_0(\overline{x}, \overline{y}) = f_0(x', y') - f_0(\overline{x}, \overline{y}) +$$

$$+ \sum_{i=1}^{n} \overline{u}_i \left[f_i(x', y') - f_i(\overline{x}, \overline{y}) + f_i(\overline{x}, \overline{y}) \right] \geq (x' - \overline{x}, g^0_x) + (y' - \overline{y}, g^0_y) -$$

$$-\varepsilon_0 + \sum_{i=1}^{n} \overline{u}_i \left[(x' - \overline{x}, g_x^i) + (y' - \overline{y}, g_y^i) - \varepsilon_i + f_i(\overline{x}, \overline{y}) \right] =$$

$$= (x' - \overline{x}, g_x^0 + \sum_{i=1}^{n} \overline{u}_i g_x^i) - \varepsilon_0 + \sum_{i=1}^{n} \overline{u}_i \left[-\varepsilon_i + f_i(\overline{x}, \overline{y}) \right].$$

Остання рівність виконується згідно з (2.33). Таким чином, $g_{\Phi}^{\varepsilon}(\overline{x}) = g_x^0 + \sum_{i=1}^{n} \overline{u}_i g_x^i$. Теорема доведена.

Наслідок 2.2. Для довільної точки $\overline{y} \in D(\overline{x})$ при досить великих $\varepsilon_i \geq 0$ існують ε_i-субградієнти $g_i = (g_x^i, g_y^i)$ функцій f_i у точці $(\overline{x}, \overline{y})$ та числа $\overline{u}_i$, $i = 1, ..., n$, для яких виконуються співвідношення (2.33), (2.34).

Для доведення розглянемо y^* – розв'язок задачі (2.13)–(2.14). З необхідних умов оптимальності випливає існування субградієнтів $g_i = (g_x^i, g_y^i)$ функцій f_i у точці $(\overline{x}, y^*)$, для яких виконуються співвідношення (2.33), (2.34). Неважко перевірити, що ці субградієнти будуть ε_i-субградієнтами функцій f^i у точці $(\overline{x}, \overline{y})$, де $\varepsilon_i = f^i(\overline{x}, \overline{y}) - f^i(\overline{x}, y^*) - (g_y^i, \overline{y} - y^*)$, $i = 0, ..., n$.

Нехай функції $f_i(x, y)$, $i = 0, 1, ..., n$, неперервно диференційовані, y^* – розв'язок задачі (2.13)–(2.14), $g^{*i} = (g_x^{*i}, g_y^{*i})$ – градієнти функцій f_i, $i = 0, 1, ..., n$, у точці $(\overline{x}, y^*)$. Позначимо u_i^*, $i = 1, ..., n$, оптимальні значення множників Лагранжа задачі (2.13)–(2.14), $I^* = \{i : i \in \{1, ..., n\}, u_i^* > 0\}$.

Наслідок 2.3. Якщо сукупність векторів $\{g_y^{*i}, i \in I^*\}$ є лінійно незалежною й утворює базис в E^m, $\overline{y} \in D(\overline{x})$, то існує досить мале $\sigma > 0$, таке, що при $\| \overline{y} - y^* \| \geq \sigma$ співвідношення (2.33), (2.34) виконуються при деяких $\overline{u}_i$, де $g_i = (g_x^i, g_y^i)$ – градієнти функцій f_i, $i = 1, ..., n$, у точці $(\overline{x}, \overline{y})$.

Доведення випливає з неперервної диференційовності функцій f_i, $i = 1, ..., n$. При цьому одним із можливих розв'язків буде такий, що $\overline{u}_i = 0$ при $i \in I^*$, інакше $-\overline{u}_i > 0$.

Розглянемо, як отримані результати можна застосувати до задачі лінійного програмування. Нехай

$$f_i(x, y) = a_x^i x + a_y^i y + b^i, \quad i = 0, ..., n,$$

де a_x^i, a_y^i – вектори-рядки відповідного розміру. Позначимо

$$A_x = \begin{Bmatrix} a_x^1 \\ ... \\ a_x^n \end{Bmatrix}, \quad A_y = \begin{Bmatrix} a_y^1 \\ ... \\ a_y^n \end{Bmatrix}, \quad b = \begin{Bmatrix} b^1 \\ ... \\ b^n \end{Bmatrix},$$

$$b(x) = b + A_x x, \quad b^0(x) = b^0 + a_x^0 x.$$

Тоді задачу лінійного програмування виду (2.13)–(2.14) та двоїсту до неї можна подати у вигляді

$$\min_y \left\{ a_y^0 y + b^0(x) : A_y y + b(x) \leq 0 \right\}, \tag{2.37}$$

$$\max_u \left\{ b(x)u + b^0(x) : A_y^T u + (a_y^0)^T = 0, \quad u \geq 0 \right\}. \tag{2.38}$$

Очевидно, що при виконанні умов теореми 2.2 можна вважати $\varepsilon_i = 0$, $i = 0, 1, ..., n$, а співвідношення (2.33), (2.34) є умовами допустимості двоїстої задачі (2.38).

Лема 2.2. *При виконанні умов теореми 2.2 для задачі (2.37) виконується рівність* $\overline{\varepsilon} = a_y^0 \overline{y} - b(\overline{x})^T \overline{u}$.

Доведення. Маємо

$$\overline{\varepsilon} = -\sum_{i=1}^{n} \overline{u}_i f_i(\overline{x}, \overline{y}) = -(A_y \overline{y} + b(\overline{x}))^T \overline{u} = a_y^0 \overline{y} - a_y^0 \overline{y} -$$

$$-(A_y \overline{y} + b(\overline{x}))^T \overline{u} = a_y^0 \overline{y} - b(\overline{x})^T \overline{u}.$$

Лема доведена.

Таким чином, якщо $\overline{u} \in \overline{\varepsilon}$-оптимальний розв'язок двоїстої задачі (2.38), то співвідношення (2.35) визначають $\overline{\varepsilon}$-субградієнт оптимального значення задачі (2.37).

Сформульована лема визначає просте правило зупинки при наближеному розв'язанні задач лінійного програмування й обчисленні $\bar{\varepsilon}$-субградієнтів функції $\Phi(\bar{x})$. При цьому мають використовуватися алгоритми розв'язання задач лінійного програмування, що генерують на кожній ітерації допустимі точки прямої і двоїстої задач (див., наприклад, [49]). Близькі результати для задач лінійного програмування отримані в [100].

У загальному випадку для обчислення $\bar{\varepsilon}$-субградієнтів функції $\Phi(\bar{x})$ мають застосовуватися спеціальні алгоритми. Результати, корисні для розробки таких алгоритмів, наводяться нижче.

Нехай задана сукупність $\{y^1, y^2, ..., y^s\}$ точок з E^m, $\bar{y} \in D(\bar{x})$, $\overline{f}_i = f_i(\bar{x}, \bar{y})$, у кожній точці $(\bar{x}, y^k)$ обчислені значення $f^{ik} = f_i(\bar{x}, y^k)$ і субградієнти $g^{ik} = (g_x^{ik}, g_y^{ik})$ функцій f_i, $i = 0, ..., n$, $k = 1, ..., s$.

Розглянемо лінійну апроксимацію задачі (2.13)–(2.14) при фіксованому $x = \bar{x}$, яка може бути подана у вигляді:

знайти

$$\min_{y,\xi} \xi \tag{2.39}$$

при обмеженнях

$$f^{ik} + (g_y^{ik}, y - y^k) \le \xi, \qquad k = 1, ..., s, \ i = 0, \tag{2.40}$$

$$f^{ik} + (g_y^{ik}, y - y^k) \le 0, \qquad k = 1, ..., s, \quad i = 1, ..., n. \tag{2.41}$$

Нехай u_{0k}, u_{ik} – двоїсті змінні, що відповідають обмеженням (2.40), (2.41), $k = 1, ..., s$, $i = 1, ..., n$. Двоїста задача записується в такий спосіб:

знайти

$$\max_{u} \sum_{i=0}^{n} \sum_{k=1}^{s} \left(f^{ik} - (g_y^{ik}, y^k) \right) u_{ik}, \tag{2.42}$$

при обмеженнях

$$\sum_{k=1}^{s} u_{0k} = 1, \tag{2.43}$$

$$\sum_{i=0}^{n} \sum_{k=1}^{s} u_{ik} g_y^{ik} = 0, \tag{2.44}$$

$$u_{ik} \ge 0, \quad i = 0, ..., n, \quad k = 1, ..., s. \tag{2.45}$$

Позначимо: y^*, ξ^* – розв'язок задачі (2.40), (2.41), $\overline{u}_{ik}$ – оптимальні значення двоїстих перемінних $k = 1, ..., s$, $i = 0, ..., n$.

Лема 2.3. *Нехай для деяких i, k виконується $\overline{u}_{ik} > 0$. Тоді g^{ik} є ε^{ik}-субградієнт функції f_i в точці $(\bar{x}, \bar{y})$, де*

$$\varepsilon^{ik} = \begin{cases} \overline{f}_i + (g_y^{ik}, y^* - \bar{y}), & i > 0, \\ \overline{f}_0 - \xi^* + (g_y^{0k}, y^* - \bar{y}), & i = 0. \end{cases} \tag{2.46}$$

Доведення. Легко бачити, що

$$\varepsilon^{ik} = \overline{f}_i - f^{ik} - (g_y^{ik}, \bar{y} - y^k) = \overline{f}_i - f^{ik} - (g_y^{ik}, \bar{y} - y^k + y^* - y^*) =$$
$$= \overline{f}_i + (g_y^{ik}, y^* - \bar{y}) - \left[f^{ik} + (g_y^{ik}, y^* - y^k) \right].$$

Оскільки $u_{ik} > 0$, то відповідне обмеження виконується як рівність в оптимальній точці. Звідси випливає твердження леми. Лема доведена.

Лема 2.4. *Нехай $\overline{u}_i = \sum_{k=1}^{s} \overline{u}_{ik} > 0$. Тоді вектор $g^i = (\overline{u}_i)^{-1} \left(\sum_{k=1}^{s} \overline{u}_{ik} g^{ik} \right)$ є ε^i-субградієнтом функції f_i в точці $(\bar{x}, \bar{y})$, де $\varepsilon^i = (\overline{u}_i)^{-1} \left(\sum_{k=1}^{s} \overline{u}_{ik} \varepsilon^{ik} \right)$, ε^{ik} визначено відповідно до леми 2.3.*

Доведення. Відповідно з лемою 2.3 маємо $f_i(x, y) \ge \overline{f}_i + (g_x^{ik}, x - \bar{x}) + (g_y^{ik}, y - \bar{y}) - \varepsilon^{ik}$, $k = 1, ..., s$. Множачи нерівності на $\overline{u}_{ik}$ і підсумовуючи по k, одержуємо твердження леми.

Теорема 2.3. *$\bar{\varepsilon}$-субградієнт функції $\Phi(x)$ в точці $x = \bar{x}$ має вигляд*

$$g_\Phi^{\pi}(\bar{x}) = \sum_{k=1}^{s}\sum_{t=0}^{n} \overline{u}_{tk} g_x^{tk}, \qquad (2.47)$$

$$\overline{\varepsilon} = \overline{f}_0 - \xi^*. \qquad (2.48)$$

Доведення. Неважко бачити, що вектори $g^t = (g_x^t, g_y^t)$ і числа u_t, визначені згідно з лемою 2.4, задовольняють співвідношення (2.33), (2.34). Підставляючи вирази для g^t в (2.35), (2.36) і з огляду на те, що $\sum_{k=1}^{s} \overline{u}_{0k} = 1$, одержуємо

$$\overline{\varepsilon} = \overline{f}_0 - \xi^* + \left(\sum_{k=1}^{s}\sum_{t=0}^{n} \overline{u}_{tk} g_y^{tk},\ \ y^* - \overline{y} \right) = \overline{f}_0 - \xi^*.$$

Остання рівність випливає з (2.44). Теорему доведено.

За допомогою теореми 2.3 можна будувати правила зупинки в процесах наближеного розв'язування задачі (2.13)–(2.14). Для цього необхідно, щоб допустимі точки $\overline{y}$ генерувалися деяким процесом, що сходиться до розв'язку задачі (2.13)–(2.14) і оптимальні значення лінійних апроксимацій також сходилися до оптимального значення задачі (2.13)–(2.14).

Сформульовані умови виконуються, якщо обмеження задачі (2.13)–(2.14) лінійні, а для розв'язання використовується метод Келлі [49]. Тоді всі точки, що генеруються, є допустимими й оптимальні значення лінійних апроксимацій сходяться до оптимального значення задачі (2.13)–(2.14).

У випадку нелінійних обмежень необхідні додаткові зусилля для формування допустимих точок і спеціальним образом побудовані лінійні апроксимації, що забезпечують збіжність оптимальних значень.

При практичному формуванні лінійної апроксимації в задачу (2.39)–(2.41) включаються тільки найбільш суттєві обмеження. Правила, що при цьому використовуються, є евристичними, однак вони дають можливість істотно понизити розмірність задачі, що генерується.

2.2.3. Декомпозиція нелінійних квазіблочних задач

Нехай задані $f_i^q(x, y^q)$ – опуклі функції $(l + m_q)$-вимірного вектора, $x \in E^l$, $y^q \in E^{m_q}$, $i = 0, ..., n$, $q = 1, ..., Q$. Будемо вважати, що функції f_i^q визначені на обмежених множинах, тобто $\operatorname{dom} f_i^q \neq E^l \times E^{m_q}$. Позначимо $Y = (y^1, ..., y^Q)$, $F(x, Y) = \sum_{q=1}^{Q} f_0^q(x, y^q)$.

Розглянемо квазіблочну задачу математичного програмування зі з'єднуючими змінними:

знайти

$$\min_{x,\ Y} F(x,Y) = \min_{x,\ Y} \sum_{q=1}^{Q} f_0^q(x, y^q) \qquad (2.49)$$

при обмеженнях

$$f_i^q(x, y^q) \le 0, \quad i = 1,...,n_q, \ \ q = 1,...,Q. \qquad (2.50)$$

При фіксованих значеннях x задача (2.49), (2.50) розпадається на підзадачі виду (2.13)–(2.14), сформульовані для кожного q:

$$\Phi^q(x) = \begin{cases} \min\left\{ \ f_0^q(x,y^q) : \ y^q \in D_q(x) \ \right\}, & x \in W_q, \\ +\infty, \quad x \notin W_q, \end{cases} \qquad (2.51)$$

де $D_q(x) = \left\{ y^q : f_i^q(x, y^q) \le 0, \quad y^q \in E^{n_q} \right\}$; W_q – множина тих значень , для яких розв'язок задачі в (2.51) існує.

Координуюча задача має вигляд:

знайти

$$\min \left\{ \sum_{q=1}^{Q} \Phi^q(x) : x \in E^l \right\}. \qquad (2.52)$$

Безпосередньо використовувати задачі (2.51), (2.52) в обчислювальних алгоритмах важко з огляду на складність опису підмножин W_q, на яких функції $\Phi^q(x)$ набувають скінченних значень. Регуляризацію, запропоновану в підрозділі 2.1, не можна застосувати у зв'язку з тим, що функції $f_i^q(x, y^q)$ визначені на обмежених множинах. Тому будемо застосовувати ін-

ший прийом. Введемо допоміжні змінні $v^q \in E^l$, $q = 1, ..., Q$, і замість задачі (2.49), (2.50) розглянемо таку задачу:

знайти

$$\min_{x,\, y,\, v} \sum_{q=1}^{Q} \left\{ f_0^q(v^q, y^q) + M_q \sum_{j=1}^{l} | x_j - v_j^q | \right\} \qquad (2.53)$$

при обмеженнях

$$f_i^q(v^q, y^q) \le 0, \quad i = 1, ..., n_q, \quad q = 1, ..., Q. \qquad (2.54)$$

З властивостей негладких штрафних функцій [45] випливає така лема.

Лема 2.5. *Нехай задача* (2.49), (2.50) *має розв'язок. Тоді при досить великих додатних значеннях* M_q, $q = 1, ..., Q$, *розв'язки задач* (2.49), (2.50) *та* (2.53),(2.54) *збігаються.*

Позначимо $z^q = (v^q, y^q)$, $z^q \in E^l \times E^{m_q}$. При фіксованих значеннях x задача (2.53),(2.54) розпадається на підзадачі для кожного q:

знайти

$$\Psi^q(x) = \min_{z^q} \left\{ f_0^q(z^q) + M_q \sum_{j=1}^{l} | x_j - z_j^q | \right\} \qquad (2.55)$$

при обмеженнях

$$f_i^q(z^q) \le 0, \quad i = 1, ..., n_q \qquad (2.56)$$

Нехай системи обмежень задач (2.55),(2.56) сумісні, й ці задачі мають розв'язки при $M_q = 0$. Тоді $\Psi^q(x)$ функції визначені при будь-яких x, а координуюча задача

$$\min \left\{ \sum_{q=1}^{Q} \Psi^q(x) : \; x \in E^l \right\} \qquad (2.57)$$

є задачою опуклого програмування без обмежень. Для розв'язування задачі (2.57) можна скористатися алгоритмами недиференційовної оптимізації [3, 21, 100], що використовують на кожному кроці ε-субградієнт функції $\Psi(x) = \sum_{q=1}^{Q} \Psi^q(x)$. Для одержання таких ε-субградієнтів необхідно наближено розв'язувати задачі (2.55), (2.56) при $q = 1, ..., Q$ й будувати ε^q-субградієнти функцій $\Psi^q(x)$.

Розділ 3. Методи локального пошуку розв'язків дискретних задач оптимізації на мережах

Інтерес дослідників до наближених методів розв'язання задач дискретної оптимізації значною мірою стимулювала та обставина, що кількість обчислень, необхідна для всіх точних методів дискретної оптимізації при розв'язанні більшості задач, експоненціально зростає при збільшенні їх розмірностей. Це призводить до того, що навіть для задач невеликих розмірів не можна гарантувати отримання точного розв'язку за прийнятний час. Крім того, в умовах, коли необхідно оперативно розв'язувати оптимізаційні задачі в реальному масштабі часу, наближений розв'язок задачі, отриманий за прийнятний час, цінніший, ніж точний розв'язок, знайдений через значний проміжок часу. При розв'язанні багатьох реальних задач початкові дані є дуже наближеними, містять помилки, і отримання точного розв'язку таких задач не має ніякого змістовного наповнення. Відзначимо, що і самі математичні моделі іноді слабо відображають реальні процеси, відбиваючи лише деякі їх сторони. Розв'язання цих задач частіше переслідує одержання якісних результатів, ніж кількісних. Всі названі причини визначають важливість наближених методів і стимулюють бурхливий розвиток наближених методів дискретної оптимізації.

В наш час продовжують активно формуватися ідейні основи, засоби і методологія розробки наближених методів дискретної оптимізації. Серед них особливе місце займають методи локальної оптимізації. Це пов'язано з широтою сфери їх застосування і успішним розв'язанням з їх допомогою багаточисельних прикладних задач (розпізнавання образів, класифікації

об'єктів, розміщення, проектування, планування, управління процесами в реальному масштабі часу та ін.). На практиці найефективнішими виявляються алгоритми, в яких поєднуються різні ідеї, що розвиваються в рамках локальної оптимізації.

У двох статтях, що з'явилися наприкінці 50-х років XX ст., запропоновано алгоритм локального пошуку розв'язків задачі про комівояжера [63, 71]. Потім були досліджені [127, 146] різні околи задачі про комівояжера і введена стратегія багатократного старту локального пошуку (тим самим зроблено перший крок до створення метаевристик). В наступні роки локальний пошук також використовувався для розв'язання ряду задач, а його концепція була узагальнена різними способами. Зокрема, складний метод побудови околів (метод змінної глибини) успішно застосовувався до задач розбиття графа [121] і задачі про комівояжера [128]. З існуючих методів локальної оптимізації відзначимо методи вектора спаду [22, 27, 25], направляючих околів [17], змінної глибини [19]. Докладний опис методів локальної оптимізації можна знайти в працях [27, 23, 26].

Процес локального пошуку розв'язків дискретної оптимізаційної задачі, визначеної на обмеженій множині G, у загальних рисах, здійснюється таким чином. Нехай задана початкова точка $x_0 \in G$. Деяким чином задамо її окіл $N(x_0)$. Якщо в цьому околі не існує допустимої точки, "кращої" за початкову, то локальний оптимум знайдено і пошук закінчуємо. У протилежному випадку вибираємо нову точку $x_1 \in N(x_0) \cap G$, що "покращує" значення цільової функції, і визначаємо її окіл $N(x_1)$. Далі обчислювальний процес повторюємо до знаходження локального оптимуму.

При організації процедури локального пошуку виникають такі основні питання: вибір початкового наближення, вибір околу, встановлення правил перебору точок в околі і переходу в інші точки. Важливою характеристикою процесу локального пошуку є кількість кроків, необхідних для досягнення

локального оптимуму. У праці [117] це питання досліджено з позицій теорії складності і визначено клас складності *PLS* (поліноміальний локальний пошук).

Відзначимо, що методи локального пошуку, в яких перебір всієї множини розв'язків оптимізаційної задачі замінюється породженням хороших локальних поліпшень, що становлять тільки малу підмножину множини розв'язків, дають можливість у певних випадках зводити оптимізаційну задачу до поліноміально розв'язуваної. Вони є дуже ефективним засобом для досягнення цієї мети. Проте таку ефективність локальних методів важко повністю пояснити. Спроба зробити це була в книзі [96], де ефективність локального пошуку пояснюється принципом подібних оптимумів (ПО-принцип). Зміст даного принципу полягає в припущенні, що хороші розв'язки мають подібні структури. Поняття структури системи оптимуму не визначено точно – воно розуміється на інтуїтивному рівні, хоча і є ключем до успішного застосування локального пошуку, яке передбачає знаходження цих „прихованих структур", спільних для хороших розв'язків. При цьому важливо так визначити околи локального пошуку, щоб ці структури зберігалися при здійсненні переходів у околах. У праці [127], наприклад, відзначено, що число ребер, спільних у будь-яких двох локально оптимальних розв'язках відносно околу Ліна-Кернігана (*Lin-Kernighan*), складає в середньому приблизно 85%. В [42, 43] був запропонований метод глобального рівноважного пошуку, орієнтований на застосування до розв'язання задач, для яких характерна система подібних оптимумів. Взагалі, значний накопичений обчислювальний досвід у галузі локальної оптимізації підтверджує висунуту в [96] концепцію, яка дає інтуїтивне обгрунтування успішному застосуванню алгоритмів локального пошуку.

Далі ретельно розглянемо складові компоненти будь-якого дієвого алгоритму локального пошуку, а саме: систему околів, цільову функцію, стратегію переміщень, стратегію побудови початкових розв'язків.

Система околів. Поняття системи околів тісно пов'язане з поняттям "природного збурення" допустимого розв'язку. При виборі певної системи околів важливо зберегти структуру околів, загальну для хороших розв'язків. З цією метою доцільно розглядати кілька околів різних розмірів і структури. Наведемо деякі приклади.

Для багатовимірної задачі про ранець з булевими змінними в [27] розглядаються системи околів точки $x \in G$ радіуса r вигляду

$$O(x, r) = \{y \in G: \rho(x, y) \le r\}, \; r = 1, 2, ..., n,$$

де $\rho(x,y) = \sum_{j=1}^{n} |y_j - x_j|$ – відстань між n-вимірними векторами x, y, координати яких дорівнюють 0 або 1; G – множина допустимих розв'язків задачі.

У праці [127] для задачі про комівояжера запропоновано такі околи: перестановка міст, вставка міста, вставка підшляху, k-заміна. Окіл "перестановка міст" складається з розв'язків, які можна отримати з початкового розв'язку, змінюючи в перестановці положення двох довільних міст. Окіл "вставка міста" містить розв'язки, одержані з початкового шляхом зміни положення довільного міста в перестановці. Окіл "вставка підшляху" складається з розв'язків, які можна отримати з початкового розв'язку, видаляючи підшлях максимальної довжини (що дорівнює трьом) і вставляючи його в інше місце в перестановці. Окіл "k-заміна" містить розв'язки, одержувані з початкового розв'язку видаленням k ребер шляху і заміною їх новими k ребрами. Очевидно, що окіл "вставка міста" є підмножиною околу "вставка підшляху", а остання – підмножиною околу "3-заміна".

Проведені обчислювальні експерименти продемонстрували так звану "силу" околів "вставка підшляху", "2, 3-заміна", з одного боку, і "слабкість" околу "вставка міста" – з другого. Цей факт, виходячи з ПО-принципу, можна пояснити тим, що цільова функція в задачі комівояжера дорівнює сумі ребер,

які належать до шляху. В зв'язку з цим кількість різних ребер у будь-яких двох шляхах може служити прийнятною оцінкою відстані між ними. Відстань між поточним шляхом і довільним відмінним від нього шляхом в околі "перестановка міст" дорівнює чотирьом, в околах "вставка міста" і "вставка підшляху" – трьом, а в околі "k-заміна" – k. Це означає, що розв'язки в околі "перестановка міст" не достатньо близькі до поточного розв'язку порівняно з іншими околами.

Вибір розміру околу також виявляється дуже важливим. З накопиченого значного обчислювального досвіду випливає, що якість локально оптимальних розв'язків звичайно поліпшується з використанням більших околів. Проте час, необхідний для дослідження такого околу, природно також збільшується. Як показано в [27], якість локального оптимуму, одержуваного із застосуванням околу $O(x, 2)$, набагато краща, ніж з використанням околу $O(x, 1)$. Однак, хоча якість локального оптимуму в околі $O(x, 3)$ і є трохи кращою, ніж при використанні околу $O(x, 2)$, але таке незначне поліпшення не є суттєвим і не виправдовує додаткових витрат машинного часу. У праці [127] аналогічні результати отримано для околів "2, 3, 4-заміна". Таким чином, розмір околів необхідно утримувати всередині певних меж.

При використанні методу змінної глибини [121] застосовується спосіб побудови кроків локального пошуку, що дає можливість не обов'язково проглядати весь окіл, а обмежуватися деякими простими кроками в малих околах, на основі яких і виконувати комплексні кроки у великих околах. Цей метод був успішно використаний для задач розбиття графів [121], а потім – для задачі комівояжера [128]. Подібна, але більш загальна, ідея застосовується і в рамках методу табу до різних задач [96, 168]. В працях [19, 116] вказується на необхідність ретельного вибору простих кроків, що потім використовуються для створення комплексних кроків з метою максимального врахування структури розв'язуваної задачі і з тим, щоб не допустити експоненціального зростання об'єму околів.

Важливим уявляється питання щодо порядку перегляду точок в околі. Найпростіше використовувати природний лексикографічний порядок, який визначається нумерацією координат вектора розв'язку. Можливо також упорядкування розв'язків в околі випадковим чином. Такий спосіб має ту перевагу, що навіть при старті з одної початкової точки і застосовуючи спосіб першого поліпшення, можна знайти цілий набір випадково розподілених локальних оптимумів. Цей підхід може бути корисним у випадку, коли важко одержувати початкові допустимі розв'язки. Упорядковувати розв'язки в околі можна, виходячи також з деякої інформації про задачу, отриману, наприклад, в процесі розв'язання відповідних прямої або двоїстої неперервних задач [169]. При цьому слід врахувати, що околи близьких розв'язків мають непорожній перетин, і тому необхідно вжити заходів, які дозволять уникнути багатократного перегляду одних і тих же точок [27].

Цільова функція. При розв'язанні задачі дискретної оптимізації загального вигляду доцільно в деяких випадках використовувати не початкову цільову функцію $f(x)$, а модифіковану функцію $g(x)$, наприклад штрафну функцію

$$g(x) = \begin{cases} f(x), & \text{якщо } x \in G, \\ f(x) + M & - \text{ у протилежному випадку,} \end{cases}$$

де M – досить велике число. Така модифікована функція надає можливість адаптивно керувати процесом локального пошуку, але буває досить складною [168].

За рахунок модифікації цільової функції $f(x)$ інколи можна збільшити "силу" околу [52]. При цьому використовується модифікована функція $g(x)$, яка має таку властивість: якщо розв'язок $y \in G$ є локальним мінімумом функції $g(x)$, то він є локальним мінімумом і функції $f(x)$ [24].

Можливе також використання множини допоміжних цільових функцій, яка визначається, виходячи з передісторії оптимізаційного процесу, і враховує інформацію, накопичену про задачу [96, 148, 161].

Важливо також підкреслити, що на кожному кроці процесу локального пошуку з необхідністю використовуються не числові значення цільової функції в поточному і попередньому досліджуваних розв'язках, а лише знак їх різниці, який, можливо, і визначається набагато простіше, ніж ці значення. Тому при застосуванні деяких методів, наприклад методу вектора спаду [27], доцільно "працювати" не з самою цільовою функцією, а із спеціально побудованими різницями її значень у різних точках.

Стратегія переміщення. Процедура пошуку наступного розв'язку в поточному околі обов'язково пов'язана з встановленням деякого критерія вибору такого розв'язку і може полягати як у пошуку першого поліпшення, коли відразу, без подальшого пошуку, вибирається розв'язок, що відповідає меншому значенню цільової функції, або бути процедурою найшвидшого спуску, коли перевіряється весь окіл і вибирається розв'язок, який доставляє найменше значення цільовій функції. Проте додаткові витрати часу при реалізації найшвидшого спуску часто бувають невиправданими, але цей факт не можна узагальнювати. Значна перевага способу вибору першого поліпшення полягає в тому, що тільки в останньому околі необхідно проводити повний перебір точок, за рахунок чого локальний оптимум можна в більшості випадків знайти швидше.

Стратегія побудови початкових розв'язків. Вибір деякого розміру околів, які використовуються на кожному кроці алгоритму локального пошуку, обумовлює певну "силу" околів, тобто локальний оптимум, знайдений за допомогою локального пошуку в околах певного розміру, має обумовлену середню якість. Ця "сила", мабуть, багато в чому пов'язана з існуючою кореляцією між якістю початкових допустимих розв'язків і отриманого локального оптимуму. В багатьох випадках "сильні" околи приводять до локального оптимуму, якість якого дуже слабо залежить від початкових розв'язків, а "слабі" околи – до локального оптимуму, якість якого значною мірою пов'язана з тим чи іншим вибором початкових розв'язків. Тим самим,

сила околів визначає необхідність використання спеціально підготовлених розв'язків (наприклад, за допомогою розв'язання відповідних неперервних задач, використання допоміжних алгоритмів [27, 64, 96, 115] або зовсім випадкових початкових розв'язків).

Далі зупинимося на деяких теоретичних результатах, що стосуються локального пошуку.

Широке застосування методів локальної оптимізації стимулювало проведення досліджень в галузі опуклого аналізу, пов'язаних з побудовою моделей опуклості на дискретних множинах. Була розглянута проблема про виділення класів задач дискретної оптимізації, цільові функції яких не мають локальних екстремумів, відмінних від глобальних. У праці [27] описана єдина схема, що дозволяє певним чином узагальнити відомі моделі опуклості, запропоновані різними авторами з метою дослідження тих або інших задач дискретної оптимізації. Для дискретної множини достатньо загальної структури були визначені поняття, аналогічні загальновідомим поняттям опуклих (квазіопуклих) множин і функцій. Запропоновано таке розширення цих понять, яке залишає в силі твердження про збіг локального і глобального мінімумів для задач опуклої оптимізації. Досліджені також питання про збіг різних моделей опуклості.

Звичайно, обчислювальний час, необхідний для перегляду околу поточного розв'язку, обмежений поліномом. Проте число кроків, необхідних для досягнення локального оптимуму, виходячи з деякого початкового розв'язку, не завжди може бути обмежено поліномом. Зокрема, побудовано приклад задачі про комівояжера і її початкові розв'язки, для яких локальний пошук з околом "2-заміна" здійснюється за $O(2^{n/2})$ кроків до попадання в локальний мінімум. Подібні результати отримані і для околу "3-заміна" [68] та узагальнені в [116] для задачі про комівояжера. У [117] з метою аналізу складності знаходження локального оптимуму введено поняття класу складності

PLS (поліноміальний локальний пошук), в якому кожна задача локального пошуку представлена як пара, що складається із задачі оптимізації і вибраного околу. Багато важливих задач локального пошуку є повними для класу *PLS*, і це означає, що час, необхідний для пошуку локального оптимуму будь-яким методом локального типу не є поліноміально обмеженим, якщо тільки не всі *PLS*-задачі можуть бути розв'язані за поліноміальний час. Наприклад, локальний пошук розв'язку задачі про комівояжера з околом "*k*-заміна" для досить великих k є *PLS*-повною задачею. Достатньо повний огляд з теорії *PLS*-повноти наведений у [170].

Незважаючи на ці зовсім не обнадійливі теоретичні результати, на практиці час, необхідний для досягнення локального оптимуму, звичайно дуже малий. Здійснено дослідження, в результаті яких отримані теоретичні оцінки середнього числа кроків до знаходження локального оптимуму. Для двовимірної випадково згенерованої евклідової задачі про комівояжера, наприклад, показано [68], що при використанні околу "2-заміна" це число обмежено величиною $O(n^{10} log\ n)$. Проте з експериментальних досліджень [116] випливає, що середнє число таких кроків обмежено лише величиною $O(n\ log\ n)$, дуже далекою від теоретичної межі.

Завершуючи короткий огляд проблем і досягнень, що стосуються розробки алгоритмів локальної оптимізації, важливо підкреслити, що хоча локальний пошук є одним із найстаріших підходів, створених для розв'язання дискретних задач оптимізації; проте і нині цю проблематику продовжують опрацьовувати багато учених. Стимулом для подальшого розвитку теорії та практики локального пошуку є "успіхи" метаевристичних алгоритмів, основаних на ідеях локального пошуку. Деяким із них, які можна вважати подальшим розвитком методів локального пошуку, і присвячений даний розділ.

Звичайно при застосуванні традиційних методів локального пошуку для розв'язання задач дискретної оптимізації досліджується окіл поточного розв'язку і як новий поточний

вибирається тільки той розв'язок, що строго зменшує значення цільової функції. У 80-х роках XX ст. з'явилися нові перспективні підходи до розв'язання складних задач дискретного програмування, які використовують локальний пошук, і були запропоновані алгоритми, в яких допустимими вважалися переходи до розв'язків з більшим (гіршим), у порівнянні з поточним, значенням цільової функції. Крім того, були дозволені також переходи в недопустимі розв'язки, тобто розв'язки, що не належать множині G. За рахунок таких переходів ці алгоритми, зокрема, відпалу [65, 67, 122] і табу [92-94], можуть уникати зупинки в локальних мінімумах низької якості. Локальний пошук також був поєднаний з іншими евристичними алгоритмами, наприклад, з генетичним, мурашиних колоній [76, 77]. Запропоновані підходи, які були названі метаевристиками, успішно застосовуються до розв'язання багатьох важливих задач дискретної оптимізації. Метаевристики займають значну частину існуючих евристичних підходів, і в більшості з них локальний пошук є центральною компонентою. Існує безліч публікацій, наприклад [51, 90, 91, 139-143, 145], стосовно теорії та практичного застосування метаевристичних алгоритмів.

Згідно з працею [137] сформулюємо умови, яким задовольняють метаевристики та алгоритми, побудовані на їх основі.

1. Простота. Метаевристика грунтується на простому і зрозумілому принципі, який може широко використовуватися у різних застосуваннях.

2. Узгодженість. Всі кроки в алгоритмі для розв'язання певної проблеми виходять із заявленого метаевристичного принципу.

3. Сила. Алгоритм розв'язання певної проблеми знаходить оптимальні або майже оптимальні розв'язки для всіх або принаймні для більшості задач. Бажано, щоб алгоритм знаходив оптимальні розв'язки для більшості тестових прикладів.

4. Ефективність. Алгоритм дуже швидко знаходить хороші розв'язки, тобто об'єм обчислень обмежено поліномом не вище другого ступеня від розмірів задачі.

5. Робастність. Хороші розв'язки алгоритм знаходить без попереднього настроювання і на всій множині задач, яка цікавить дослідника.

6. Зручність. Алгоритм є простим для розуміння та використання і має незначну кількість параметрів, а в ідеалі взагалі їх не має.

7. Інновація. Метаевристичний принцип та (або) сила і ефективність побудованих на його основі алгоритмів приводять до нових типів застосувань.

На нашу думку, ці умови необхідно доповнити такими.

8. Стратегія. Під час розв'язання задачі алгоритм самостійно вибирає стратегію пошуку, періодично переходячи від режиму інтенсифікації пошуку до режиму диверсифікації пошуку і навпаки.

9. Інформація. При пошуку чергового розв'язку і виборі відповідної стратегії на кожному кроці алгоритму використовується інформація про задачу, отримана на попередніх етапах, але запам'ятовується тільки корисна інформація і не допускається накопичення надмірної і, можливо, шкідливої інформації.

10. Рандомізація. Алгоритм є максимально рандомізованим, переходи між кроками мають випадковий характер.

11. Евристика. В алгоритмі поєднані різні евристики, тобто створена "команда" алгоритмів, яка дозволяє якнайкращим способом використовувати переваги алгоритмів, що входять до неї.

У праці [122], виходячи з аналогії між складними задачами дискретної оптимізації і макроскопічними рівноважними системами, які досліджуються в статистичній фізиці, був запропонований новий локальний метод дискретної оптимізації, в якому передбачено можливість імовірнісних переходів від поточних розв'язків до розв'язків з більшим ("гіршим") значенням цільової функції. Дозволивши переходи до розв'язків з непокращеним значенням цільової функції, метод відпалу стимулював нові ідеї в розробці ефективних алгоритмів дискретної

оптимізації. Крім імовірнісних варіантів методу відпалу, в [69, 79] запропоновано його детерміновані варіанти.

Відомий метод табу вперше був описаний Гловером у працях [92, 93] як детермінований варіант методу відпалу, в якому пам'ять, а не імовірність, грає провідну роль у переходах від розв'язку до розв'язку. Щоб уникнути зациклення, проміжні розв'язки явно або неявно запам'ятовуються, і в алгоритмі впродовж певної кількості ітерацій забороняються переходи в ці розв'язки. Такий механізм, реалізований за допомогою списку заборонених (табу) кроків, дає можливість не тільки перешкоджати зацикленню протягом деякого часу, але і розширювати область дослідження. В цьому і полягає основна ідея методу табу. Введення ряду важливих понять (довгострокової пам'яті, інтенсифікації і стратегічного коливання [96]) сприяло широкому розповсюдженню і успішному застосуванню цієї метаевристики.

3.1. Метод вектора спаду

Розглянемо таку задачу дискретної оптимізації:

$$\min_{x \in G} f(x), \qquad (3.1)$$

де G – множина (скінченна або зчисленна) допустимих розв'язків задачі; $f(x)$ – визначена на ній цільова функція. Передбачається, що на множині G задана система околів N, тобто кожному допустимому розв'язку $x \in G$ поставлена у відповідність множина $N(x) \subset N \subset G$, причому $x \in N(x)$.

Означення 3.1. Точку $x \in G$ назвемо точкою локального мінімуму функції $f(x)$ відносно околу $N(x)$, якщо цей окіл не вироджений ($N(x)\backslash\{x\}$ – непорожня множина) і для будь-якої його точки y виконується умова $f(x) \le f(y)$.

Всі відомі методи локальної оптимізації основані на єдиному підході до пошуку розв'язків задачі, який полягає в поліпшенні отриманого розв'язку шляхом пошуку серед тих елементів множини G, що належать деякому околу цього

розв'язку. Загальна схема методів локального пошуку може бути подана у вигляді таких процедур:

procedure локальний_пошук
[початкова_установка (s, x^s)
while пошук_покращення $(x^s) \neq$ "ні" *do*
[x^{s+1} = пошук_покращення(x^s)
$s = s + 1$
comment знайдено локальний мінімум x^s.

Процедури початкова установка (s, x^s) і пошук покращення (x^s) визначаються таким чином:

procedure початкова_установка (s, x^s)
[кількість ітерацій $s = 0$
вибір деякої початкової точки $x^0 \in G$;

пошук_покращення $(x^s) =$ { будь-якому розв'язку $y \in N(x^s)$: $f(y) < f(x^s)$, якщо такий розв'язок існує; "ні" – у протилежному випадку.

До групи методів, що використовують процедури локального пошуку, належать, зокрема, методи вектора спаду [27], відпалу [122], табу [92, 93], глобального рівноважного пошуку [42]. Загальною проблемою для цих методів є вибір ефективної стратегії пошуку, правильного поєднання процесів розширення і звуження області пошуку з метою отримання глобального оптимуму. Важливо, щоб після отримання деякого локального оптимуму можна було здійснити перехід в область тяжіння іншого локального оптимуму, для чого необхідно вий-

ти з околу знайденого локального оптимуму, тим самим розширивши область пошуку (так звана диверсифікація пошуку). З іншого боку, для знаходження поліпшеного розв'язку логічно дослідити точки, близькі до отриманого локального оптимуму, тобто звузити область пошуку (інтенсифікація пошуку). Знаходження необхідного балансу між розширенням і звуженням області пошуку розв'язку є найважливішою проблемою для локальних методів.

Схема методу вектора спаду дозволяє створювати цілий спектр алгоритмів для розв'язання конкретних задач вигляду (3.1) і після проведення відповідного аналізу вибирати найбільш придатний із них.

Введемо поняття вектора спаду значень цільової функції f в околі деякої точки $x \in G$. Нехай для $k = 1, ..., r$, $r \geq 1$, задані системи околів $N^k(x)$, такі, що $\forall\ x \in G: N^j(x) \subset N^i(x)$ при $j < i$.

Означення 3.2. Векторна функція $\Delta^k(x)$, визначена для кожної точки $x \in G$ і її околу $N^k(x)$, $k = 1, ..., r$, називається вектором спаду функції f відносно околу $N^k(x)$, якщо виконуються умови:

— значення функції $\Delta^k(x)$ в будь-якій точці $x \in G$ є q-вимірним ($q = q(k,x) \leq |N^k(x)|$) вектором з координатами $\Delta_1^k(x), ..., \Delta_q^k(x)$ – дійсними числами;

— деяка точка $x \in G$ є точкою локального мінімуму функції f відносно околу $N^k(x)$ тоді і тільки тоді, коли $\Delta_i^k(x) \geq 0$ при всіх $i = 1, ..., q$;

— якщо деяка точка $x \in G$ не є точкою локального мінімуму функції f відносно околу $N^k(x)$, то за допомогою вектора спаду можна визначити точку $y \in N^k(x)$, таку, що $f(y) < f(x)$.

Виходячи з даного означення, робимо висновок, що за допомогою вектора $\Delta^k(x)$ можна для кожної точки $x \in G$ визначити напрями зменшення ("спаду") значень функції f в

околі $N^k(x)$. Ідея використання вектора спаду $\Delta^k(x)$ для відшукання таких напрямів і лежить в основі одноіменного методу. Слід зазначити, що часто координати $\Delta_1^k(x), ..., \Delta_q^k(x)$ вектора спаду можна обчислити набагато простіше, ніж самі значення функції f в точках околу $N^k(x)$. Крім того, в більшості випадків для виявлення в $N^k(x)$ точки y, такої, що $f(y) < f(x)$, або для встановлення факту локальної мінімальності значення $f(x)$ достатньо обмежитися розглядом лише частини координат вектора $\Delta^k(x)$.

Подамо загальну схему методу вектора спаду таким чином:

procedure вектор спаду
початкова_установка$(s, x^s, s_{\max}, r)$
while $(s < s_{\max})$ *do*
  $k = 1$
  while $(k \leq r)$ *do*
    if (пошук_покращення$(x^s, \Delta^k(x^s))$ = "ні" *then*
      $k = k + 1$
    else
      x^{s+1} = пошук_покращення$(x^s, \Delta^k(x^s))$
      $s = s + 1$
      $k = 1$
  x^{s+1} = нова_точка(x^s)
  $s = s + 1$
  comment новий старт локального пошуку.

Процедури "початкова_установка", "пошук_покращення" і "нова_точка" визначаються таким чином:

procedure початкова_установка $(s, x^s, s_{\max}, r)$

$$\begin{cases} \text{кількість ітерацій} & s = 0, \\ \text{вибір деякої початкової точки} & x^0 \in G, \\ \text{задання максимального числа} & s_{\max} \text{ ітерацій і} \\ \text{максимального номера} & r \text{ околу;} \end{cases}$$

пошук_покращення $(x^s, \Delta^k(x^s))$ =

$$= \begin{cases} \text{будь-якому розв'язку } y \in N^k(x^s): f(y) < f(x^s), \\ \text{якщо } \exists i : \Delta_i^k(x^s) < 0, \ 1 \le i \le q(k, x^s); \\ \text{"ні" – у протилежному випадку;} \end{cases}$$

нова_точка (x^s) =

$$= \begin{cases} \text{будь-якому розв'язку } y \in N^k(x^s) : f(y) < f(x^s), \\ \text{якщо } r < k \le R, \exists i : \Delta_i^k(x^s) < 0, \ 1 \le i \le q(k, x^s); \\ \text{будь-якому розв'язку } y \in G, \text{ вибраному випадковим чином, –} \\ \text{у протилежному випадку.} \end{cases}$$

Відзначимо, що в методі вектора спаду питання про диверсифікацію і інтенсифікацію локального пошуку розв'язуються таким чином: після відшукання точки локального мінімуму розглядається окіл з номером, більшим за максимальний (інтенсифікація), або здійснюється перехід у нову початкову точку (диверсифікація).

На основі наведеної вище загальної схеми методу вектора спаду можна побудувати такі алгоритми.

Алгоритм I

Вибираємо деяке початкове наближення $x^0 \in G$ (випадково або з урахуванням особливостей розв'язуваної задачі) і задаємо радіус $r > 0$. Покладаємо $k = 0$.

На $(k + 1)$-у кроці алгоритму виконуємо такі дії.

1. Розглядаємо окіл $N^r(x^k)$. Якщо він вироджений, то для застосування алгоритму необхідно або збільшити радіус r, або вибрати інше початкове наближення x^0. Якщо окіл $N^r(x^k)$ не вироджений, переходимо до п.2 алгоритму.

2. За значеннями координат вектора спаду $\Delta^r(x^k)$ визначаємо, чи є x^k точкою локального мінімуму функції $f(x)$ щодо околу радіуса r. Якщо так, то робота за алгоритмом закінчується, інакше – до п.3.

3. За значеннями координат вектора спаду знаходимо точку $x^{k+1} \in N^r(x^k)$, яка задовольняє умову $f(x^{k+1}) < f(x^k)$. Замінюючи k на k+1, переходимо до п.1.

Алгоритм II

Вибираємо початкове наближення $x^0 \in G$, задаємо радіус $r > 0$ і послідовність радіусів $r_1, ..., r_m$, таких, що $0 < r_1 < ... < r_m = r$, $m \ge 1$. Покладаємо $k = 0$.

На $(k + 1)$-у кроці алгоритму виконуємо такі дії.

1. Покладаємо $j = 1$.

2. Якщо окіл $N^{r_j}(x^k)$, що розглядається, є виродженим, то при $j = m$ для подальшого застосування даного алгоритму необхідно збільшити радіус r або вибрати інше початкове наближення. При $j < m$ замінюємо j на $j + 1$ і знову виконуємо даний пункт алгоритму.

3. У випадку, коли окіл $N^{r_j}(x^k)$ невироджений, за значеннями координат вектора $\Delta^{r_j}(x^k)$ з'ясовуємо, чи є значення $f(x^k)$ локальним мінімумом функції $f(x)$ щодо околу радіуса r_j. Якщо так, то при $j < m$, замінивши j на $j + 1$, переходимо до початку п. 3, а при $j = m$ робота за алгоритмом закінчується, оскільки x^k є шуканою точкою локального мінімуму функції $f(x)$ в околі радіуса r.

4. Якщо ж x^k не є точкою локального мінімуму, то за значеннями координат вектора спаду $\Delta^{r_j}(x^k)$ знаходимо в множині $N^{r_j}(x^k)$ точку x^{k+1}, для якої $f(x^{k+1}) < f(x^k)$. Переходимо до п.1, змінюючи k на k +1.

Суть методу вектора спаду полягає саме в раціональному поєднанні першого і другого алгоритмів при організації пошуку локального мінімуму функції $f(x)$.

Зробимо три зауваження, що стосуються відмінностей двох наведених вище алгоритмів.

По-перше, необхідність дослідження околів максимального радіуса r_m виникає лише для незначної частини кроків алгоритму II. Для більшості кроків виявляється достатнім обмежитися обчисленням координат вектора спаду функції $f(x)$ в околі $N^{r_1}(x^k)$. Це в багатьох випадках набагато простіше за процедуру обчислення координат вектора спаду $\Delta^r(x^k)$, якою доводиться користуватися на кожному кроці алгоритму I при $r = r_m > 1$.

По-друге, вказана вище перевага алгоритму II над алгоритмом I може бути повністю реалізована у тому випадку, коли в алгоритмі I на перехід від точки x^k до точки x^{k+1} накладено додаткову умову $f(x^{k+1}) = \min\limits_{x \in N^r(x^k)} f(x^k)$. Очевидно, що схема алгоритму II не допускає накладання такої умови на перехід від x^k до x^{k+1} як обов'язкової.

По-третє, при виконанні алгоритму II у тих випадках, коли на деякому $(k+1)$-у кроці необхідно розглянути околи з радіусами, відмінними від r_1, замість будь-якого такого околу $N^{r_j}(x^k)$, $j > 1$, достатньо розглянути множину точок $N^{r_j}(x^k) \setminus N^{r_{j-1}}(x^k)$ і дослідити лише координати вектора спаду $\Delta^{r_j}(x^k)$, що відповідають цим точкам.

Збіжність розглянутих вище обчислювальних схем методу вектора спаду випливає з такої теореми.

Теорема 3.1. *Нехай функція $f(x)$, визначена на множині G, задовольняє дві умови: вона обмежена знизу на G, тобто існує таке число $c = \text{const}$, що $f(x) \geq c$ $\forall x \in G$; існує таке число $\delta > 0$, що $\forall x', x'' \in G$, таких, що $f(x') \neq f(x'')$, виконується нерівність $|f(x') - f(x'')| \geq \delta$. Тоді при будь-якому виборі початкового наближення $x^0 \in G$, радіуса $r > 0$ і послідовності $r_1, \ldots, r_m$, де $r = r_m$, процеси обчислень по обох алгоритмах збігаються за скінченне число кроків, що не перевищує величини $|f(x^0) - c|\delta^{-1}$.*

Справедливість теореми випливає безпосередньо з правил побудови алгоритмів I і II. Дійсно, елементи послідовності $\{f(x^k) \mid k = 0, 1, \ldots\}$, одержуваної в результаті роботи будь-якого з двох алгоритмів, задовольняють співвідношення

$$f(x^0) > f(x^1) > \ldots > f(x^k) > f(x^{k+1}) > \ldots$$

З цих нерівностей і умов теореми випливає, що $f(x^0) - c \geq \alpha\delta$, де α – число кроків певного алгоритму. Таким чином, справедлива нерівність $\alpha \leq \dfrac{f(x^0) - c}{\delta}$, що і доводить теорему.

Очевидно, що для функції $f(x)$, визначеної на деякій скінченній множині, виконуються обидві умови теореми 3.1, і, отже, в цьому випадку обчислювальний процес по методу вектора спаду закінчується за скінченне число кроків. У тих випадках, коли методом вектора спаду розв'язується оптимізаційна задача, цільова функція якої не задовольняє умови теореми 3.1, до алгоритму (I чи II) можна вводити ту або іншу ознаку закінчення обчислювального процесу. Наприклад, роботу по алгоритму можна закінчити, коли число кроків перевищить деяку наперед задану величину. Крім того, на кожному k-му кроці можна оцінювати, якою мірою отримане значення $f(x^k)$ задовольняє певні практичні вимоги. Обчислювальний процес може тривати до того кроку алгоритму, на якому будуть задоволені ці вимоги щодо розв'язку задачі. Процес пошуку можна також завершити і після закінчення деякого відведеного проміжку часу t, якщо, наприклад, екстремальна задача розв'язується в деякій системі прийняття рішень, що функціонує в реальному масштабі часу. В останньому випадку, природно, точність отриманого розв'язку задачі істотно залежить від величини t.

Нехай час розв'язання екстремальної задачі не лімітується, цільова функція $f(x)$ задовольняє лише першу умову теореми 3.1, і при реалізації методу вектора спаду склалася така ситуація, що $\lim\limits_{k \to \infty} f(x^k) = \text{const}$. Тоді зупинку обчислювального процесу

можна здійснити і за таким правилом. Для досить великих значень індексу k можна всі значення

$$f(x^k), f(x^{k+1}), \ldots, \qquad (3.2)$$

які мало відрізняються один від одного, вважати "наближеними локальними мінімумами". Продовживши обчислення за алгоритмом до того кроку, коли сусідні величини в послідовності (3.2) виявляться дуже "близькими" між собою, можна зупинитися і відповідну точку x^k вважати наближеним розв'язком задачі.

З метою більш повного опису загальної схеми методу вектора спаду зупинимося на такому можливому випадку, коли отриманий у результаті застосування одного з алгоритмів локальний розв'язок x^* задачі не задовольняє деяку додаткову задану умову (наприклад, умову, що встановлює максимально допустиме відхилення локального розв'язку від глобального оптимуму). Для пошуку інших наближених розв'язків задачі в цьому випадку пропонуються такі шляхи.

1. Здійснити обчислювальний процес за алгоритмом, виходячи з отриманої точки x^* як нового початкового наближення і збільшуючи радіус r. Проте при великих значеннях радіуса r обчислювальний процес може значно ускладнитися, у зв'язку з чим для підвищення точності розв'язку задачі доцільно скористатися алгоритмом II.

2. Збільшивши радіус r, виконати лише один крок алгоритму, що використовується, для отримання точки x', для якої справджується нерівність $f(x') < f(x^*)$. Далі рекомендується продовжити обчислювальний процес по алгоритму, повернувшись до первинної величини радіуса r.

3. Вибрати деяке інше початкове наближення $x^{01} \in G$ і застосувати алгоритм (I чи II), виходячи з цієї нової початкової допустимої точки, але не змінюючи r. У загальному випадку в результаті одержимо інший локальний розв'язок задачі $x^{*1} \neq x^*$. Потім слід вибрати той із розв'язків, який доставляє мінімальне значення функції $f(x)$.

4. Якщо в околі $N'(x^k)$, де $x^k = x^*$, існує така точка x', що $f(x') \neq f(x^*)$, то роботу алгоритму можна продовжити, здійснивши перехід на наступний його крок на основі точки x', тобто вибравши $x^{k+1} = x'$. Таке доповнення до алгоритму в ряді випадків може привести до точнішого розв'язку задачі.

Відзначимо, що метод вектора спаду з успіхом застосовувався для розв'язання багатьох практичних задач [20, 27].

3.2. Метод відпалу

Розвиток ідей локального пошуку привів до створення методу відпалу, який допускає здійснювати імовірнісні переходи не тільки в "кращі" точки, а і в "гірші". Такий підхід дає можливість уникати поганих локальних оптимумів і знаходити розв'язки, близькі за значеннями цільової функції до глобального оптимуму (на практиці – одержувати глобальні розв'язки).

Подамо схему методу відпалу у вигляді таких процедур:

procedure відпал

початкова_установка(s, x^s, r, mt, M, nt)
repeat for $i = 0$ *to* mt
  $\mu = M(i)$
  repeat for $j = 1$ *to* nt
    y = випадковий_вибір(x^s, r)
    if $f(y) \le f(x^s)$ *then* $x^{s+1} = y$
    else
      ξ = випадковий_датчик()
      if $\xi \le \exp\{\mu^*(f(y) - f(x^s))\}$
      then $x^{s+1} = y$
      else $x^{s+1} = x^s$
    $s = s + 1$

procedure початкова_установка(s, x^s, r, mt, M, nt)

кількість ітерацій $s = 0$,
вибір деякої початкової точки $x^0 \in G$ і
номера r околу,
вибір величини mt і масиву $M(0:mt)$, які визначають розклад відпалу, та величини nt – числа ітерацій для одержання рівноважного розподілу;

випадковий_вибір (x^s, r) =

$$= \begin{cases} \text{довільному, випадково вибраному} \\ \text{розв'язку } y \in N^r(x^s); \end{cases}$$

випадковий_датчик ( ) =

$$= \begin{cases} \text{реалізації випадкової величини,} \\ \text{рівномірно розподіленої на відрізку } [0,1]. \end{cases}$$

Далі проведемо теоретичний аналіз методу відпалу, для чого перенумеруємо всі допустимі розв'язки задачі (3.1). Позначимо $Q = |G|$, де $|G|$ – потужність множини G, $q(x)$ – порядковий номер розв'язку $x \in G$. Виберемо числа r_{kl} ($k, l = 1, ..., Q$), так, щоб виконувалися умови

$$r_{ij} \geq 0,\ r_{ij} = r_{ji},\ (i, j = 1, ..., Q),\ \sum_{y \in N(x)} r_{q(x)q(y)} = 1,\ x \in G, \quad (3.3)$$

і використаємо ці числа в процедурі випадкового пошуку таким чином: якщо $i = q(x^s)$, то випадковий вибір розв'язку $y \in N(x)$ здійснюємо з імовірністю $r_{iq}(y)$. Очевидно, що при фіксованому значенні величини μ метод відпалу породжує марківський ланцюг $\{x^s\}_{s=0}^{\infty}$. Покладемо $p_{q(x)p(y)}(s) = P\{x^s = y \mid x^0 = z\}$.

Теорема 3.2. *Якщо числа r_{kl} ($k, l = 1, ..., Q$), такі, що марківський ланцюг $\{x^s\}_{s=0}^{\infty}$ є єдиним класом істотних станів без підкласів, то для будь-якого стану $y \in G$ існує не залежна від $z \in G$ границя*

$$\lim_{s \to \infty} p_{q(x)q(y)}(s) = \pi_{q(y)} = \frac{\exp\{-\mu f(y)\}}{\sum_{x \in G} \exp\{-\mu f(x)\}}. \quad (3.4)$$

Доведення. Існування границі (3.4) випливає з умови теореми про те, що марківський ланцюг $\{x^s\}_{s=0}^{\infty}$ є єдиним класом істотних станів без підкласів. Виходячи з поданих вище процедур методу відпалу, робимо висновок, що елементи матриці перехідних імовірностей $\{d_{ij}\}_{i,j=1}^{Q}$ марківського ланцюга $\{x^s\}_{s=0}^{\infty}$ мають такий вигляд:

$$d_{ij} = P\{x^{s+1} = y, i = q(y) \mid x^s = z, j = q(z)\} =$$

$$= \begin{cases} r_{ij}, & \text{якщо } f(y) \geq f(z), \\ r_{ij} \exp\{-\mu(f(z) - f(y))\}, & \text{якщо } f(y) < f(z). \end{cases}$$

Тепер покажемо, що величини π_i, $i = 1, ..., Q$, задовольняють систему лінійних рівнянь

$$\pi_i = \sum_{j=1}^{Q} \pi_j d_{ji}, \quad (3.5)$$

$$\sum_{i=1}^{Q} \pi_i = 1. \quad (3.6)$$

Дійсно, з (3.4) випливає співвідношення (3.6), а з (3.3) – рівність

$$\pi_i d_{ij} = \pi_j d_{ji} \quad (i, j = 1, ..., Q). \quad (3.7)$$

Тоді перетворюючи (3.5) і використовуючи (3.6), (3.7), отримаємо

$$\pi_i = \sum_{j=1}^{Q} \pi_i d_{ij} = \pi_i,\ i = 1, ..., Q,$$

що і доводить теорему.

Позначимо X^* множину оптимальних розв'язків задачі (3.1), f^* – оптимальне значення цільової функції,

$K^* = \{k : k = q(x),\ x \in X^*\}$, і припустимо, що цільова функція набуває цілочислових значень. Розглянемо імовірність того, що марківський процес збігається до множини оптимальних розв'язків:

$$\sum_{k \in K^*} \pi_k(\mu) =$$

$$= \sum_{y \in X^*} \frac{\exp\{-\mu f(y)\}}{\sum_{x \in G} \exp\{-\mu f(x)\}} = \frac{1}{|K^*| + \sum_{x \in G \setminus K^*} \exp\{-\mu(f(x) - f^*)\}} \geq$$

$$\geq \frac{1}{|K^*| + (Q - |K^*|) \exp\{-\mu\}}.$$

Нехай числа r_{kl} $(k, l = 1, ..., Q)$ і околи $N(x^k)$ допустимих розв'язків $x^k \in G$ вибрані так, що при будь-якому $\mu > 0$ виконуються умови теореми 3.2 і співвідношення (3.3). При цьому у випадку, коли $M[i] = \infty$, $i = 1, ..., mt$, алгоритм відпалу перетворюється на звичайний метод випадкового пошуку з локальною оптимізацією: в процесі пошуку здійснюються випадковий перехід тільки в допустимі точки з меншим значенням цільової функції і зупинка в поглинаючому стані, тобто в локальному оптимумі. Якщо ж $M[i] = 0$, $i = 1, ..., mt$, то алгоритм відпалу виконує випадкове блукання по множині допустимих розв'язків G. В останньому випадку $\pi_l = Q^{-1}$, $l = 1,...,Q$. Передбачається, що якщо задати відповідним чином величини $M[i]$, вдасться в результаті роботи алгоритму отримати допустимий розв'язок, близький по цільовій функції до оптимального розв'язку задачі (3.1). Таке припущення основане на таких міркуваннях.

Імовірнісний розподіл, який подано формулою (3.4), відомий як розподіл Больцмана і має для статистичної фізики фундаментальне значення. Для макроскопічної системи, що знаходиться в тепловій рівновазі з тепловим резервуаром з абсолютною температурою T, імовірність перебування в стані S з енергією E_s задається формулою $P\{s\} = \exp\{-E_s/kT\}/Z$, де k –

стала Больцмана, $Z = \sum_s \exp\{-E_s / kT\}$ – статистична сума, в якій сумування здійснюється по всіх допустимих станах системи. При зниженні абсолютної температури теплового резервуару макроскопічна система еволюціонує до станів з нижчою енергією, а при T=0° К вона знаходиться в станах з мінімальною енергією. Якщо задача (3.1) має велику розмірність, то за певних умов можна знайти глибокі аналогії між поведінкою марківського ланцюга $\{y_t(\mu)\}_{t=0}^{\infty}$ і еволюцією макроскопічної системи при температурі $T = (k\mu)^{-1}$, що прямує до нуля.

Необхідно відзначити, що, на жаль, розклади відпалу $M[i]$, $i = 1, ..., mt$, які забезпечують збіжність методу до глобального оптимуму, на практиці застосовуватися не можуть, оскільки вимагають величезних витрат машинного часу. Крім того, встановлено [82], що асимптотична ефективність методу відпалу гірша, ніж методу вектора спаду. Переваги методу відпалу полягають не тільки в його практичній значимості, а і в тому, що він стимулював нові наукові ідеї, наприклад, щодо створення методу глобального рівноважного пошуку [42].

3.3. Метод табу

Наприкінці 80-х років XX ст. був запропонований метод табу, який висунув нові високі вимоги до пам'яті і швидкодії комп'ютерів. Зокрема, в описаній тут версії цього методу необхідно запам'ятовувати всі точки, розглянуті під час локального пошуку. Основною ідеєю методу табу є заборона деяких використаних переходів з тим, щоб уникнути попадання в точки, що вже були розглянуті.

В останнє десятиріччя відбувся могутній розвиток обчислювальної техніки, який дозволив у багато разів збільшити обчислювальну швидкість і доступну пам'ять сучасних комп'ютерів. Якщо зовсім недавно за прийнятний час можна було отримати одиниці, максимум – десятки локальних оптимумів і тому не виникало потреби особливо замислюватися над за-

гальною стратегією пошуку глобального розв'язку, то тепер ситуація суттєво змінилася.

Принципи, покладені в основу методу табу, можна знайти і в методах відсікання, гілок і границь, і в різних методах локального пошуку. Розвиток алгоритмів, що використовують стратегії заборони, пов'язаний з ім'ям Гловера [92–97]. Через різноманітність і велику кількість алгоритмів, які базуються на методі табу, важко навести його канонічну форму. Проте можна виділити дві основні риси, властиві більшості з цих алгоритмів: 1) застосування локального пошуку і 2) використання заборони деяких переходів, можливих у поточній точці. Нагадаємо, що локальний пошук є ефективним, якщо структура околів, що розглядаються, відповідає структурі розв'язків оптимізаційної проблеми. Але звичайний локальний пошук припиняється при першому ж попаданні в точку локального мінімуму, коли стають недоступними поліпшуючі кроки. Алгоритми, основані на методі табу, дають можливість продовжити пошук, не втрачаючи результатів виконаної роботи, як це відбувається, наприклад, при повторному старті локального пошуку з нової початкової точки.

Основною перевагою методу табу перед іншими методами, які використовують процедури локального пошуку, вважається вдале використання інформації, отриманої в процесі пошуку, що дозволяє попереднім етапам пошуку впливати на його подальші етапи. Такої можливості не передбачено, наприклад, у методі відпалу, оскільки марковість породжуваного ним процесу означає саме незалежність майбутнього від минулого. Існуюча в методі табу така залежність дає можливість одержувати кращі результати, але, з іншого боку, значно ускладнює його теоретичний аналіз.

Опишемо різновиди методу табу. Для цього додатково введемо деякі поняття. Позначимо BM множину базових ходів. Вважатимемо, що окіл $N(x)$ будь-якої точки x можна побудувати, виходячи з точки x, за допомогою застосування до неї базових ходів, тобто $N(x) = \{y : y = m \circ x,\ m \in BM\}$. Наприклад,

якщо $x = (x_1,...,x_n)$ – вектор з булевими координатами, то можна вважати, що базовий хід m_i, $i = 1,...,n$, полягає в застосуванні операції заперечення до значення координати x_i. Зрозуміло, що окіл $N^1(x)$ радіуса $r = 1$ складається з точки x і тих точок, які отримані застосуванням до x базових ходів m_i, $i = 1,...,n$. В процесі роботи за методом табу можуть бути заборонені окремі ходи. Нехай AM – множина, що складається з допустимих ходів. Тоді допустимий для точки x окіл $AN(x)$ може бути побудовано за допомогою застосування до неї допустимих ходів: $AN(x) = \{y : y = m \circ x,\ m \in AM\}$. Вкажемо спосіб побудови траєкторії пошуку в методі табу. Якщо x^{k+1} – точка, отримана на $(k+1)$-й ітерації, $C(\ )$ – задана процедура вибору точки з допустимого околу, то $x^{k+1} = C(AN(x^k \mid x^0,...,x^{k-1}))$, де $AN(x^k \mid x^0,..., x^{k-1}) = \{y : y = m \circ x^k,\ m \in AM(x^0, ..., x^{k-1})\}$. Процедура $C(\ )$ вибору наступної точки може бути детермінованою чи стохастичною і враховувати специфіку задачі. Звичайно, здійснюється вибір $BC()$ найкращої точки, тобто $f(BC(AN(x))) \le f(y)\ \forall y \in AN(x)$.

Виділимо три основні різновиди методу табу, які характеризуються відмінними правилами заборони (табу) на можливі переходи: строге табуювання, фіксоване табуювання, реактивне табуювання. При застосуванні строгого табу забороняються переходи в ті точки, які вже раніше розглядалися –

$$AN(x^k \mid x^0,...,x^{k-1}) = \{y : y = m \circ x^k,\ y \ne x^0,\ ...,\ x^{k-1},\ m \in BM\}.$$

При фіксованому табу задається фіксована величина t і забороняються переходи, які використовувалися на останніх t $(t < k)$ ітераціях, тобто

$$AN(x^k \mid x^0,\ ...,\ x^{k-1}) = \{y : y = m \circ x^k,\ m \in BM \setminus \{m_j\},\ j = k - t,...,k - 1\},$$

де m_j – хід, зроблений на j-й ітерації. При застосуванні реактивного (адаптивного) табу величина t змінюється в процесі пошуку і залежить від досягнутих результатів.

Найпростіша схема методу табу (різновид строгого табу) може бути подана за допомогою таких процедур:

procedure табу

початкова_установка($s, x^s, s_{\max}, r, T^s$)

while $s < s_{\max}$ do

x^{s+1} = пошук_найкращого_розв'язку (x^s, T^s, r)

$T^{s+1} = T^s \cup \{x^{s+1}\}$

$s = s + 1$;

procedure початкова_установка($s, x^s, s_{\max}, r, T^s$)

кількість ітерацій $s = 0$,

задання максимального числа ітерацій $s_{\max}$,

вибір деякої початкової точки $x^0 \in G$,

задання номера r околу,

визначення множини $T^0 = \{x^0\}$ заборонених точок;

пошук_найкращого_розв'язку $((x^s, T^s, r) =$

$=$ argmin $\{f(x): x \in AN^r(x^s)\}$, де $AN^r(x^s) = \{x: x \in N^r(x^s) \backslash T^s\}$.

Про величезний інтерес до методу табу свідчить постійне зростання кількості присвячених йому публікацій. Схема цього методу є досить гнучкою і дозволяє створювати різні алгоритми, які з успіхом застосовуються при розв'язанні різноманітних задач дискретної оптимізації. Проте, незважаючи на очевидні переваги методу табу, в ньому не повною мірою використовуються поточні результати, одержані при розв'язанні задачі. Вони враховуються лише для того, щоб уникнути зациклення, а не з тим, щоб ефективно організувати подальший пошук глобального оптимуму.

3.4. Метод глобального рівноважного пошуку

Розглянемо метод адаптивного випадкового пошуку розв'язку задач дискретної оптимізації, названий методом глобального рівноважного пошуку [42]. Ідеї цього методу близькі до ідей методу відпалу [122], асимптотична ефективність якого, незважаючи на успішне застосування для розв'язання багатьох складних оптимізаційних задач [52], є нижчою [82], ніж навіть у тривіального методу повторюваного випадкового локального пошуку (*RLMS*). Методу глобального рівноважного пошуку (ГРП) властиві всі переваги методу відпалу, і при цьому він має більш високу асимптотичну ефективність. Результати чисельних експериментів, отримані з використанням цього методу, дозволили стверджувати про реабілітацію ідей больцманівської оптимізації, яка є, на наш погляд, найкращою, взятою у самої природи.

Далі подамо деякі попередні викладки, необхідні для опису методу ГРП.

Розглянемо задачу цілочислового лінійного програмування з булевими змінними вигляду:

мінімізувати

$$f(x) = \sum_{j=1}^{n} c_j x_j \tag{3.8}$$

при обмеженнях

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \ , \ i = 1, \ldots, m, \tag{3.9}$$

$$0 \le x_j \le 1, j = 1, \ldots, n, \tag{3.10}$$

$$x_j - \text{цілі}, j = 1, \ldots, n. \tag{3.11}$$

Вважаємо, що $c_j \ge 0, j = 1,\ldots,n$.

Позначимо S множину допустимих розв'язків задачі (3.8)–(3.11) і $Z(\mu) = \sum_{x \in S} \exp(-\mu f(x))\ \forall \mu \geq 0$. Випадковий вектор $\xi(\omega)$ визначимо за допомогою такої формули:

$$\pi(x,\mu) = P\{\xi(\mu,\omega) = x\} = \exp(-\mu f(x))/Z(\mu),\ x \in S. \quad (3.12)$$

З імовірнісним розподілом (3.12), відомим у статистичній фізиці як розподіл Больцмана, пов'язаний і метод відпалу, про що було сказано в § 3.2 (див. формулу (3.4)). Стаціонарна імовірність знаходження марківського ланцюга, породжуваного даним методом, в деякій точці x визначається виразом (3.12).

Позначимо також $S_j^1 = \{x : x \in S,\ x_j = 1\}$ і $S_j^0 = \{x : x \in S,\ x_j = 0\}$ множини допустимих розв'язків задачі (3.8)–(3.11), в яких j-а ($j = 1,...,n$) координата дорівнює, відповідно, 1 і 0. Очевидно, що $S = S_j^1 \cup S_j^0$. Для будь-якого $j = 1,...,n$ визначимо

$$Z_j^1(\mu) = \sum_{x \in S_j^1} \exp(-\mu f(x)),\ Z_j^0(\mu) = \sum_{x \in S_j^0} \exp(-\mu f(x)),$$

$$p_j(\mu) = P\{\xi_j = 1\}.$$

Легко показати, що

$$Z(\mu) = Z_j^1 + Z_j^0,\ \ p_j(\mu) = Z_j^1(\mu)/Z(\mu),\ j = 1,...,n,$$

$$\langle f \rangle_\mu = Mf(\xi(\omega)) = -\frac{\partial}{\partial \mu} \ln Z(\mu) = \sum_{j=1}^{n} c_j p_j(\mu).$$

З урахуванням рівності

$$\langle f \rangle_\mu = p_j(\mu) \langle f \rangle_\mu^{x_j=1} + (1 - p_j(\mu)) \langle f \rangle_\mu^{x_j=0},\ j = 1,...,n,$$

обчислимо похідну імовірностей $p_j(\mu)$ за формулами

$$\frac{\partial p_j}{\partial \mu} = \frac{\partial}{\partial \mu} \frac{Z_j^1(\mu)}{Z(\mu)} = \frac{Z(\mu)\frac{\partial Z_j^1(\mu)}{\partial \mu} - Z_j^1(\mu)\frac{\partial Z(\mu)}{\partial \mu}}{(Z(\mu))^2} =$$

$$= p_j(\mu)(\langle f \rangle_\mu - \langle f \rangle_\mu^{x_j=1}),\ j = 1,...,n. \quad (3.13)$$

Очевидно, що

$$\frac{\partial p_j}{\partial \mu} = p_j(\mu)(1 - p_j(\mu))(\langle f \rangle_\mu^{x_j=0} - \langle f \rangle_\mu^{x_j=1}),\ j = 1,...,n, \quad (3.14)$$

де умовні математичні очікування визначаються таким чином:

$$\langle f \rangle_\mu^{x_j=1} = M(f(\xi(\mu,\omega)) \mid x_j = 1) = \frac{\sum_{x \in S_j^1} f(x)\exp(-\mu f(x))}{Z_j^1(\mu)},$$

$$\langle f \rangle_\mu^{x_j=0} = M(f(\xi(\mu,\omega)) \mid x_j = 0) = \frac{\sum_{x \in S_j^0} f(x)\exp(-\mu f(x))}{Z_j^0(\mu)}.$$

Розглянемо незалежні випадкові величини $\eta_j(\mu, \omega)$, $j = 1,...,n$, для яких $P\{\eta_j(\mu,\omega) = 1\} = p_j(\mu)$, $P\{\eta_j(\mu,\omega) = 0\} = 1 - p_j(\mu)$, і випадковий вектор $\zeta(\mu,\omega) = (\zeta_0(\mu,\omega), \zeta_1(\mu,\omega),..., \zeta_m(\mu,\omega))$, де $\zeta_0(\mu,\omega) = \sum_{j=1}^{n} c_j \eta_j(\mu,\omega)$, $\zeta_i(\mu,\omega) = \sum_{j=1}^{n} a_{ij}\eta_j(\mu,\omega)$, $i = 1,...,m$.

При певних припущеннях можна встановити асимптотичну нормальність нормованих випадкових величин $\zeta_i(\mu, \omega)$, $i = 0,...,m$. Для простоти подальших міркувань припустимо нормальність цих величин і, отже, нормальність умовного розподілу величини $\zeta_0(\mu, \omega)$ при обмеженнях $\zeta_i(\mu, \omega) = b_i$, $i = 0, ..., m$. Для цього випадку обчислимо умовне математичне очікування [12]

$$M(\zeta_0(\mu,\omega) \mid \zeta_i(\mu,\omega) = b_i) = M\zeta_0(\mu,\omega) + \sum_{i=1}^{m} r_i(b_i - M\zeta_i(\mu,\omega)) =$$

$$= \sum_{j=1}^{n} c_j p_j(\mu) + \sum_{i=1}^{m} r_i(\mu)\left(b_i - \sum_{j=1}^{n} a_{ij} p_j(\mu)\right) = \sum_{j=1}^{n} c_j p_j(\mu),$$

де $r_i(m)$, $i = 0,...,m$, – коефіцієнти регресії, які визначаються з такої системи лінійних рівнянь:

$$\sum_{i=1}^{m}\left(\sum_{j=1}^{n} a_{kj}a_{ij}p_j(\mu)(1-p_j(\mu))\right) r_j(\mu) = \sum_{j=1}^{n} c_j a_{ij} p_j(\mu)(1-p_j(\mu)),$$

$$k = 1, ..., m. \qquad (3.15)$$

В припущенні, що виконується обмеження

$$\langle f\rangle_{\mu}^{x_j=0} - \langle f\rangle_{\mu}^{x_j=1} =$$

$$M(\zeta_0(\mu,\omega)\,|\,\zeta_i(\mu,\omega) = b_i,\ i = 1,...,m,\ x_j = 0) -$$

$$-M(\zeta_0(\mu,\omega)\,|\,\zeta_i(\mu,\omega) = b_i,\ i = 1,...,m,\ x_j = 1),$$

одержимо співвідношення

$$\langle f\rangle_{\mu}^{x_j=0} - \langle f\rangle_{\mu}^{x_j=1} =$$

$$= \sum_{k=1,k\neq j}^{n} c_k p_k(\mu) + \sum_{i=1}^{m} r_i(\mu)\left(b_i - \sum_{k=1,k\neq j}^{n} a_{ik}p_k(\mu)\right) -$$

$$-c_j - \sum_{k=1,k\neq j}^{n} c_k p_k(\mu) - \sum_{i=1}^{m} r_i(\mu)\left(b_i - a_{ij} - \sum_{k=1,k\neq j}^{n} a_{ik}p_k(\mu)\right) =$$

$$= \sum_{i=1}^{m} r_i(\mu)a_{ij} - c_j\ .$$

Отже, приходимо до формули

$$\frac{\partial p_j(\mu)}{\partial \mu} = p_j(\mu)(1-p_j(\mu))\left(\sum_{i=1}^{m} r_i(\mu)a_{ij} - c_j\right), j = 1, ..., n, \qquad (3.16)$$

яка відповідає варіанту неперервного аналога методу внутрішніх точок для задачі лінійного програмування з двосторонніми обмеженнями (3.8)–(3.10) [5], [10].

Оскільки згідно з (3.15) справджуються співвідношення

$$\sum_{j=1}^{n} a_{kj}\frac{\partial p_j(\mu)}{\partial \mu} = \sum_{j=1}^{n} a_{kj}p_j(\mu)(1-p_j(\mu))\left(\sum_{i=1}^{m} r_i(\mu)a_{ij} - c_j\right) =$$

$$= \sum_{i=1}^{m}\left(\sum_{j=1}^{n} a_{kj}a_{ij}p_j(\mu)(1-p_j(\mu))\right) r_i(\mu) -$$

$$-\sum_{j=1}^{n} c_j a_{kj} p_j(\mu)(1-p_j(\mu)) = 0,\ \ k = 1, ..., m,$$

то за умови, що при $\mu = 0$ мають місце обмеження вигляду (3.9), тобто

$$\sum_{j=1}^{n} a_{ij}p_j(0) = b_i\ ,\ i = 1, ..., m,$$

вказані обмеження виконуватимуться і при будь-яких $\mu > 0$ і $p_j(\mu)$, $j = 1,...,n$, отриманих при інтегруванні системи (3.16).

Згідно з методом відпалу обчислимо похідну від середнього значення цільової функції

$$\frac{\partial\langle f\rangle_{\mu}}{\partial\mu} = \frac{\partial}{\partial\mu}\left(\frac{\sum_{x\in S} f(x)\exp(-\mu f(x))}{Z(\mu)}\right) =$$

$$= -\frac{\sum_{x\in S} f^2(x)\exp(-\mu f(x))}{Z(\mu)} + \left(\frac{\sum_{x\in S} f(x)\exp(-\mu f(x))}{Z(\mu)}\right)^2,$$

яка дорівнює дисперсії значень цільової функції із знаком мінус. Крім того, справедливі співвідношення

$$\frac{\partial\langle f\rangle_{\mu}}{\partial\mu} = \frac{\partial}{\partial\mu}\left(\sum_{j=1}^{n} c_j p_j(\mu)\right) =$$

$$= \sum_{j=1}^{n} c_j p_j(\mu)(1 - p_j(\mu)) \left(\sum_{i=1}^{m} r_i(\mu) a_{ij} - c_j \right) =$$

$$= -\sum_{j=1}^{n} c_j^2 p_j(\mu)(1 - p_j(\mu)) +$$

$$+ \sum_{i=1}^{m} r_i(\mu) \left(\sum_{j=1}^{n} c_j a_{ij} p_j(\mu)(1 - p_j(\mu)) \right) =$$

$$= D^2(\zeta_0(\mu, \omega) \mid \zeta_i(\mu, \omega) = b_i, \quad i = 1, ..., m),$$

де передостанній вираз із знаком мінус являє собою умовну дисперсію випадкової нормально розподіленої величини $\zeta_0(m, \omega)$.

Таким чином, приходимо до висновку, що за умов, які забезпечують асимптотичну нормальність випадкових змінних $\zeta_i(\mu, \omega)$, $i = 0, ..., m$, метод відпалу подібний методам внутрішніх точок, які вважаються одними з найбільш ефективних при розв'язанні задач лінійного програмування (методи Кармаркара [119] і Дікіна-Барнса [59], як наголошується в [47], вельми схожі з ними).

Розв'язки систем (3.13) і (3.14) мають відповідно вигляд

$$p_j(\mu) = p_j(0) \exp\left(\int_0^{\mu} \left(\langle f \rangle_t - \langle f \rangle_t^{x_j = 1} \right) dt \right), \qquad (3.17)$$

$$p_j(\mu) = \frac{1}{1 + \dfrac{1 - p_j(0)}{p_j(0)} \exp\left(-\displaystyle\int_0^{\mu} \left(\langle f \rangle_t^{x_j = 0} - \langle f \rangle_t^{x_j = 1} \right) dt \right)}. \qquad (3.18)$$

Скориставшись формулами (3.16), прийдемо також до таких розв'язків:

$$p_j(\mu) = \frac{1}{1 + \dfrac{1 - p_j(0)}{p_j(0)} \exp\left(-\displaystyle\int_0^{\mu} \left(\sum_{i=1}^{m} r_i(t) a_{ij} - c_j \right) dt \right)} =$$

$$= \frac{1}{1 + \dfrac{1 - p_j(0)}{p_j(0)} \exp\left(\mu c_j - \displaystyle\sum_{i=1}^{m} a_{ij} \int_0^{\mu} r_i(t) dt \right)}. \qquad (3.19)$$

Перейдемо тепер безпосередньо до опису методу глобального рівноважного пошуку при застосуванні його до розв'язання задачі (3.8)–(3.11).

Позначимо $\tilde{S}$ деяку підмножину множини розв'язків задачі (3.8)–(3.11), знайдених методом ГРП, і покладемо

$$\tilde{S}_j^1 = \{x : x \in \tilde{S},\ x_j = 1\},\ \tilde{S}_j^0 = \{x : x \in \tilde{S},\ x_j = 0\}.$$

Задамо числа K і $\mu_0 < \mu_1 < ... < \mu_k$. Для всіх значень $k = 0,...,K$ і $j = 1,...,n$ визначимо такі величини:

$$\tilde{Z}_k = \sum_{x \in \tilde{S}} \exp\{-\mu_k f(x)\},$$

$$\tilde{F}_k = \sum_{x \in \tilde{S}} f(x) \exp\{-\mu_k f(x)\},$$

$$\tilde{E}_k = \frac{\tilde{F}_k}{\tilde{Z}_k},$$

$$\tilde{Z}_{kj}^1 = \sum_{x \in \tilde{S}_j^1} \exp\{-\mu_k f(x)\},\ \tilde{F}_{kj}^1 = \sum_{x \in \tilde{S}_j^1} f(x) \exp\{-\mu_k f(x)\},$$

$$\tilde{E}_{kj}^1 = \frac{\tilde{F}_{kj}^1}{\tilde{Z}_{kj}^1},$$

$$\tilde{Z}_{kj}^0 = \sum_{x \in \tilde{S}_j^0} \exp\{-\mu_k f(x)\},$$

$$\tilde{F}_{kj}^0 = \sum_{x \in \tilde{S}_j^0} f(x) \exp\{-\mu_k f(x)\},$$

$$\tilde{E}_{kj}^0 = \frac{\tilde{F}_{kj}^0}{\tilde{Z}_{kj}^0},$$

$$\tilde{p}_{ij} = \frac{1}{1+\dfrac{1-\tilde{p}_{0j}}{\tilde{p}_{0j}}\exp\left\{-0{,}5\sum_{i=0}^{k-1}\left(\tilde{E}^0_{ij}+\tilde{E}^0_{i+1,j}-\tilde{E}^1_{ij}+\tilde{E}^1_{i+1,j}\right)\left(\mu_{i+1}-\mu_i\right)\right\}}. \quad (3.20)$$

Зауважимо, що вираз (3.20) отримано застосуванням формули трапецій до інтеграла у співвідношенні (3.18).

Схему методу глобального рівноважного пошуку подамо у вигляді таких процедур:

procedure ГРП
початкова_установка ($s, s_{max}, r, K, M, nt, S$)
while $s < s_{max}$ *do*
repeat for $k = 0$ *to* K
$\mu = M(k)$
обчислення згідно з формулою (3.20)
імовірностей $\tilde{p}_j(\mu)$, $j = 1,...,n$,
repeat for $i = 1$ *to* nt
$x^s =$ імовірнісна_генерація ($\tilde{p}(\mu)$)
while пошук_покращення $(x^s, r) \neq$ "ні" *do*
$x^{s+1} =$ пошук_покращення (x^s, r)
$s = s + 1$
if $x^s \notin S$ *then* $S = S \cup \{x^s\}$
$s = s + 1$

procedure початкова_установка($s, s_{max}, r, K, M, nt, S$)
число ітерацій $s = 0$,
множина знайдених розв'язків $S = \varnothing$,
вибір величини s_{max} – максимального числа ітерацій,
задання номера околу r, величини K і масиву $M[0:K]$,
де $M[k] = \mu_k$, $k = 0,...,K$, які визначають розклад відпалу,
вибір величини nt – кількості локальних оптимумів;

імовірнісна_генерація($\tilde{p}(\mu)$) = { деякому розв'язку $y \in G$, одержаному генерацією випадкового вектора з розподілом $\tilde{p}_k(\mu)$, $k = 1,...,n$;

пошук_покращення(x^s, r) = { будь-якому розв'язку $y \in N^r(x^s)$, $f(y) < f(x^s)$, якщо такий розв'язок існує, "ні" – в протилежному випадку.

Детальніше зупинимось на деяких моментах наведеної схеми методу ГРП.

Імовірності випадкового пошуку. При $\mu = 0$ імовірності $p_j(0)$, $j = 1, ..., n$, можна визначити за допомогою статистичного моделювання таким чином. Генерується необхідна кількість рівноймовірних перестановок координат вектора розв'язку задачі. Далі для розв'язання початкової задачі (3.8)–(3.11) застосовується алгоритм послідовного призначення одиниць, що використовує отриману перестановку. Обчислена частота присвоєння одиничного значення змінній x_j, $j = 1,...,n$, може бути хорошою оцінкою для величини $p_j(0)$. При $\mu_0 = 0$ маємо $\tilde{p}_{0j} = p_j(0)$, $j = 1, ..., n$. Розраховувати $p_j(\mu)$ для не дуже великих значень μ можна, чисельно інтегруючи вираз (3.19) і покладаючи $\mu_0 = \mu$, $\tilde{p}_{0j} = p_j(\mu)$. При визначенні μ_k, $k = 1,...,K$, доцільно

вибрати $\alpha > 1$, покласти $\mu_k = \alpha^k \mu_0$ і виходити з умови, згідно з якою при "температурі" μ_k (див. [65, 122]) імовірності $\bar{p}_{kj}$ приблизно дорівнюють найкращому з отриманих розв'язків.

Якщо в температурному циклі по k не знайдено кращий розв'язок, то має сенс збільшити nt. У протилежному випадку доцільно не змінювати (не зменшувати) nt.

Визначення множини $\bar{S}$. Припустимо, що в температурному циклі знайдено кращий розв'язок. У такому випадку можна в $\bar{S}$ залишити тільки цей розв'язок або кілька кращих розв'язків (так звану еліту). У протилежному випадку зберігаємо попередню множину $\bar{S}$. Для запам'ятовування отриманих розв'язків (множини $\bar{S}$) можна застосовувати техніку хешування [11]. Зауважимо, що для початкового "розгону" методу ГРП іноді доцільно попередньо застосувати швидкодійний наближений метод, наприклад метод вектора спаду [27]. Тоді перед першими розрахунками температурного циклу слушно вважати, що множина $\bar{S}$ складається із знайденого за методом вектора спаду розв'язку.

Головною рисою методу ГРП є навіть не притаманна йому обчислювальна ефективність, а високі "командні" якості, під якими розуміємо здатність методу ефективно використовувати розв'язки, отримані іншими алгоритмами. Очевидно, в класичному методі локального пошуку показник командності дорівнює нулю. Командні властивості дозволяють дуже ефективно застосовувати ГРП при паралельних розрахунках, що є особливо важливим при оптимізації в комп'ютерних мережах. На наш погляд, будь-яка задача оптимізації, яка задовольняє певні вимоги до структури оптимуму (зокрема, ПО-принципу), може ефективно розв'язуватися за допомогою методу ГРП.

Частина II

Оптимізація проектування надійних мереж

Розділ 4. Математичні моделі оптимізації проектування надійних мереж

Задачі проектування мережі мінімальної вартості з врахуванням виходу з ладу окремих її ланок виникають при проектуванні надійних комунікаційних мереж за умови забезпечення повноцінного функціонування системи при відмовах окремих компонент мереж (ребер, вузлів).

4.1. Задачі знаходження пропускних здатностей дуг надійної орієнтованої мережі

Поняття "пошкодження мережі" є центральним поняттям у даному розділі при поданні математичних моделей надійних мереж, тобто таких, що можуть функціонувати у випадку можливих пошкоджень. Пошкодженням мережі будемо називати такий її стан, при якому зменшується пропускна здатність однієї або кількох її дуг. Конкретизуємо поняття "пошкодження мережі" для орієнтованої мережі, застосовуючи такі позначення.

Нехай $N(V, A)$ – орієнтована мережа з множиною вершин V і множиною дуг A. Дугу мережі $N(V, A)$, спрямовану з вершини $i \in V$ до вершини $j \in V$, позначимо $(i, j) \in A$. Нехай $Y\{y_{ij}, (i, j) \in A\}$ – вектор невід'ємних значень пропускних

здатностей дуг мережі $N(V, A)$. Вважаємо, що пошкодженню мережі $N(V, A)$ відповідає такий її стан, при якому значення пропускних здатностей дуг мережі $N(V, A)$ стають такими: $Y'=\{y'_{ij}, (i, j) \in A\}$, де $y'_{ij} = \mu_{ij} y_{ij}$, $0 \le \mu_{ij} \le 1$, $(i, j) \in A$. Якщо $\mu_{ij} = 0$ для дуги $(i, j) \in A$, то при пошкодженні мережі $N(V, A)$ це буде рівносильно повній відмові цієї дуги, при якій пропускна здатність дуги (i, j) дорівнює нулю ($y'_{ij} = 0$), і пересилати по ній потік неможливо. Якщо $\mu_{ij} = 1$, то при пошкодженні мережі $N(V, A)$ це означає повноцінне функціонування дуги (i, j), коли пропускна здатність дуги (i, j) залишається такою ж, як і до пошкодження ($y'_{ij} = y_{ij}$), а максимальний обсяг потоку, який можна пересилати по цій дузі, також не змінюється. Проте можливі й різні проміжні стани, коли $0 < \mu_{ij} < 1$.

За допомогою введеного вище поняття "пошкодження мережі" можна описати різноманітні аварійні ситуації, які відбуваються у функціонуючих мережах будь-якого типу: автодорожних, залізничних, комунікаційних тощо.

Відзначимо, що окремим прикладом пошкодження мережі є ситуація, пов'язана з одиничними відмовами дуг мережі $N(V, A)$, коли може відмовити тільки одна, але будь-яка дуга. В цьому випадку кількість можливих пошкоджень T буде дорівнювати кількості дуг мережі $N(V, A)$.

Ситуацію, пов'язану з повною відмовою деякого заданого списку дуг, також можна описати подібним чином. У цьому випадку, як і при одиничних відмовах, параметр μ_{ij} може дорівнювати тільки одиниці або нулю.

Далі буде розглянуто дві задачі знаходження пропускних здатностей дуг надійної орієнтованої мережі. Для них надійною орієнтованою мережею вважається така орієнтована мережа, для якої можна задовольнити всі задані вимоги на передачу об'ємів потоків у мережі як при відсутності пошкоджень у мережі, так і в тому випадку, коли буде тільки одне пошкодження, але довільне з усіх можливих пошкоджень орієнтованої мережі. Пошкодження в орієнтованій мережі задані списком можливих пошкоджень, де конкретним пошкодженням є зменшення пропускної здатності однієї або одразу кількох дуг орієнтованої мережі.

4.1.1. Задача знаходження пропускних здатностей дуг надійної орієнтованої мережі з передачею потоків по довільних шляхах

При формулюванні математичної моделі задачі, яка розглядається в даному підрозділі, будемо використовувати такі вхідні дані:

(і) орієнтована мережа $N(V, A)$ задана множиною вершин V та множиною дуг A. Для дуги $(i, j) \in A$ позначимо c_{ij} вартість створення одиниці пропускної здатності, а y^0_{ij} – наявний ресурс пропускної здатності;

(іі) задано множину вимог D щодо обсягів потоків між парами вершин з деякої підмножини $V_0 \subset V$. Кожен елемент множини D задається трьома числовими значеннями: парою (r, s) та відповідним значенням обсягу потоку d_{rs}, що слід пропустити через мережу від вершини $r \in V_0$ (джерело) до вершини $s \in V_0$ (стік). Для всіх пар (r, s) відповідні значення обсягів d_{rs} задані в тих же одиницях, що і значення пропускних здатностей дуг;

(ііі) задано множину T можливих пошкоджень мережі $N(V, A)$. Пошкодження $t \in T$ мережі характеризується набором коефіцієнтів $0 \le \mu_{ijt} \le 1$ для всіх $(i, j) \in A$, де конкретний коефіцієнт μ_{ijt} вказує на те, що пропускна здатність дуги (i, j) у мережі при пошкодженні t зменшиться в $1/\mu_{ijt}$ раз. Для зручності опису математичних моделей індекс $t = 0$ умовимося вважати нульовим пошкодженням мережі $N(V, A)$, якому відповідає набір коефіцієнтів $\mu_{ij0} = 1$ для всіх $(i, j) \in A$. Фактично це означає, що "нульове" пошкодження мережі рівносильне функціонуванню мережі $N(V, A)$ в режимі, коли пошкоджень немає.

Для позначення потужності (розміру) розглянутих вище множин будемо користуватися символом $|\cdot|$, тобто $|N|$ – число

вершин, а $|A|$ – число дуг у мережі $N(V, A)$; $|T|$ – число пошкоджень у мережі $N(V, A)$; $|D|$ – число пар вершин з мережі $N(V, A)$, для яких задані обсяги потоків, що пересилаються.

Зробимо таке припущення щодо даних (і)–(ііі).

Припущення 4.1. Для орієнтованої мережі $N(V, A)$ всі вимоги D по передачі обсягів потоків між парами вершин можна задовольнити при необмежених значеннях пропускних здатностей дуг цієї мережі $N(V, A)$.

Нехай $Y=\{y_{ij},\ (i, j) \in A\}$ – множина невідомих значень пропускних здатностей дуг $(i, j) \in A$, які потрібно додати до вже існуючих значень пропускних здатностей дуг $Y^0=\{y^0_{ij},\ (i, j) \in A\}$ мережі, а x^{rs}_{ijt} – невідоме значення тієї частини обсягу потоку d_{rs}, $(r, s) \in D$, який при t-му пошкодженні мережі буде пропущений по дузі $(i, j) \in A$. Тоді математичну модель задачі знаходження "оптимальних за сумарною вартістю" значень Y для надійної орієнтованої мережі $N(V, A)$ можна сформулювати так:

мінімізувати лінійну функцію

$$\sum_{(i,j)\in A} c_{ij} y_{ij} \tag{4.1}$$

при обмеженнях

$$\sum_{(r,s)\in D} x^{rs}_{ijt} \le \mu_{ijt}(y^0_{ij} + y_{ij}),\ t \in (0 \cup T), (i, j) \in A, \tag{4.2}$$

$$\sum_{j:(i,j)\in A} x^{rs}_{ijt} - \sum_{j:(j,i)\in A} x^{rs}_{jit} = \begin{cases} d_{rs}, & i = r, \\ 0, & i \ne r, s, \\ -d_{rs}, & i = s, \end{cases} \quad t \in (0 \cup T), (r, s) \in D, \tag{4.3}$$

$$x^{rs}_{ijt} \ge 0,\ t \in (0 \cup T),\ (i, j) \in A,\ (r, s) \in D, \tag{4.4}$$

$$y_{ij} \ge 0,\ (i, j) \in A. \tag{4.5}$$

Тут функція (4.1), яка мінімізується, задає сумарні витрати з врахуванням вартості тих пропускних здатностей (що доповнюють вже існуючі) дуг, які потрібно знайти з метою забезпечення надійності мережі $N(V, A)$.

Зміст обмежень (4.2)–(4.5) досить прозорий. Так, обмеження (4.3) вказують на обов'язкове виконання вимог D по передачі обсягів потоку в мережі $N(V, A)$ і повинні бути виконані як при відсутності пошкоджень мережі ($t = 0$), так і в кожному з випадків, коли відбудеться тільки одне (але будь-яке з множини T) пошкодження мережі. При кожному $t \in (0 \cup T)$ необхідно задовольнити й обмеження виду (4.2). Вони означають, що сумарний потік через дугу $(i, j) \in A$ не повинен перевищувати пропускної здатності цієї дуги з врахуванням можливих змін при пошкодженні мережі. Пропускна здатність дуги $(i, j) \in A$ складається з уже наявної (y^0_{ij}) та тієї невідомої (y_{ij}) пропускної здатності, яка додається. Обмеження (4.4) і (4.5) пов'язані з невід'ємністю змінних x^{rs}_{ijt} та y_{ij}, відповідно.

Зауважимо, що система обмежень (4.2)–(4.5) може бути несумісною, навіть якщо виконане припущення 4.1. Це може статися через неможливість виконати обмеження (4.2) за даної структури пошкоджень мережі. Для забезпечення сумісності системи обмежень (4.2)–(4.5) введемо додаткові додатні змінні $x_{ijt} \ge 0$, $(i, j) \in A$, $t \in (0 \cup T)$ та дамо їм таке тлумачення: x_{ijt} відповідає тому значенню пропускної здатності дуги $(i, j) \in A$, якого не вистачає при виникненні t-го пошкодження мережі для виконання обмеження виду (4.2), що відповідають цим $(i, j) \in A$ і $t \in (0 \cup T)$. Тоді аналог задачі (4.1)–(4.5) можна сформулювати в такому вигляді:

мінімізувати лінійну функцію

$$\sum_{t\in(0\cup T)} \sum_{(i,j)\in A} Q_{ij} x_{ijt} + \sum_{(i,j)\in A} c_{ij} y_{ij} \tag{4.6}$$

при обмеженнях

$$\sum_{(r,s)\in D} x^{rs}_{ijt} - x_{ijt} \le \mu_{ijt}(y^0_{ij} + y_{ij}),\ t \in (0 \cup T), (i, j) \in A, \tag{4.7}$$

$$\sum_{j:(i,j)\in A} x_{ij}^{rs} - \sum_{j:(j,i)\in A} x_{ji}^{rs} = \begin{cases} d_{rs}, & i = r, \\ 0, & i \neq r,s, \\ -d_{rs}, & i = s, \end{cases} \quad t \in (0 \cup T),(r,s) \in D, \tag{4.8}$$

$$x_{ij}^{rs} \geq 0,\ x_{ij} \geq 0,\ t \in (0 \cup T),(i,j) \in A,(r,s) \in D, \tag{4.9}$$

$$y_{ij} \geq 0,\ (i,j) \in A, \tag{4.10}$$

де "штрафні" множники Q_{ij} вибрані такими, щоб значення додаткових змінних в оптимумі збігалися до нуля, якщо система (4.2)–(4.5) сумісна. Для цього досить вибрати значення Q_{ij}, більші за $\sum_{(i,j)\in A} c_{ij}$.

Система обмежень (4.7)–(4.10) завжди буде сумісною при виконанні припущення 4.1. При $Q_{ij} > \sum_{(i,j)\in A} c_{ij}$ додаткові змінні в оптимумі будуть набувати нульових значень, якщо надійне функціонування мережі $N(V, A)$ можливо забезпечити. Якщо ж через структуру пошкоджень надійне функціонування мережі $N(V, A)$ неможливе, то ненульові оптимальні значення x_{ij}^{*} будуть характеризувати ті критичні місця в мережі $N(V, A)$, через які неможливо забезпечити її надійне функціонування.

Задача (4.6)–(4.10) є задачею лінійного програмування. Вона має велику вимірність (велику кількість змінних і обмежень) для мереж з кількістю дуг порядку сотень і такою ж за порядком кількістю пошкоджень мережі. В такому випадку число змінних становить

$$N_1 = |A| + |A| \times |D| \times (|T| + 1) + |A| \times (|T| + 1),$$

де перший доданок задає повну кількість змінних y_{ij}, другий – змінних x_{ij}^{rs} і третій – змінних x_{ij}, а число обмежень (без врахування найпростіших) становить

$$M_1 = (1 + |T|) \times |E| + (1 + |T|) \times |D| \times |V|,$$

де перший доданок у сумі задає повне число обмежень (4.7), а другий – число обмежень (4.8).

Наприклад, при $|A| \sim 100$, $|D| \sim 100$, $|V| \sim 100$, $|T| \sim 1000$ кількість змінних N_1 і обмежень M_1 має порядок 10^7. У цьому випадку задачу (4.6)–(4.10) можна віднести до класу важко розв'язуваних.

Однак обмеження в задачі (4.6)–(4.10) мають блочну структуру, і це дає можливість будувати методи її розв'язування і для більших, ніж наведені вище, розмірів мережі, наприклад, при $|A| \sim 1000$. Структуру задачі подамо такою схемою:

мінімізувати

$C_1^T X_0$	$C_1^T X_1$	...	$C_1^T X_{|T|}$	$C_2^T Y$	

при обмеженнях

WX_0				$-I_{|A|}Y$	'≤' чи '='	b_0
	WX_1			$-I_{|A|}Y$	'≤' чи '='	b_1
		...		...	...	...
			$WX_{|T|}$	$-I_{|A|}Y$	'≤' чи '='	$b_{|T|}$

Матриця W, в свою чергу, має таку структуру:

$$W = \begin{array}{|c|c|c|c|} \hline W_0 & W_0 & ... & W_0 \\ \hline B_0 & & & \\ \hline & B_0 & & \\ \hline & & ... & \\ \hline & & & B_0 \\ \hline \end{array}$$

Тут $I_{|A|}$ – одинична матриця розміром $|A| \times |A|$; матриці W, W_0, B_0, вектори C_1, C_2 і b_k ($k = 0, 1, ..., |T|$) побудовані на основі початкової інформації про задачу; $X_k = \{x_{ij}^{rs}, x_{ij}\}$, $k = 0, 1, ..., |T|$; $Y = \{y_{ij}, (i,j) \in A\}$.

З наведеної структури задачі легко бачити, що змінні $Y \in$ зв'язуючими в задачі (4.6)–(4.10). Тому для її розв'язання доцільно використовувати схему декомпозиції за змінними $Y = \{y_{ij}, (i, j) \in A\}$ (див. підрозділ 2.2). При цьому координуюча (зовнішня) задача полягає в мінімізації негладкої опуклої функції $F(Y)$ від зв'язуючих змінних Y, і для знаходження її мінімуму можна застосовувати r-алгоритм (див. підрозділ 1.1), який вважається одним з ефективних методів негладкої оптимізації.

Внутрішня підзадача, яку потрібно розв'язати для обчислення субградієнтів функції $F(Y)$, буде пов'язана з пошуком значень двоїстих змінних до обмежень (4.12) для задачі лінійного програмування такого виду:

$$\sum_{(i,j)\in A} Q_{ij} x_{ij} \to \min, \tag{4.11}$$

$$\sum_{(r,s)\in D} x_{ij}^{rs} - x_{ij} \le \mu_j (y_{ij}^0 + \overline{y}_{ij}), \ (i,j) \in A, \tag{4.12}$$

$$\sum_{j:(i,j)\in A} x_{ij}^{rs} - \sum_{j:(j,i)\in A} x_{ji}^{rs} = \begin{cases} d_{rs}, & i = r; \\ 0, & i \ne r, s; \\ -d_{rs}, & i = s; \end{cases} \quad (r,s) \in D, \tag{4.13}$$

$$x_{ij}^{rs} \ge 0, \ x_{ij} \ge 0, \ (i,j) \in A, (r,s) \in D, \tag{4.14}$$

де $\overline{y}_{ij}$ – відомі поточні (на даний момент) значення пропускних здатностей. При обчисленні субградієнта функції $F(Y)$ внутрішню підзадачу доведеться розв'язувати $(1 + |T|)$ раз.

Для розв'язання внутрішньої підзадачі (4.11)–(4.14) можна залучити як схему декомпозиції за обмеженнями з використанням r-алгоритму, так і стандартні програми для розв'язування задач лінійного програмування (наприклад, методи внутрішніх точок [4, 5], що дозволяють врахувати блочну структуру підзадачі). При використанні r-алгоритму потрібна тільки підпрограма для знаходження найкоротших шляхів в орієнтованій мережі.

Відзначимо, що описана задача знаходження оптимальних пропускних здатностей дуг надійної мережі може бути віднесена до задач синтезу мережі (див., наприклад, [113], с. 248).

4.1.2. Задача знаходження пропускних здатностей дуг надійної орієнтованої мережі з передачею потоків по заданій множині допустимих шляхів

Для орієнтованої мережі $N(V, A)$ розглянемо задачу знаходження мінімальних витрат на підвищення пропускних здатностей дуг. Ця задача багато в чому близька до задачі з п. 4.1.1, але спосіб передачі обсягів потоку з джерел у стоки в мережі тут інший. Якщо в задачі з п. 4.1.1 для передачі обсягу потоку з джерела в стік міг бути будь-який шлях, який можна було побудувати в мережі $N(V, A)$, то для розглядуваної задачі можна брати тільки ті шляхи, що належать до множини допустимих шляхів між джерелом і стоком.

Тут буде використана інформація (i)–(iii) з п. 4.1.1 з такими доповненнями:

(iv) задано множину допустимих шляхів у мережі $P = \bigcup_{(r,s)\in D} P(r,s)$, де $P(r, s)$ – підмножина шляхів у мережі $N(V, A)$, які з'єднують джерело r зі стоками s, $(r, s) \in D$, по яких (і тільки по них) можна пересилати потік із вершини до вершини s. Будемо припускати, що такі набори шляхів задані для всіх пар $(r, s) \in D$. Конкретний шлях $P_k(r, s) \in P(r, s)$, тобто шлях з номером k для передачі обсягу потоку з вершини r до вершини s, будемо задавати вектором довжини a_k^{rs}, який складається з нулів і одиниць, де тим дугам мережі $N(V, A)$, через які проходить цей шлях, відповідають одиниці, а тим дугам, що не входять у цей шлях, відповідають нулі. *Зауважимо, що при необхідності збереження такого вектора досить використовувати тільки перелік дуг мережі, які входять у цей шлях.*

Потужність множини можливих шляхів будемо позначати $|P|$ (це кількість заданих шляхів у мережі $N(V, A)$), а $|P(r, s)|$ – кількість заданих шляхів у мережі $N(V, A)$, що можуть бути використані для пересилання потоку з джерела r у стік s.

Припущення 4.2. Множина P містить непорожні підмножини $P(r, s)$ для всіх пар $(r, s) \in D$.

Нову задачу знаходження для орієнтованої мережі $N(V, A)$ мінімальних за сумарною вартістю значень пропускних здатностей дуг $Y = \{y_{ij}, (i, j) \in A\}$, які потрібно додати до вже існуючих (Y_0), щоб мережа $N(V, A)$ стала надійною, можна сформулювати таким чином:

мінімізувати лінійну функцію

$$\sum_{(i,j)\in A} c_{ij} y_{ij} \tag{4.15}$$

при обмеженнях

$$\sum_{(r,s)\in D} \sum_{k\in P(r,s)} a_k^{rs} x_{kt}^{rs} \le \mu_{ijt}(y_{ij}^0 + y_{ij}),\ t \in (0 \cup T),\ (i,j) \in A, \tag{4.16}$$

$$\sum_{k\in P(r,s)} x_{kt}^{rs} = d_{rs},\ t \in (0 \cup T),\ (r,s) \in D, \tag{4.17}$$

$$x_{kt}^{rs} \ge 0,\ t \in (0 \cup T),\ (r,s) \in D,\ k \in P(r,s), \tag{4.18}$$

$$y_{ij} \ge 0,\ (i,j) \in A, \tag{4.19}$$

де x_{kt}^{rs} – невідомі значення частини обсягу потоку d_{rs}, який буде пропущений шляхом $k \in P(r, s)$ при t-му пошкодженні мережі.

Система обмежень (4.16)–(4.19) має зміст, аналогічний тому, що й система обмежень (4.2)–(4.5). Відмінність полягає в тому, що в мережі $N(V, A)$ вимоги D слід виконувати дещо іншим способом, при якому обсяг потоку з джерела r у стік s може бути пересланий тільки тими шляхами з r в s, які задані множиною $P(r, s)$. Ця особливість характеризується за допомогою змінної x_{kt}^{rs}, яка має інший зміст, ніж змінна x_{ijt}^{rs} в п. 4.1.1.

Система обмежень (4.16)–(4.19) може бути несумісною при виконанні припущення 4.2. Це може статися через неможливість задовольнити обмеження (4.16) при існуючій структурі пошкоджень мережі. Для забезпечення сумісності цієї системи обмежень використовуються додаткові додатні змінні $x_{ijt} \ge 0$,

$(i, j) \in A$, $t \in (0 \cup T)$, що мають такий же зміст, як і в п. 4.1.1. Тоді аналог задачі (4.15)–(4.19) можна сформулювати в такому вигляді:

мінімізувати лінійну функцію

$$\sum_{t\in(0\cup T)} \sum_{(i,j)\in A} Q_{ij} x_{ijt} + \sum_{(i,j)\in A} c_{ij} y_{ij} \tag{4.20}$$

при обмеженнях

$$\sum_{(r,s)\in D} \sum_{k\in P(r,s)} a_k^{rs} x_{kt}^{rs} - x_{ijt} \le \mu_{ijt}(y_{ij}^0 + y_{ij}),\ t \in (0 \cup T), (i,j) \in A, \tag{4.21}$$

$$\sum_{k\in P(r,s)} x_{kt}^{rs} = d_{rs},\ t \in (0 \cup T),\ (r,s) \in D, \tag{4.22}$$

$$x_{kt}^{rs} \ge 0, x_{ijt} \ge 0,\ t \in (0 \cup T),\ (r,s) \in D,\ k \in P(r,s), \tag{4.23}$$

$$y_{ij} \ge 0,\ (i,j) \in A, \tag{4.24}$$

де x_{ijt} та Q_{ij} – додаткові змінні та відповідні штрафні множники, які мають такий же зміст, як і в п. 4.1.1. Досить вибрати "штрафні" множники Q_{ij} більшими, ніж $\sum_{(i,j)\in A} c_{ij}$, щоб додаткові змінні в оптимумі збігалися до нуля, якщо система (4.16)–(4.19) сумісна.

Система обмежень (4.21)–(4.24) завжди буде сумісною за умови виконання припущення 4.2. При $Q_{ij} > \sum_{(i,j)\in A} c_{ij}$ значення додаткових змінних в оптимумі будуть дорівнювати нулю, якщо можна забезпечити надійне функціонування мережі $N(V, A)$. Якщо ж через структуру пошкоджень мережі забезпечити це неможливо, то ненульові оптимальні значення змінних x_{ijt}^* будуть характеризувати ті критичні дуги мережі $N(V, A)$, через які неможливо забезпечити її надійне функціонування.

Задача (4.20)–(4.24) є задачею лінійного програмування, і для мереж з величинами $|A|$, $|T|$ і $|P|$ порядку сотень має велику вимірність. Обмеження в ній характеризуються аналогічною

блочною структурою, як і в задачі (4.15)–(4.19). Для розв'язування задачі (4.20)–(4.24) будемо використовувати схему декомпозиції по зв'язуючим змінним $Y = \{y_{ij}, (i, j) \in A\}$ (див. підрозділ 2.2). Координуючою (зовнішньою) задачею буде задача мінімізації негладкої опуклої функції $F(Y)$ від змінних Y, для розв'язання якої можна застосувати r-алгоритм (див. підрозділ 1.1).

Щоб обчислити субградієнт функції $F(Y)$, потрібно $(|T| + 1)$ раз розв'язати підзадачу, пов'язану із знаходженням двоїстих змінних до обмежень (4.26) для такої задачі лінійного програмування:

мінімізувати функцію

$$\sum_{(i,j)\in A} Q_{ij} y_{ij} \tag{4.25}$$

при обмеженнях

$$\sum_{(r,s)\in D} \sum_{k\in P(r,s)} a_k^{rs} x_k^{rs} - x_{ij} \le \mu_{ij}(y_{ij}^0 + \overline{y}_{ij}),\ (i,j) \in A, \tag{4.26}$$

$$\sum_{k\in P(r,s)} x_k^{rs} = d_{rs},\ (r,s) \in D, \tag{4.27}$$

$$x_k^{rs} \ge 0,\ x_{ij} \ge 0,\ (r,s) \in D,\ k \in P(r,s), \tag{4.28}$$

де $\overline{y}_{ij}$ – відомі поточні (на даний момент) значення пропускних здатностей. Ця підзадача простіша, ніж відповідна підзадача з п. 4.1.1, і для її розв'язування можна використовувати r-алгоритм у комбінації зі схемою декомпозиції за обмеженнями (див. підрозділ 2.1).

4.1.3. Програми для знаходження оптимальних пропускних здатностей дуг надійної орієнтованої мережі та їх тестування

Для задач знаходження мінімальних за сумарною вартістю пропускних здатностей дуг для побудови надійної орієнтованої мережі реалізовані такі програми: програма ModelA для задачі (4.6)–(4.10) та програма ModelC для задачі (4.20)–(4.24) (мова

програмування ФОРТРАН). Алгоритми розв'язування обох задач базуються на подвійному використанні двох схем декомпозиції (одна в одній):

- схема декомпозиції за змінними, якими є невідомі значення пропускних здатностей дуг орієнтованої мережі;
- схема декомпозиції за обмеженнями (для розв'язання підзадач, які виникають для кожного пошкодження при фіксованих значеннях пропускних здатностей дуг).

Схемі декомпозиції за змінними відповідає координуюча задача мінімізації негладкої опуклої функції, а схемі декомпозиції за обмеженнями – задача максимізації негладкої ввігнутої функції. Задачі негладкої оптимізації, що відповідають вказаним схемам декомпозиції, розв'язуються за допомогою r-алгоритму з адаптивним регулюванням крокового множника (див. підрозділ 1.1), де параметри r-алгоритму вибрані таким чином: коефіцієнт розтягу простору $\alpha = 4$, а параметри адаптивного регулювання крокового множника – $n_k = 3$, $q_1 = 1$, $q_2 = 1{,}1$. Вибране максимальне число ітерацій, відведене r-алгоритму для розв'язування підзадач, дорівнює 500.

В результаті роботи програм ModelA та ModelC отримуємо такі параметри надійної орієнтованої мережі $N(V, A)$:

- мінімальні за сумарною вартістю пропускні здатності дуг мережі $N(V, A)$, що доповнюють вже існуючі, тобто для кожної дуги $(i, j) \in A$ знаходиться y_{ij}^* – ресурс пропускної здатності дуги (i, j), що доповнює вже існуюче значення пропускної здатності цієї дуги y_{ij}^0;
- достатня умова того, що неможливо виконати всі вимоги на передачу обсягів потоків, щоб орієнтована мережа була надійною. Це може бути з двох причин: а) структура мережі $N(V, A)$ така, що задовольнити вимоги до потоків неможливо; б) структура пошкоджень у мережі $N(V, A)$ така, що задовольнити вимоги до потоків неможливо.

Достатньою умовою невиконання всіх вимог на передачу обсягів потоків в мережі є нерівність $\sum_{(i,j)\in A} c_{ij} y_{ij}^* < F^*$, де

F^* – оптимальне значення функції (4.6) або функції (4.20). Наведена нерівність означає, що оптимальне значення цільової функції більше, ніж мінімальні за вартістю знайдені пропускні здатності дуг орієнтованої мережі. Це має місце тоді і тільки тоді, коли в F^* вносить ненульовий вклад "штрафна" частина цільової функції

$$\sum_{t=0}^{|T|} \sum_{(i,j)\in A} Q_{ijt} x_{ijt}^* ,$$

де штрафні множники вибираються однаковими:

$$Q_{ijt} = \sum_{(i,j)\in A} c_{ij} + 1.$$

Загальна структура програми ModelA наведена на рис.4.1, де вказані блоки виконують такі функції.

Головна програма ModelA керує процесом знаходження розв'язку задачі (4.6)–(4.10). В програмі встановлюються штрафні параметри (на основі вхідних даних), запускається процесс розв'язування задачі (4.6)–(4.10), аналізується отриманий розв'язок, результати розрахунку виводяться у файл протоколу роботи програми. Програма ModelA використовує підпрограми RdataA та CoordA.

Підпрограма RDataA читає вхідні дані задачі та перевіряє їх коректність за рядом ознак. Вхідні дані читаються послідовно з трьох файлів, які містять: а) структуру орієнтованої мережі; б) обсяги потоків між парами вершин, що пересилаються по мережі; в) структуру пошкоджень мережі.

Підпрограма CoordA реалізує роботу r-алгоритму для розв'язування координуючої негладкої задачі, пов'язаної із схемою декомпозиції за змінними (використовує підпрограму FunGA1).

Підпрограма FunGA1 обчислює значення негладкої координуючої функції та її субградієнта, для чого послідовно для кожного з пошкоджень використовується підпрограма DualA.

Підпрограмма DualA знаходить двоїсті змінні до обмежень (4.12) підзадачі (4.11)–(4.14). Для їх знаходження за допомогою r-алгоритму розв'язується задача максимізації негладкої ввігнутої функції, що відповідає схемі декомпозиції за обмеженнями (4.12).

Підпрограма FunGA1 обчислює значення функції та субградієнта для підпрограми DualA, для чого використовується процедура знаходження найкоротшого шляху в орієнтованій мережі [53, с.137].

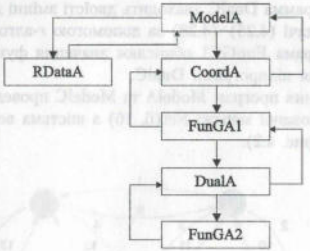


Рис. 4.1. Структура програми ModelA

За аналогічною схемою, як і на рис. 4.1, реалізована програма ModelC, де відповідні блоки виконують такі функції.

Головна програма ModelC керує процесом знаходження розв'язку задачі (4.20)–(4.24). В програмі встановлюються штрафні параметри (на основі вхідних даних), запускається алгоритм розв'язування задачі (4.20)–(4.24), аналізується отриманий розв'язок, результати розрахунку виводяться у файл протоколу роботи програми. Програма ModelC використовує підпрограми RdataC та CoordC.

Підпрограма RDataC читає вхідні дані задачі та перевіряє їх коректність. Вхідні дані читаються послідовно з чотирьох файлів, які містять: а) структуру орієнтованої мережі; б) обсяги потоків, що пересилаються по мережі; в) структуру пошкоджень мережі; г) допустимі шляхи для передачі потоків у мережі.

Підпрограма CoordC реалізує роботу r-алгоритму для розв'язування координуючої негладкої задачі, пов'язаної із схемою декомпозиції за змінними Y для задачі (4.20)–(4.24).

Підпрограма FunGC1 обчислює значення негладкої координуючої функції та її субградієнта, які використовуються програмою CoorDC. Для обчислення цих значень послідовно для кожного з пошкоджень використовується підпрограма DualC.

Підпрограмма DualC знаходить двоїсті змінні до обмежень (4.26) підзадачі (4.25)–(4.28) за допомогою r-алгоритму.

Підпрограма FunGC1 обчислює значення функції та субградієнта для підпрограми DualC.

Тестування програм ModelA та ModelC проведено на прикладі орієнтованої мережі Net(6, 16) з шістьма вершинами та 16 дугами (рис. 4.2).

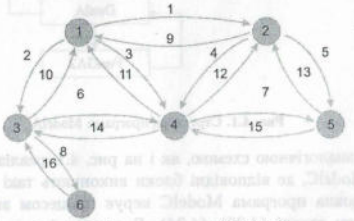


Рис. 4.2. Структура мережі Net(6, 16)

Вартості створення одиниці пропускної здатності для дуг (1,2) та (2,1) візьмемо 1,5, а для всіх інших дуг – 1. Будемо вважати, що між усіма парами вершин потрібно переслати по мережі один і той же обсяг потоку, що дорівнює десяти одиницям. Таким чином, множина $D = \{1, ..., 30\}$ складається з усіх можливих трійок $(r, s, 10)$, де $s = 1, ..., 6$, $s = 1, ..., 6$, $s \neq r$. Для мережі Net(6, 16) розглянемо такі варіанти множин $T = \{1, ..., |T|\}$ можливих пошкоджень:

а) відсутність пошкоджень ($|T| = 0$);

б) відмова будь-якої, але тільки одної дуги, за винятком дуг, зв'язаних з вершиною 6 ($|T| = 14$). Відзначимо, що у випадку повної відмови однієї з дуг (3,6) або (6,3) неможливо буде виконати вимоги до передачі потоків у мережі;

в) всього одне пошкодження ($|T| = 1$), коли одночасно відмовляють чотири дуги: (1,2), (2,1), (2,4) та (4,2);

г) структура пошкоджень мережі така, що із вхідних у кожну вершину дуг залишається тільки одна дуга, а решта вхідних дуг може відмовляти ($|T| = 14$);

д) побудований за типом варіанта г), де аналогічні відмови мають місце як для вхідних у вершину дуг, так і для вихідних з цієї вершини дуг ($|T| = 28$);

е) будь-яка з дуг, але тільки одна, може зменшити свою пропускну здатність у два рази ($|T| = 16$).

Мінімальні по сумарною вартістю пропускні здатності дуг мережі Net(6, 16) для варіантів а)–е) наведені в табл. 4.1 та 4.2. Вони одержані програмою ModelA відповідно при нульових значеннях y_{ij}^0 (табл. 4.1) та ненульових значеннях y_{ij}^0 (табл. 4.2).

У табл. 4.2 розглянуті варіанти наведено в тому ж порядку, що і в табл. 4.1. Виняток становить тільки те, що в табл. 4.2 відсутній варіант а). Це пов'язано з тим, що розрахунки в табл. 4.2 враховують, що y_{ij}^0 дорівнює значенню $y_{ij}^*(a)$ з табл. 4.1. Тому тривіальний розв'язок для варіанта а), що дорівнює нульовому вектору, в табл. 4.2 не наводиться.

Із табл. 4.1 та 4.2 видно, що чим складніша система пошкоджень у мережі, тим більші значення пропусних здатностей дуг потрібні, щоб можна було задовольнити всі вимоги на передачу потоків у мережі, як при відсутності пошкоджень у мережі, так і в тому випадку, коли буде мати місце одне і тільки одне, але будь-яке із заданого списку пошкодження орієнтованої мережі. Так, наприклад, витрати на пропускні здатності дуг для забезпечення надійності мережі Net(6, 16) при варіанті д) будуть майже в два з половиною рази більші,ніж тоді, коли пошкоджень у мережі Net(6, 16) немає (1250 одиниць проти 490 одиниць).

Таблиця 4.1. Оптимальні пропускні здатності дуг мережі Net(6, 16) при пошкодженнях а)-е) та нульових значеннях y_{ij}^0

N	(i, j)	c_{ij}	y_{ij}^0	y_{ij}^*					
				а	б	в	г	д	е
1	(1,2)	1,5	0	10	50	0	90	90	49,14
2	(1,3)	1,0	0	20	80	20	90	90	53,33
3	(1,4)	1,0	0	20	40	30	50	50	14,18
4	(2,4)	1,0	0	30	30	0	50	50	13,73
5	(2,5)	1,0	0	10	50	50	80	80	25,43
6	(3,4)	1,0	0	60	80	60	80	90	53,37
7	(4,5)	1,0	0	40	50	80	80	80	49,14
8	(3,6)	1,0	0	50	50	50	50	50	100,00
9	(2,1)	1,5	0	10	50	0	80	90	44,19
10	(3,1)	1,0	0	20	80	20	80	90	53,32
11	(4,1)	1,0	0	20	40	30	50	50	19,14
12	(4,2)	1,0	0	30	30	0	50	50	6,29
13	(5,2)	1,0	0	10	50	50	50	80	27,91
14	(4,3)	1,0	0	60	80	60	90	90	53,34
15	(5,4)	1,0	0	40	50	80	50	80	44,18
16	(6,3)	1,0	0	50	50	50	50	50	100,00
$\sum_{(i,j)\in A} c_{ij} y_{ij}^*$				490	910	580	1155	1250	753,4

Тестування програми ModelC проводилось на тих же даних, що і тестування програми ModelA. Очевидно, що якщо розглянути всі можливі шляхи для передачі потоків у мережі Net(6, 16) між усіма парами вершин, то для всіх варіантів а)–е) ми повинні отримати такі ж самі значення оптималь

Таблиця 4.2. Оптимальні пропускні здатності дуг мережі Net(6, 16) при пошкодженнях б)-е) та ненульових значеннях y_{ij}^0

N	(i, j)	c_{ij}	y_{ij}^0	y_{ij}^*				
				б	в	г	д	е
1	(1,2)	1,5	10	40	0	80	80	20,00
2	(1,3)	1,0	20	60	0	70	70	20,43
3	(1,4)	1,0	20	20	10	30	30	1,51
4	(2,4)	1,0	30	0	0	20	20	0,00
5	(2,5)	1,0	10	40	40	70	70	20,00
6	(3,4)	1,0	60	20	0	20	30	16,98
7	(4,5)	1,0	40	10	40	40	40	0,00
8	(3,6)	1,0	50	0	0	0	0	50,00
9	(2,1)	1,5	10	40	0	70	80	20,00
10	(3,1)	1,0	20	60	0	60	70	21,51
11	(4,1)	1,0	20	20	10	30	30	0,43
12	(4,2)	1,0	30	0	0	20	20	0,00
13	(5,2)	1,0	10	40	40	40	70	20,00
14	(4,3)	1,0	60	20	0	30	30	19,14
15	(5,4)	1,0	40	10	40	10	40	0,00
16	(6,3)	1,0	50	0	0	0	0	50,00
$\sum_{(i,j)\in A} c_{ij} y_{ij}^*$				420	180	665	760	280,00

них пропускних здатностей, як і наведені в табл. 4.1 та 4.2. Саме це підтвердили розрахунки для програми ModelC, коли розглядалися всі можливі шляхи (їх загальна кількість $|P| = 140$).

Другий варіант тестових розрахунків для ModelC проводився тоді, коли кількість шляхів для передачі потоків була

обмежена: між кожною парою вершин вибирались тільки два найкоротші шляхи, що не перетинаються по дугах (за винятком дуг, зв'язаних із вершиною 6). Цей набір дозволених шляхів для передачі потоків між парами вершин у мережі Net(6, 16) було задано таким чином:

(1→2) – (1,2), (1,4,2);
(1→3) – (1,3), (1,4,3);
(1→4) – (1,4), (1,2,4);
(1→5) – (1,2,5), (1,4,5);
(1→6) – (1,3,6), (1,4,3,6);
(2→3) – (2,1,3), (2,4,3);
(2→4) – (2,4), (2,1,4);
(2→5) – (2,5), (2,4,5);
(2→6) – (2,1,3,6), (2,4,3,6);
(3→4) – (3,4), (3,1,4);
(3→5) – (3,4,5), (3,1,2,5);
(3→6) – (3,6);
(4→5) – (4,5), (4,2,5);
(4→6) – (4,3,6), (4,1,3,6);
(5→6) – (5,4,3,6), (5,2,1,3,6).

Шлях $(i \to j)$ з вершини i в вершину j збігається (по вершинах) в зворотному напрямку з шляхом $(j \to i)$.

Мінімальні за вартістю пропускні здатності мережі Net(6, 16), одержані програмою ModelC, наведено в табл. 4.3. Тут розглядаються тільки варіанти а), б), е) при нульовому векторі y_{ij}^0 та варіанти б), е) при ненульовому векторі y_{ij}^0 ($y_{ij}^0 = y_{ij}^*(a)$). Для всіх інших варіантів неможливо забезпечити надійність мережі Net(6, 16) із-за структури пошкоджень у мережі.

З табл. 4.3 видно, що в більшості випадків мінімальні витрати на пропускні здатності дуг для забезпечення надійності мережі Net(6, 16) більші, ніж для задачі (4.6)–(4.10). Так, це має місце для всіх розглянутих варіантів, крім варіанта е) при ненульових значеннях наявного ресурсу пропускної здатності дуг, та обумовлено тим, що розглядається обмежене число шляхів для передачі потоків в мережі Net(6, 16).

Таблиця 4.3. Оптимальні пропускні здатності дуг мережі Net(6, 16) при обмеженому наборі шляхів для передачі

N	(i, j)	c_{ij}	y_{ij}^0	y_{ij}^*			y_{ij}^0	y_{ij}^*	
				а	б	е		б	е
1	(1,2)	1,5	0	40	0	41,34	10	30	20,00
2	(1,3)	1,0	0	60	0	52,21	20	60	29,28
3	(1,4)	1,0	0	20	10	20,71	20	30	2,34
4	(2,4)	1,0	0	0	0	19,40	30	10	0,00
5	(2,5)	1,0	0	40	40	30,00	10	40	20,00
6	(3,4)	1,0	0	20	0	55,90	60	20	15,32
7	(4,5)	1,0	0	10	40	40,00	40	10	0,00
8	(3,6)	1,0	0	0	0	100,00	50	0	150,00
9	(2,1)	1,5	0	40	0	41,21	10	30	20,00
10	(3,1)	1,0	0	60	0	52,05	20	60	22,34
11	(4,1)	1,0	0	20	10	21,00	20	30	9,28
12	(4,2)	1,0	0	0	0	19,33	30	10	0,00
13	(5,2)	1,0	0	40	40	30,00	10	40	20,00
14	(4,3)	1,0	0	20	0	55,58	60	20	1,43
15	(5,4)	1,0	0	10	40	40,00	40	10	0,00
16	(6,3)	1,0	0	0	0	100,00	50	0	15,00
$\sum_{(i,j)\in A} c_{ij} y_{ij}^*$				490	920	760,00		430	280,00

Кожній розв'язаній програмами ModelA та ModelC задачі (4.6)–(4.10) та задачі (4.20)–(4.24) відповідає задача лінійного програмування з конкретним числом змінних N та числом обмежень M. Для задачі (4.6)–(4.10) вони визначаються за формулами

$$N_a = |A| \times (|T| + 2) + |A| \times |D| \times (|T| + 1),$$

$$M_a = (|T|+2)\times(|A|+|D|\times|V|),$$

а для задачі (4.20)–(4.24) – за формулами

$$N_c = |A|\times(|T|+2)+|P|\times(|T|+1),$$

$$M_c = (|T|+1)\times(|A|+|D|).$$

В табл. 4.4 наведені характеристики роботи програм ModelA та ModelC для задач лінійного програмування, що відповідають варіантам а)–е) на мережі Net(6, 16) при нульових y_{ij}^0. Для задачі (4.20)–(4.24) розглядалися всі можливі шляхи для передачі потоків в мережі Net(6, 16) між усіма парами вершин. Тому оптимальні значення функції для задач (4.6)–(4.10) та задач (4.20)–(4.24), що відповідають варіантам, збігалися. Тут iter – затрати (в ітераціях) r-алгоритму з адаптивним регулюванням крокового множника для розв'язання координуючої задачі негладкої мінімізації для схеми декомпозиції за змінними; time – затрачений час на розв'язування задач програмами ModelA та ModelC на IBM PC / Pentium 750 MHz. Наряду з характеристиками роботи програм ModelA та ModelC наведені розміри задач лінійного програмування (кількість змінних та кількість обмежень), що відповідають варіантам а)–е).

Таблиця 4.4. Характеристики роботи програм

Варіант	ModelA				ModelC			
	N_a	M_a	iter	time (с)	N_c	M_c	iter	time (с)
а)	512	392	152	10	172	46	151	6
б)	7456	3136	224	110	2356	690	206	94
в)	1008	588	139	17	328	92	165	12
г)	7456	3136	139	15	2356	690	178	66
д)	14400	5880	274	45	4540	1334	158	102
є)	8448	3528	159	111	2668	782	188	99

Задачі лінійного програмування, що відповідають варіантам а)–е) для мережі Net(6, 16), мають відносно невеликі розміри і можуть бути розв'язані за допомогою стандартних ЛП-програм (програм лінійного програмування), наприклад CPLEX, MINOS тощо. Для більших розмірів мережі значно зростатимуть і розміри відповідних задач лінійного програмування. На таких задачах програми ModelA та ModelC можуть виявитися швидшими за стандартні ЛП-програми. Так, задача (4.6)–(4.10) для надійної орієнтованої мережі з 15 вершинами та 30 дугами, якій відповідає задача лінійного програмування з числом змінних $N = 196260$ та числом обмежень $M = 101760$, була розв'язана за допомогою програми ModelA за 58 хв на IBM PC / Pentium 750 MHz.

4.2. Проектування оптимальної логічної структури надійної мережі

Розглянемо деякі задачі існування та побудови такої надійної логічної структури комунікаційної мережі, за якої витрати по експлуатації мережі є мінімальними.

Опишемо таку модель комунікаційної мережі. Нехай задано граф $G(V, E)$, де V і E – відповідно множини вершин і дуг (неорієнтованих), і введено поняття комунікаційної лінії як одиниці виміру пропускної здатності дуг. Таким чином, кожна дуга $e \in E$ графа G характеризується числом c_e комунікаційних ліній, що їй відповідають. Позначимо SP множину шляхів між усіма парами вершин графа G.

Вважатимемо заданою також множину D запитів (наприклад, інформаційних запитів у мережі зв'язку, транспортних запитів у мережі шляхів сполучення). Позначимо d_k, $k = 1, ..., |D|$, значення повного запиту (повного потоку), яке подається відповідною кількістю комунікаційних ліній, що проводяться крізь мережу між двома даними вершинами s_k і r_k з множини V, тобто $d_k = d_k(s_k, r_k)$.

Введемо поняття фізичної і логічної структур комунікаційної мережі.

Означення 4.1. Фізична структура $PS = PS(G(V, E), c_e, e \in E)$ мережі цілком визначається графом $G(V, E)$ і числами c_e, $e \in E$.

Означення 4.2. Логічна структура $LS = LS(SP, x_p, p \in SP)$ мережі цілком визначається множиною SP шляхів між усіма парами вершин графа G із зазначенням кількості комунікаційних ліній x_p у кожному шляху $p \in SP$.

Для будь-якого шляху $p \in SP$ і будь-якої дуги $e \in E$ введемо логічну функцію

$$\chi\{p,e\} = \begin{cases} 1, & e \in p, \\ 0, & e \notin p. \end{cases}$$

Означення 4.3. Логічна структура $LS = LS(SP, x_p, p \in SP)$ відповідає фізичній структурі мережі, якщо виконуються такі співвідношення:

$$\sum_{p \in SP} x_p \chi\{p,e\} \le c_e, \; e \in E. \tag{4.29}$$

Нехай $SP(i, j) \subset SP$ – множина шляхів між вершинами $i, j \in V$, $i \neq j$.

Означення 4.4. Логічна структура $LS = LS(SP, x_p, p \in SP)$ задовольняє запити D, якщо

$$\sum_{p \in SP(s_k, r_k)} x_p \ge d_k, \; k = 1, \ldots, |D|. \tag{4.30}$$

Зупинимось на деяких питаннях, пов'язаних з відмовами компонент комунікаційної мережі, а саме її дуг.

Припустимо, що задано список аварійних ситуацій TL, в якому перераховані можливі відмови компонентів. Кожний елемент $t_r \in TL$, $r = 1, \ldots, |TL|$, цього списку є набір, що складається з $q(r)$ компонентів $e_{i_1}, \ldots, e_{i_{q(r)}}$ – дуг, які виходять з ладу при r-й аварійній ситуації. Вважаємо, що в мережі одночасно може скластися тільки одна аварійна ситуація.

Будь-яка задана фізична структура PS мережі визначає певну множину $LSS(PS)$ логічних структур, які відповідають цій фізичній структурі PS і задовольняють запити D. Очевидно,

довільна аварійна ситуація $t_r \in TL$ перетворює фізичну структуру: $PS \to PS'(t_r)$.

Будемо вважати логічну структуру $LS \in LSS(PS)$ надійною щодо множини аварійних ситуацій TL, якщо при будь-якій аварійній ситуації $t_r \in TL$ ця логічна структура може бути перебудована в деяку логічну структуру $LS' \in LSS(PS'(t_r))$, що відповідає фізичній структурі мережі PS', отриманій внаслідок аварійної ситуації t_r і задовольняє запити D. Тут мова йде не про існування нової логічної структури $LS' \in LSS(PS'(t_r))$, а про можливість перебудови існуючої структури LS при аварійній ситуації. Наприклад, у випадку мереж зв'язку це пов'язано з тим, що при виході з ладу деяких їх компонентів не змінюються всі інформаційні потоки, а тільки ті, які пов'язані з пошкодженими компонентами системи. Перетворення інформаційних потоків відбувається відповідно до прийнятого в мережі зв'язку алгоритму відновлення (Self-Healing Algorithm – див., наприклад, [105]).

Означення 4.5. Набір логічних структур $RLS = \{LS(t_r), LS(t_r) \in LSS(PS'(t_r)), t_r \in TL\}$ назвемо надійним відносно списку TL, якщо логічні структури $LS(t_r)$ породжуються алгоритмом відновлення з вихідної логічної структури LS при аварійній ситуації t_r.

Серед різноманітних алгоритмів відновлення можна виділити, як найбільш гнучкі, алгоритми відновлення дуги і шляху. Будемо розглядати алгоритми відновлення, що базуються на алгоритмі відновлення шляху.

Для будь-якої аварійної ситуації $t_r = \{e_{i_1}, \ldots, e_{i_{q(r)}}\}$, $r = 1, \ldots, |TL|$, позначимо $SP(i, j, t_r)$ множину шляхів між парою вершин $i, j \in V$, $i \neq j$, що не проходять через дуги $e_{i_1}, \ldots, e_{i_{q(r)}}$; $y_p(t_r)$ – кількість додаткових комунікаційних ліній на шляху $p \in SP$ у логічній структурі $LS(t_r)$; $SP(t_r) = \bigcup_{i,j \in V, i \neq j} SP(i, j, t_r)$.

Покладемо $\chi\{p, t_r\} = \min\left(\sum_{j=1}^{q(r)} \chi\{p, e_{i_j}\}, \; 1\right)$.

Нехай задано фізичну структуру $PS = PS(G(V,E), c_e, e \in E)$ деякої комунікаційної мережі, множина запитів D і список аварійних ситуацій TL. Постає питання, чи існує надійний відносно списку TL набір логічних структур $RLS = \{LS(t_r), LS(t_r), t_r \in TL\}$ цієї мережі. Якщо перейти від цілочислових змінних до неперервних x_p і $y_p(t_r)$, де $p \in SP$, $t_r \in TL$, то відповідь можна одержати, розв'язавши таку задачу:

знайти

$$\min_{\eta \in R^1, x_p, y_p(t_r), p \in SP, t_r \in TL} \eta \qquad (4.31)$$

при обмеженнях

$$\sum_{k=1}^{|D|} \sum_{p \in SP(s_k, r_k)} x_p \chi\{p,e\} \le c_e + \eta, \ e \in E, \qquad (4.32)$$

$$\sum_{p \in SP(s_k, r_k)} x_p \ge d_k, \ k = 1,\ldots,|D|, \qquad (4.33)$$

$$x_p \ge 0, \ p \in SP, \qquad (4.34)$$

$$\sum_{p \in SP(s_k, r_k)} x_p \chi\{p,t_r\} = \sum_{p \in SP(s_k, r_k, t_r)} y_p(t_r), \ k = 1,\ldots,|D|, \ t_r \in TL, \qquad (4.35)$$

$$\sum_{k=1}^{|D|} \sum_{p \in SP(s_k, r_k, t_r)} \left(x_p + y_p(t_r)\right) \chi\{p,e\} \le c_e + \eta, \ e \in E, \ t_r \in TL, \qquad (4.36)$$

$$y_p(t_r) \ge 0, \ p \in SP, \ t_r \in TL. \qquad (4.37)$$

$$y_p(t_r) = 0, \ p \in SP \setminus SP(t_r), \ t_r \in TL. \qquad (4.38)$$

Виконання обмежень (4.32)–(4.34) забезпечує існування вихідної логічної структури LS, яка відповідає фізичній структурі $LS(t_r)$ мережі та задовольняє вимоги з множини D. Співвідношення (4.35) гарантують для кожної аварійної ситуації збіг суми додаткових комунікаційних ліній із сумою

комунікаційних ліній, розташованих на шляхах, що проходять через ушкоджені ребра. Тим самим забезпечується виконання запитів D при кожній логічній структурі $LS(t_r)$. Обмеження (4.36) гарантують відповідність логічних структур $LS(t_r)$, $t_r \in TL$, фізичній структурі мережі. Умови (4.37), (4.38) відповідають алгоритму відновлення шляху.

Нехай η^* – оптимальний розв'язок задачі (4.31)–(4.38). Якщо $\eta^* \le 0$, то існує надійний відносно списку TL набір логічних структур $RLS = \{LS, LS(t_r), t_r \in TL\}$, у протилежному випадку такого набору немає. Позначимо $F(LS)$ експлуатаційні витрати при функціонуванні мережі з логічною структурою LS.

Для конкретної заданої множини аварійних ситуацій TL становить інтерес така задача пошуку надійної структури LS, яка забезпечує мінімальні експлуатаційні витрати по мережі: знайти arg min $F(LS)$ за умов, що структура LS є надійною відносно аварійних ситуацій TL, $LS \in LSS(PS)$, PS – задана фізична структура мережі.

Для деякої фізичної структури мережі і заданих коефіцієнтів f_e, $e \in E$, розглянемо функцію $F(LS)$ у вигляді

$$F(LS) = \sum_{e \in E} f_e \left(\sum_{k=1}^{|D|} \sum_{p \in SP(s_k, r_k)} x_p \chi\{p,e\} \right).$$

Тоді задача про мінімізацію експлуатаційних витрат може бути сформульована в такий спосіб:

знайти

$$\min_{x_p, y_p(t_r), p \in SP, t_r \in TL} \sum_{e \in E} f_e \left(\sum_{k=1}^{|D|} \sum_{p \in SP(s_k, r_k)} x_p \chi\{p,e\} \right) \qquad (4.39)$$

при обмеженнях

$$\sum_{k=1}^{|D|} \sum_{p \in SP(s_k, r_k)} x_p \chi\{p,e\} \le c_e, \ e \in E, \qquad (4.40)$$

$$\sum_{p\in SP(s_k,r_k)} x_p \ge d_k,\ k=1,\ldots,|D|, \tag{4.41}$$

$$x_p \ge 0,\ p \in SP, \tag{4.42}$$

$$\sum_{p\in SP(s_k,r_k,t_r)} x_p \chi\{p,t_r\} = \sum_{p\in SP(s_k,r_k,t_r)} y_p(t_r),\ k=1,\ldots,|D|,\ t_r \in TL, \tag{4.43}$$

$$\sum_{k=1}^{|D|} \sum_{p\in SP(s_k,r_k,t_r)} \left(x_p + y_p(t_r)\right)\chi\{p,e\} \le c_e,\ e \in E,\ t_r \in TL, \tag{4.44}$$

$$y_p(t_r) \ge 0,\ p \in SP,\ t_r \in TL, \tag{4.45}$$

$$y_p(t_r) = 0,\ p \in SP \setminus SP(t_r),\ t_r \in TL. \tag{4.46}$$

Розглянемо можливі підходи до розв'язання задач (4.31)–(4.38) та (4.39)–(4.46).

Якщо зв'язати множники Лагранжа з обмеженнями (4.32), (4.36) або (4.40), (4.44), то розв'язання задач (4.31)–(4.38) та (4.39)–(4.46) можна звести до розв'язання недиференційованих двоїстих задач за допомогою алгоритмів субградієнтного типу [45]. При цьому процес розв'язання полягає в розв'язуванні незалежних підзадач у кількості $|D|$.

Якщо число шляхів $|SP(s_k, r_k)|$, $k = 1, .., |D|$, не є дуже великим (~1000), то для розв'язання задач (4.31)–(4.38) та (4.39)–(4.46) можна застосовувати схему декомпозиції за змінними x_p (див. розділ 2). Зауважимо, що в цьому випадку для розв'язання дискретних задач можна ефективно застосовувати методи локального пошуку (див. розділ 3).

4.3. Задача модернізації надійних мереж

Розглядається задача модернізації мереж (транспортних, комунікаційних і т.п.), що мають надійно працювати як у номінальному, так і в аварійному режимі. Модернізацію розуміємо як додавання в мережу нових дуг, аварійний режим – як вихід з ладу деяких дуг мережі.

Вважається заданим орієнтований граф $G = (V, E)$, де $V = (1, .., |V|)$– множина вершин; $E = (1, .., |E|)$– множина дуг, в яку входять як вже існуючі в мережі, так і можливі нові дуги; E^0 – множина вже існуючих дуг, $E^0 \subseteq E$. Розглядається задача вибору розширеної підмножини дуг $\overline{E}$, $E^0 \subseteq \overline{E} \subseteq E$ що поліпшує якість функціонування мережі. Поставимо у відповідність кожній дузі $(i, j) \in E$ змінну $y_{ij} = (0 \vee 1)$ так, що $y_{ij} = 1$, якщо дуга входить у підмножину $\overline{E}$, інакше – $y_{ij} = 0$. Якщо дуга (i, j) вже існує в мережі, то $y_{ij} = 1$. Дуги можуть знаходитися в робочому або аварійному стані. По дугах, що знаходяться в робочому стані, можуть передаватися потоки продуктів різних типів. Величина потоку невід'ємна, напрямок збігається з орієнтацією дуги. Частина дуг характеризується пропускною здатністю, для інших пропускна здатність вважається необмеженою. Вершини можуть бути джерелами, споживачами продуктів або перерозподіляти потоки між дугами. Позначимо R множину типів продуктів. Для кожної вершини $i \in V$ вважається заданою величина B_i^r, $r \in R$ (потреба в продукті, якщо $B_i^r < 0$ або виробництво продукту, якщо $B_i^r > 0$). При цьому

$$\sum_{i\in V} B_i^r = 0,\quad r \in R. \tag{4.47}$$

Будемо вважати заданим список $T = (1,..,|T|)$ аварійних ситуацій. Кожен елемент $t \in T$ містить перелік тих дуг $(i, j) \in E$ (одну або кілька), що виходять з ладу при t-й аварійній ситуації. Передбачається, що при настанні будь-якої аварійної ситуації мережа залишається зв'язною. Будемо розглядати функціонування мережі в безаварійному режимі і при настанні однієї (будь-якої) аварійної ситуації $t \in T$. У випадку настання аварійної ситуації $t \in T$ потоки, що проходять у безаварійному режимі по дугах, що відповідають пошкодженню t, повинні бути перерозподілені по дугах мережі, що залишилися. З кожною аварійною ситуацією $t \in T$ зв'язана деяка розширена су-

купність дуг O_t – оточення ситуації t. Потоки з оточення O_t також перерозподіляються при настанні аварійної ситуації $t \in T$. Потоки по інших дугах мережі можуть бути подані у вигляді суми потоку до аварійної ситуації і величини перерозподілу.

Позначимо:

x_{ij}^r – потік r-го типу продукту, переданий по дузі $(i, j) \in E$, $r \in R$, $x_{ij}^r \geq 0$;

x_{ij}^{rt} – потік -го типу продукту, переданий по дузі $(i, j) \in E$, $r \in R$ при настанні аварійної ситуації $t \in T$, $x_{ij}^{rt} \geq 0$;

y_{ij} – змінна, що визначає, чи включається дуга $(i, j) \in E$ у підмножину дуг $\bar{E}$ мережі, що модернізується, $y_{ij} = (0 \vee 1)$;

E^L – підмножина дуг з обмеженою пропускною здатністю $E^L \subset E$;

d_{ij} – пропускна здатність дуги $(i, j) \in E^L$;

$P(i)$ – множина вершин, зв'язаних з вершиною дугами, що виходять з вершини i, $(i, j) \in E$;

$Q(i)$ – множина вершин, зв'язаних з вершиною дугами, що входять до вершини i, $(i, j) \in E$;

$P^t(i)$ – множина вершин, зв'язаних з вершиною дугами, що виходять з вершини i, $(i, j) \in E \backslash \{i', j'\} \in t$;

$Q^t(i)$ – множина вершин, зв'язаних з вершиною дугами, що входять до вершини i, $(i, j) \in E \backslash \{i', j'\} \in t$.

Будемо припускати, що визначені коефіцієнти c_{ij}^r, c_{ij}^{rt}, b_{ij} які мають такий зміст:

c_{ij}^r – вартість передачі одиниці продукту r-го типу по дузі $(i, j) \in E$;

c_{ij}^{rt} – вартість передачі одиниці аварійного перерозподілу продукту r-го типу по дузі $(i, j) \in E$ при настанні аварійної ситуації $t \in T$;

b_{ij} – витрати на створення дуги $(i, j) \in E$, якщо дуга (i, j) уже є в мережі, то $b_{ij} = 0$.

Позначимо:

$$x = (x_{ij}^r, x_{ij}^{rt}, (i,j) \in V, r \in R, t \in T), \; y = (y_{ij}, (i,j) \in V).$$

Передбачається, що загальні витрати на функціонування мережі зображені у вигляді

$$f(x,y) = \sum_{(i,j)\in V} \sum_{r\in R} c_{ij}^r x_{ij}^r + \sum_{t\in T} \sum_{(i,j)\in V} \sum_{r\in R} c_{ij}^{rt} x_{ij}^{rt} + \sum_{(i,j)\in V} b_{ij} y_{ij}.$$

Із врахуванням введених позначень задачу модернізації мереж можна записати таким способом:

знайти

$$q^* = \min\left\{ \sum_{(i,j)\in V} \sum_{r\in R} c_{ij}^r x_{ij}^r + \sum_{t\in T} \sum_{(i,j)\in V} \sum_{r\in R} c_{ij}^{rt} x_{ij}^{rt} + \sum_{(i,j)\in V} b_{ij} y_{ij} \right\} \quad (4.48)$$

при обмеженнях

$$\sum_{j\in P(i)} x_{ij}^r - \sum_{j\in Q(i)} x_{ji}^r = B_i^r, \; i \in V, \; r \in R, \quad (4.49)$$

$$\sum_{r\in R} x_{ij}^r \leq d_{ij} y_{ij}, \quad (i,j) \in E^L, \quad (4.50)$$

$$X_i^{rt} = B_i^r - \left(\sum_{j\in P(i)\backslash O_t} x_{ij}^r - \sum_{j\in Q(i)\backslash O_t} x_{ji}^r \right), \; i \in V, \; r \in R, \quad t \in T, \quad (4.51)$$

$$\sum_{j\in P^t(i)} x_{ij}^{rt} - \sum_{j\in Q^t(i)} x_{ji}^{rt} = X_i^{rt}, \quad i \in V, \quad r \in R, \quad t \in T, \quad (4.52)$$

$$\sum_{r\in R} x_{ij}^{rt} \leq d_{ij} y_{ij}, \quad (i,j) \in O_t, \; t \in T, \quad (i,j) \in E^L, \quad (4.53)$$

$$\sum_{r\in R} x_{ij}^{rt} \leq d_{ij} y_{ij} - \sum_{t\in R} x_{ij}^r, \quad (i,j) \in E \setminus O_t, \; t \in T, \quad (i,j) \in E^L, \quad (4.54)$$

$$x_{ij}^r \geq 0, \quad x_{ij}^{rt} \geq 0, \quad (i,j) \in E, \quad r \in R, \quad t \in T, \quad (4.55)$$

$$0 \leq y_{ij} \leq 1, \quad (i,j) \in E, \quad (4.56)$$

$$x_{ij}^{rt} = 0, \quad (i,j) \in t, \quad r \in R, \quad t \in T, \quad (4.57)$$

$$y_{ij} = (0 \vee 1), \quad (i, \quad j) \in E. \tag{4.58}$$

Обмеження (4.49), (4.50) відповідають безаварійному режиму роботи мережі – потреби вершин у продуктах мають бути задоволені, сумарний потік по кожній дузі не перевищує пропускної здатності. Обмеження (4.51), (4.52) визначають перерозподіл потоків, зв'язаний з довільною аварійною ситуацією $t \in T$ (перерозподіляються потоки, що проходять по дугах околу O_t). Обмеження (4.53), (4.54) – обмеження по пропускних здатностях дуг у випадку аварійних ситуацій, у (4.54) враховуються потоки безаварійного режиму мережі.

У телекомунікаційних мережах кожен тип продукту r відповідає потокові повідомлень з деякого джерела $s(r) \in V$ у стік $p(r) \in V$, тобто

$$B^r_{s(r)} > 0, \quad B^r_{p(r)} = -B^r_{s(r)}, \quad B^r_i = 0, \quad i \neq s(r), \quad i \neq p(r). \tag{4.59}$$

Більше того, всі повідомлення часто передаються по тому самому маршруту. Для врахування цієї додаткової умови досить вимоги, щоб

$$x^r_{ij}, \quad x^{rt}_{ij} = 0 \vee B^r_{s(r)}, \quad (i, j) \in E, \quad t \in T, \quad r \in R. \tag{4.60}$$

Сформульована задача є задачею великої розмірності, і для її розв'язання не завжди можуть бути використані стандартні засоби програмного забезпечення. При розробці спеціального програмного забезпечення необхідно враховувати специфіку задачі. Наприклад, можна вважати, що число дуг, які додаються, порівняно невелике, а по пропускних здатностях враховуються лише суттєві обмеження (вузькі місця, дуги, що додаються). Таких (активних) обмежень також може бути небагато. Аварійні ситуації будемо розглядати тільки найбільш ймовірні. При таких припущеннях, коли число обмежень, зв'язаних із пропускними здатностями дуг, не перевищує 300, для розв'язання задачі (4.48)–(4.58), (4.59) можуть використовуватися методи декомпозиції за цими обмеженнями.

Визначимо множники Лагранжа $u_{ij} \geq 0$ для обмежень (4.50), для обмежень (4.53), $u^t_{ij} \geq 0$ (4.54). Функцію Лагранжа $L(x, y, u)$

для зазначених обмежень можна звести до такого вигляду:

$$L(x,y,u) = \sum_{(i,j)\in V}\sum_{r\in R} c^r_{ij}(u)x^r_{ij} + \sum_{t\in T}\sum_{(i,j)\in V}\sum_{r\in R} c^{rt}_{ij}(u)x^{rt}_{ij} + \sum_{(i,j)\in V} b_{ij}(u)y_{ij}, \tag{4.61}$$

де

$$c^r_{ij}(u) = c^r_{ij} + u_{ij} + \sum(u^t_{ij} \mid \ t : (i,j) \notin O_t);$$

$$c^{rt}_{ij}(u) = \begin{cases} c^{rt}_{ij} + u^t_{ij}, & (i,j) \notin t; \\ c^{rt}_{ij}, & (i,j) \notin t; \end{cases}$$

$$b_{ij}(u) = b_{ij} - d_{ij}\left(u_{ij} + \sum(u^t_{ij} \mid \ t : (i,j) \notin t)\right).$$

Позначимо S множину значень пари (x, y), що задовольняють обмеження (4.49), (4.51), (4.52), (4.55)–(4.58):

$$w(u) = \min\{L(x,y,u) : \ (x,y) \in S\}, \tag{4.62}$$

$$w^* = \max\{w(u) : \ u \geq 0\}. \tag{4.63}$$

Величина w^* є оцінкою знизу оптимального значення q^* і може використовуватися в алгоритмах гілок і границь. Оптимальні розв'язки задач (4.62), (4.63) можуть використовуватися в евристичних алгоритмах побудови наближених допустимих розв'язків початкової задачі.

Задача (4.63) є задачею опуклого програмування, і для її розв'язання можуть використовуватися субградієнтні алгоритми. Ефективність цих алгоритмів залежить від ефективності алгоритмів розв'язання задачі (4.62). Ця задача є задачею лінійного програмування великої розмірності, яка має блочну структуру і може бути подана у вигляді

$$w(u) = \sum_{r\in R} L^r(u) +$$

$$+\min_y\left\{\sum_{(i,\ j)\in E} b_{ij}(u)y_{ij} : y_{ij} = (0 \vee 1),\ (i,\ j) \in E \setminus E^0,\ y_{ij} = 1,\ (i,\ j) \in E^0\right\}, \tag{4.64}$$

де $L^r(u)$ – оптимальне значення підзадачі (окремого блока) для r-го типу продукту, що має такий вигляд (індекс r опущений):

знайти

$$L(u) = \min\left\{\sum_{(i,j)\in E} c_{ij}(u)x_{ij} + \sum_{t\in T}\left[\sum_{(i,j)\in E\setminus t} c_{ij}^t(u)x_{ij}^t\right]\right\}, \quad (4.65)$$

при обмеженнях

$$\sum_{j\in P(i)} x_{ij} - \sum_{j\in Q(i)} x_{ji} = B_i, \quad i\in V, \quad (4.66)$$

$$X_i^t = B_i - \left(\sum_{j\in P(i)\setminus O_t} x_{ij} - \sum_{j\in Q(i)\setminus O_t} x_{ji}\right), \quad i\in V, \quad t\in T, \quad (4.67)$$

$$\sum_{j\in P(i)\setminus t} x_{ij}^t - \sum_{j\in Q(i)\setminus t} x_{ji}^t = X_i^t, \quad i\in V, \quad t\in T, \quad (4.68)$$

$$x_{ij}\ge 0, \quad x_{ij}^t \ge 0, \quad (i,j)\in E, \quad t\in T. \quad (4.69)$$

Значення змінних для задачі (4.64) визначаються тривіально.

У загальному випадку для кожного блока можуть застосовуватись сучасні алгоритми лінійного програмування. В деяких окремих випадках для кожного блока можуть бути розроблені спеціальні ефективні алгоритми.

4.4. Задачі оптимізації мереж із врахуванням неповної інформації

У даному підрозділі наведено математичні моделі для знаходження оптимальних пропускних здатностей та потоків, що є узагальненням задач, розглянутих у п. 4.1.1, 4.1.2, на випадок, коли такі характеристики мережі як вимоги до потоків та ступінь працездатності ребер носять прогнозний характер. У

прикладних задачах на мережах вихід з ладу окремих ланок чи будь-яка інша причина, що призводить до суттєвих змін стану мережі, є, як правило, подією випадковою. Таку ситуацію можна описати в термінах стохастичного програмування – глибоко розробленої теорії (див., наприклад, [6, 118, 163, 164, 89, 106, 131, 144]).

Задачі з неповною інформацією часто виникають у перевантажених транспортних та телекомунікаційних мережах, де час подорожі (чи передачі інформації) в найкращому випадку апріорно відомий лише з неточністю. В таких стохастичних мережах можна спробувати скористатися апріорно відомим найменшим очікуваним часом подорожі чи спробувати поліпшити вибір маршрутів, коли час пересування по пройдених дугах та моменти прибуття в проміжні пункти відомі (див. [134]).

Цікава добірка моделей адаптивного перерозподілу потоків наведена в праці [83], де розглянута мережа, що знаходиться в одному із скінченної множини станів, переходи між якими здійснюються у відповідності до марківського процесу з неперервним часом. Кожному стану відповідає свій час проходження кожної дуги мережі. Розроблено стратегію, яка мінімізує очікуваний час подорожі. Розглянуті моделі зводяться до так званих *двохетапних задач стохастичного програмування* (див., наприклад, [6]), в яких задача другого етапу описує перерозподіл потоків у перехідному періоді.

4.4.1. Математичні моделі

Нехай задано структуру мережі. Будемо вважати, що вимоги до потоків, ступінь працездатності дуг, витрати на передачу одиниці потоку є дискретними стохастичними величинами, тобто відомі тільки їх можливі значення з відповідними ймовірностями. Необхідно знайти пропускні здатності та оптимально маршрутизувати потоки в мережі, коли відомі тільки прогнозні значення параметрів мережі.

Розглянемо мережу з множиною вузлів N та множиною дуг $A\subset N\times N$. Будемо вважати, що вимоги до потоків, ступінь працездатності дуг, витрати на передачу одиниці потоку

є детермінованими функціями випадкового параметра $l \in \{0, ..., L\}$, який будемо називати *сценарієм*. Сценарій показує, в якому з можливих станів може знаходитись мережа: для кожного l вимоги до потоків дорівнюють d_{rsl}; параметр, який задає ступінь працездатності ребра (i, j), має значення $\mu_{ijl}(0 \le \mu_{ijl} \le 1)$; витрати на передачу одиниці потоку по дузі (i, j) дорівнюють q_{ijl}, $(i, j) \in A$. Ймовірність реалізації сценарію l дорівнює p_l. Оптимальні потоки відшукуються, виходячи з *критерію мінімального математичного очікування витрат на їх передачу* за всіма сценаріями.

Наведемо формальний опис моделі, яка певною мірою є узагальненням задачі (4.1)–(4.5):

знайти

$$\min \sum_{(i,j)\in A} c_{ij} y_{ij} + \sum_l p_l \left[\sum_{(r,s)\in D} \sum_{(i,j)\in A} q_{ijl} x_{ijl}^{rs} + \sum_{(r,s)\in D} q_{rs}^{+} s_{rs}^{l} \right],$$

при обмеженнях

$$0 \le y_{ij} \le u_{ij}, \ (i,j) \in A,$$

$$\sum_{j:(i,j)\in A} x_{ijl}^{rs} - \sum_{j:(j,i)\in A} x_{jil}^{rs} = \begin{cases} d_{rsl} - s_{rs}^{l}, & i = r, \\ 0, & i \ne r, s, \\ s_{rs}^{l} - d_{rsl}, & i = s, \end{cases} \quad l \in L, \ (r,s) \in D, \tag{4.70}$$

$$\sum_{(r,s)\in D} x_{ijl}^{rs} \le \mu_{ijl}(y_{ij0} + y_{ij}), \ l \in L, (i,j) \in A,$$

$$x_{ijl}^{rs} \ge 0, s_{rs}^{l} \ge 0, \ l \in L, (i,j) \in A, (r,s) \in D.$$

Тут s_{rs}^{l} – кількість вимог, що не були виконані для пари (r, s) при реалізації l-го сценарія; q_{rs}^{+} – штрафи за невиконання вимог.

Модель (4.70) формально подається як задача лінійного програмування спеціальної структури. Вона подібна до лінійних двоетапних задач стохастичного програмування з дис-

кретними стохастичними параметрами: технологічною матрицею, правими частинами та цільовими коефіцієнтами другого етапу (див., наприклад, [6, 118]). Змінні y_{ij} можна розглядати як змінні першого етапу, які вибираються, коли ще невідомі реалізації стохастичних параметрів мережі, але з врахуванням можливих сценаріїв. Потоки x_{ijl}^{rs} можна розглядати як змінні другого етапу, коли вже відома реалізація конкретного сценарію та пропускні здатності з першого етапу.

Модель (4.70) можна модифікувати для опису можливих аварійних ситуацій в мережі, інформація про які носить прогнозний характер. Тут під аварійною ситуацією можна розуміти не тільки повний вихід з ладу окремих ребер мережі, а й зменшення пропускних здатностей ребер. У цьому випадку аварійну ситуацію можна характеризувати стохастичним параметром $0 \le \mu_{ij}^{l} \le 1$, $(i, j) \in A$ який задає ступінь працездатності ребра (i, j). Коли $\mu_{ij}^{l} \ne 0$ то μ_{ij}^{l} характеризує різні аварійні ситуації. Очевидно, коли пропускна здатність ребра дорівнює нулеві, це рівносильно повному виходу цього ребра з ладу.

Аналогічно можна узагальнити і модель (4.15)–(4.19) з п. 4.1.2 (тут змінними другого етапу будуть x_{kl}^{rs} та s_{rs}^{l}, де x_{kl}^{rs} – невідоме значення частини обсягу потоку d_{rs}, який буде пропущений шляхом $k \in K(r, s)$ при l-му сценарії; s_{rs}^{l} – кількість вимог, що не були виконані для пари (r, s) при l-му сценарії):

$$\min \sum_{(i,j)\in A} c_{ij} y_{ij} + \sum_l p_l \left[\sum_{(r,s)\in D} \sum_{k\in P(r,s)} q_{kl} a_{kl}^{rs} x_{kl}^{rs} + \sum_{(r,s)\in D} q_{rs}^{+} s_{rs}^{l} \right], \tag{4.71}$$

$$\sum_{k\in P(r,s)} x_{kl}^{rs} + s_{rs}^{l} = d_{rs}, \ l \in L, (r,s) \in D,$$

$$\sum_{(r,s)\in D} \sum_{k\in P(r,s)} a_k^{rs} x_{kl}^{rs} \le \mu_{ijl}(y_{ij0} + y_{ij}), \ l \in L, (i,j) \in A,$$

$$x_{kl}^{rs} \ge 0, \ l \in L, (r,s) \in D, k \in P(r,s).$$

$$0 \le y_{ij} \le u_{ij}, \ (i,j) \in A,$$

$$s^l_{rs} \geq 0.$$

Вектор $a^{rs}_k \in Z_n$ – вектор інцидентності, який визначається так:

$$(a^{rs}_k)_j = \begin{cases} 1, & \text{якщо } j \in k, \\ 0, & \text{в протилежному випадку.} \end{cases}$$

Моделі (4.70), (4.71) є задачами лінійного програмування великої вимірності з блочно-діагональною структорою матриць обмежень та із зв'язуючими змінними y_{ij}. Методи, засновані на використанні схем декомпозиції по змінних та алгоритмів негладкої оптимізації (наприклад, *r*-алгоритму), можуть бути ефективно застосовані для розв'язання цього класу задач (див. розділи 1, 2).

4.4.2. Розв'язування двохетапних задач стохастичного програмування на основі алгоритмів недиференційованої оптимізації

Методи для розв'язування двохетапних задач стохастичного програмування умовно можна поділити на два класи. Якщо випадкові параметри моделі мають скінченний дискретний розподіл ймовірностей, задачу можна подати як детерміновану, і застосовувати весь арсенал методів детермінованої оптимізації (див., наприклад, [147]). Крім того, можна використовувати стохастичні методи оптимізації [6], але вони, як правило, характеризуються повільнішою збіжністю. Нижче розглядається підхід до розв'язування двохетапних задач стохастичного програмування на основі методів недиференційовної оптимізації. Слід додати, що такий підхід дозволяє розглянути й випадок, коли функціонал задачі першого етапу є нелінійним, наприклад, квадратичним чи кусочно-лінійним.

Розглянемо таку стохастичну задачу:
знайти

$$\min_x (c, x) \tag{4.72}$$

при обмеженнях

$$T(\omega)x = h(\omega), \quad x \in X, \tag{4.73}$$

де $X := \{x \in R^n_+ : Ax = b\}$ характеризує звичайні детерміновані обмеження та обмеження на невід'ємність. Стохастичні обмеження $T_i(\omega)x = h_i(\omega)$, $i \in I := \{1,...,m\}$ являють собою різні цілі, які ми хотіли б задовольнити. Однак в той час, коли має бути прийняте рішення відносно x, фактичні значення матриці $T(\omega)$ та правих частин $h_i(\omega)$ ще невідомі, тому що вони залежать від випадкового вектора ω. Припустимо, що ця залежність лінійна. Також будемо вважати, що відома тільки ймовірнісна інформація відносно ω, тобто задано розподіл для ω.

Такі задачі дуже поширені на практиці. Наприклад, у випадку, коли модель (4.72)–(4.73) є задачею планування виробництва, $T(\omega)$ може описувати можливу технологію, що використовується, щоб задовольнити невідомий попит на вироби $h(\omega)$.

Відомим підходом є формулювання двохетапної моделі з *рекурсією*. Основна ідея полягає в тому, що рішення x приймається перед тим, як стають відомі реалізації , а потім, як тільки конкретна реалізація ω стає відомою, відхилення $h(\omega) - T(\omega)x$ слід скоригувати, враховуючи (мінімізуючи) витрати на цю корекцію. Витрати на корекцію (або *рекурсію*), що задаються *функцією рекурсії* v як $v(\omega - Tx)$, обчислюються для кожної можливої реалізації випадкового параметра ω. Зважені з відповідними імовірностями, вони становлять *очікувані витрати на рекурсію* $Q(x) := E_\omega[v(\omega - Tx)]$, які відповідають даному рішенню x. Таким чином, оптимальним критерієм для вибору $x \in$ мінімізація загальних очікуваних витрат, що складаються з прямих витрат cx і очікуваних витрат на рекурсію $Q(x)$. Цю модель можна зобразити у вигляді

$$\min_{x \in X}[(c,x) + Q(x)], \tag{4.74}$$

де $X := \{x \in R_+^n : Ax = b\}$. Функція рекурсії v звичайно визначається як розв'язок деякої задачі лінійного програмування, яку називають задачею *другого етапу*: для $s \in R^m$,

$$v(s) := \min_y qy,$$
$$Wy = s, \qquad (4.75)$$
$$y \in Y,$$

де W називається *матрицею рекурсії*, вектор q визначає вартість одиниці рекурсії, а Y описує допустиму множину рекурсій.

Така модель рекурсії є досить гнучкою: структуру рекурсії (q, W, Y) можна вибрати багатьма способами. Добре відомим спеціальним випадком моделі є *проста рекурсія*, задана матрицею рекурсії виду $W = (I_m, -I_m)$, де I_m – одинична матриця розмірності $m \times m$, вектором $q = (q^+, q^-)$ і множиною $Y = R_+^{2m}$, для яких v набуває вигляду

$$v(s) := \min_y [q^+y^+ + q^-y^-],$$
$$y^+ - y^- = s,\ s \in R^m,$$
$$y^+, y^- \in R_+^m.$$

Використовуючи *сепарабельність* функції v і припускаючи, що її параметри задовольняють нерівності $q^+ + q^- \geq 0$ (покомпонентно), одержуємо $v(s) = \sum_{i=1}^{m} v_i(s_i)$, де кожна функція v_i має вигляд

$$v_i(s_i) = q_i^+(s_i)^+ + q_i^-(s_i)^-,$$

де $(u)^+ := \max\{u, 0\}$; $(u)^- := \max\{-u, 0\}$, $u \in R$.

Інтерпретація поданої моделі рекурсії полягає в тому, що лінійні штрафи призначаються за дефіцит або надлишки щодо кожного обмеження $T_i(\omega)x = \omega_i$, $i = 1, ..., m$ індивідуально.

Сепарабельність функції v є ключем до побудови дуже ефективних алгоритмів, які дають можливість розв'язувати задачі великих розмірностей на основі моделей з простою рекурсією. Дійсно, для задач із сотнями випадкових змінних, що виникають, наприклад, у деяких технічних галузях, формулювання моделі у вигляді простої рекурсії – єдиний ефективний вибір з обчислювальної точки зору. Нижче пропонується ефективний алгоритм для стохастичних задач із простою рекурсією.

Розглянемо модель із простою рекурсією та з випадковими технологічною матрицею і вектором правих частин обмежень

$$\min_x [(c, x) + Q(x)],$$
$$x \in X, \qquad (4.76)$$

де $Q(x) := E_\omega[v(h(\omega) - T(\omega)x)]$ є функцією очікуваних витрат, а

$$v(u) := \min_y \left[q^+y^+ + q^-y^-\right],$$
$$y^+ - y^- = u,\ u \in R^m, \qquad (4.77)$$
$$y^+ \in R_+^m, y^- \in R_+^m,$$

є цільовою функцією для задачі другого етапу. Згідно з припущенням, що $q_i^+ + q_i^- \geq 0$, $i = 1, ..., m$ функція v є обмеженою й опуклою.

Позначимо відхилення $\eta_i(x, \omega) := T_i(\omega)x - h_i(\omega)$, $i \in I$. Для кожного окремого відхилення штрафи q_i^+ і q_i^- призначаються відповідно за надлишки $\eta_i(x, \omega)^+ := \max\{0, \eta_i(x, \omega)\}$ та дефіцит $\eta_i(x, \omega)^- := \max\{0, -\eta_i(x, \omega)\}$. Цільова функція набуває вигляду

$$(c, x) + E_\omega\left[\sum_{i\in I}(q_i^+\eta_i(x,\omega)^+ + q_i^-\eta_i(x,\omega)^-)\right]. \qquad (4.78)$$

Скористаємося сепарабельністю функції очікуваних витрат Q. Нехай ω_I – це ті компоненти вектора ω, від яких фактично залежать $T_i(\omega)$ і $h_i(\omega)$. Тоді $(T_i(\omega), h_i(\omega)) = (T_i(\omega_I), h_i(\omega_I))$, $i = 1, ..., m$, а функція Q є сепарабельною і може бути записана у вигляді

$$Q(x) = \sum_{i=1}^{m} Q_i(x),\ x \in R^n, \qquad (4.79)$$

де

$$Q_i(x) := q_i^+E_{\omega_I}[(h_i(\omega_I) - T_i(\omega_I)x)^+] + q_i^-E_{\omega_I}[(h_i(\omega_I) - T_i(\omega_I)x)^-],$$
$$i = 1, ..., m.$$

Тут математичне очікування береться відносно розподілу ω_i. Враховуючи, що $(s)^+ - (s)^- = s$, і використовуючи раніше введене позначення $\eta_i(x, \omega) := T_i(\omega)x - h_i(\omega)$, отримуємо такий вираз:

$$Q_i(x) = q_i^+ E_{\omega_i}[\eta_i(x,\omega_i)^-] + q_i^- E_{\omega_i}[\eta_i(x,\omega_i)^+] =$$
$$= q_i^-(\tau_i x - \mu_i) + (q_i^+ + q_i^-) E_{\omega_i}[(\eta_i(x,\omega_i)^-], \qquad (4.80)$$

де $(\tau_i, \mu_i) = E_{\omega_i}[(T_i(\omega_i), h_i(\omega_i))]$.

Припустимо, що ω є випадковим вектором з дискретним розподілом $\Pr\{\omega = \omega_s\} = p_s$, $s \in S$. Крім того, нехай $(T_s, h_s) = (T(\omega_s), h(\omega_s))$ для $s \in S$. Тоді для вектора ω маємо

$$E_{\omega_i}[\eta(x,\omega_i)^-] = \sum_{s \in S_i} \max\{0, -p^s \eta(x, \omega_i^s)\}$$

і

$$Q_i(x) = q_i^+ E_{\omega_i}[\eta_i(x,\omega_i)^-] + q_i^- E_{\omega_i}[\eta_i(x,\omega_i)^+] = \qquad (4.81)$$
$$= q_i^-(\tau_i x - \mu_i) + (q_i^+ + q_i^-) \sum_{s \in S_i} \max\{0, -p^s \eta(x, \omega_i^s)\}.$$

Отже, щоб розв'язати задачу (4.76)–(4.77), можна застосувати алгоритм субградієнтного типу. Пропонуємо в цьому випадку скористатися r-алгоритмами (див. розділ 1). Є кілька переваг у використанні цього підходу.

1. Виходячи з сепарабельності функції Q, ми повинні розглянути тільки $\sum_{i=1}^{m} S_i$ задач другого етапу (за умови, що рядки є незалежними), щоб обчислити значення Q і субградієнт для кожного поточного розв'язку x^t.
2. Замість знаходження оптимального значення цільової функції для кожної задачі другого етапу за допомогою розв'язання задачі лінійного програмування можемо використовувати аналітичні формули для $Q(x^t)$ і для субградієнта в точці x^t.

Для спеціального випадку моделі з детермінованою технологічною матрицею T, коли тільки вектор правих частин $h(\omega)$ є стохастичним (ω є дискретним випадковим вектором),

маємо відхилення $\eta_i(x, \omega) := T_i x - h_i(\omega)$, $i \in I$. Введемо позначення $\chi := Tx$. Тоді вираз (4.80) можна подати у вигляді

$$Q_i(\chi_i) = q_i^-(\chi_i - \mu_i) + (q_i^+ + q_i^-) E_{\omega_i}[(\eta_i(\chi_i, \omega_i)^-],$$

і

$$E_{\omega_i}[\eta(\chi_i, \omega_i)^-] = \sum_{s \in S_i} \max\{0, -p^s \eta(x, \omega_i^s)\} =$$
$$= \sum_{s \in S_i} \max\{0, -p^s(\chi_i - h_i(\omega_i^s))\}.$$

У цьому випадку, у зв'язку з сепарабельністю моделі, можна зовсім не розглядати задачі другого етапу і обчислювати субградієнт у фіксованій точці χ, використовуючи тільки розподіл компонентів ω_i випадкового вектора ω, за допомогою досить ефективної обчислювальної процедури.

В обох розглянутих випадках для врахування обмежень першого етапу $X := \{x \in R_+^n : Ax = b\}$ можна використовувати, наприклад, метод негладких штрафних функцій [153].

Нижче описується алгоритм для задач із фіксованою рекурсією і з дискретно розподіленими випадковими технологічною матрицею і вектором правих частин.

Розглянемо модель з рекурсією (4.74)–(4.75) у випадку, коли невизначеність подана за допомогою випадкових змінних, визначених у деякому дискретному ймовірністному просторі, множиною сценаріїв $\Omega = \{1, ..., L\}$ з відповідними імовірностями $\{p_1, ..., p_L\}$. Цю модель можна подати у вигляді такої еквівалентної детермінованої моделі:

$$\min[(c, x) + \sum_{l=1}^{L} p_l (q_l, y_l)],$$
$$x \in X, \qquad (4.82)$$
$$T_l x + W y_l = h_l, \quad l = 1, ..., L,$$
$$y_1 \geq 0$$

Очевидно, що задача (4.78) має блочно-діагональну структуру:

$$\min[(c,x) + p_1 q_1 y_1 + \cdots + p_L q_L y_L],$$
$$x \in X,$$
$$\begin{aligned} Wy_1 &= h_1 - T_1 x, \\ &\ddots \quad \vdots \\ Wy_L &= h_L - T_L x, \end{aligned}$$
$$y_l \geq 0, \quad l = 1,\ldots,L,$$

яка може бути використана при розробці ефективних алгоритмів.

Для розв'язання задачі (4.78) пропонується алгоритм, що використовує схему декомпозиції за змінними з поділом останніх на змінні "першого етапу" x та "змінні другого етапу" y.

Використовуючи таку схему декомпозиції, зафіксуємо значення змінних першого етапу $x = \bar{x}$. Задача (4.78) розпадається на L лінійних підзадач із змінними другого етапу $y(\bar{x})$:

$$\begin{aligned} &\min q_l y_l, \\ &Wy_l = h_l - T_l \bar{x}, \\ &y_l \geq 0. \end{aligned} \tag{4.83}$$

Задачу (4.79) замінюємо на таку:

$$\begin{aligned} &\min[q_l y_l + R^+ y_l^+ + R^- y_l^-], \\ &Wy_l + y_l^+ - y_l^- = h_l - T_l \bar{x}, \\ &y_l \geq 0, \\ &y_l^+ \geq 0, \\ &y_l^- \geq 0. \end{aligned} \tag{4.84}$$

Додаткові змінні y_l^+, y_l^- додані в модель (4.79) для забезпечення сумісності обмежень. Символи R^+, R^- (R^+, $R^- > 0$) позначають коефіцієнти штрафу. Припустимо, що обмеження першого етапу сумісні. Очевидно, якщо в оптимальному розв'язку змінні y_l^+, y_l^- модифікованої задачі (4.84) дорівнюють нулеві, то він є також розв'язком початкової задачі. Лінійні підзадачі в схемі декомпозиції можна розв'язувати спеціалізованими симплексними алгоритмами.

Нехай $\{y_l(\bar{x}), u_l(\bar{x})\}$ – пара оптимальних прямих та двоїстих змінних, яка формує точку Куна-Таккера для l-ї лінійної підзадачі. Тоді, виходячи з схеми декомпозиції за змінними, субградієнт у точці $\bar{x}$ можна обчислити, використовуючи оптимальні двоїсті змінні $\{u_l(\bar{x})\}$. Для врахування обмежень першого етапу пропонується застосовувати метод негладких штрафних функцій. Отже, для знаходження оптимального x можна використовувати методи субградієнтного типу. Для розв'язання координуючої задачі відносно змінних першого етапу x пропонується використовувати r-алгоритми.

4.4.3. Результати обчислювальних експериментів

Тут наводяться результати ряду обчислювальних експериментів з розв'язання мережевих двохетапних стохастичних задач за допомогою розробленого програмного забезпечення – модуля FIXED-T для розв'язання стохастичних задач із фіксованою рекурсією.

Для побудови тестової задачі для обчислювальних експериментів було вибрано за основу модель типу (4.71). Нижче подано формулювання тестової моделі.

Нехай n – кількість направлених ребер мережі, пропускну здатність яких можна збільшити, а $y \in R^n$ – вектор збільшених пропускних здатностей ребер. Нехай m – кількість пар вершин мережі, які слід обслужити, а $d \in R^m$ – випадкова змінна вимог на обслуговування між парами вершин. Вважаємо, що вимоги – це випадкові величини із скінченними дискретними розподілами, які встановлюються за допомогою списку реалізацій/ймовірностей. Нехай L – кількість реалізацій; p_l – ймовірність виникнення реалізації (сценарію) $l = 1, ..., L$; d_i^l – i-а компонента випадкового вектора для l-ї реалізації. Нехай для пар вершин з $i = 1, ..., m$ величина $K(i)$ – множина шляхів, які можна використати для виконання вимоги на обслуговування між двома пунктами. Крім того, для шляху $k \in K(i)$ нехай $a_k \in Z^n$ – вектор інцидентності, який визначається так:

$$a_{kj} = \begin{cases} 1, & \text{якщо } j \in k, \\ 0, & \text{в протилежному випадку.} \end{cases}$$

Нехай $y^0 \in R^n$ – існуючі пропускні здатності мережі, c_j – вартість одиничної додаткової пропускної здатності для ребра j, s_i^l – кількість вимог, що не були виконані для сценарію l, а x_{ik}^l – кількість з'єднань, що обслуговують пару i шляхом k для сценарію l.

Якщо задано поточний стан мережі $\{a_{ik}, y^0\}$, то задача полягає в мінімізації витрат на збільшення пропускних здатностей та очікуваних витрат на обслуговування мережі за умови штрафів за невиконання вимог q_i^*.

Ця модель є двохетапною стохастичною задачею з рекурсією та випадковими правими частинами обмежень. Змінні першого етапу – це y_j, а змінні рекурсії – це s_i^l та x_{ik}^l.

Можемо записати модель у такій еквівалентній детермінованій формі:

$$\sum_{j=1}^{n} c_j y_j + \sum_{l=1}^{L} p_l \left[\sum_{j=1}^{n} q_j \left(\sum_{i=1}^{m} \sum_{k \in K(i)} a_{ij} x_{ik} \right) + \sum_{i=1}^{m} q_i^* s_i^l \right] \to \min$$

при обмеженнях

$$\sum_{i=1}^{m} \sum_{k \in K(i)} a_{ij} x_{ik}^l \le y_j^0 + y_j, \quad j = 1,\ldots,n, \quad l = 1,\ldots,L,$$

$$\sum_{k \in K(i)} x_{ik}^l + s_i^l = d_i^l, \quad i = 1,\ldots,m, \quad l = 1,\ldots,L,$$

$$0 \le y_j \le u_j,$$

$$s_i^l \ge 0, \; x_{ik}^l \ge 0.$$

Для тестування було розглянуто приклад на основі мережі, зображеної на рис. 4.3. У цій мережі 8 ребер та 6 вузлів. Детерміновані параметри мережі подані в табл. 4.5.

Кількість незалежних випадкових величин дорівнювала 15. Емпіричні дискретні розподіли було одержано за допомогою дискретизації рівномірного розподілу з лівим значенням 0 та правим 10 для різної кількості реалізацій у різних тестових задачах. Максимальна кількість сумісних реалізацій дорівнювало $2^{10} = 1024$. Значення штрафів за

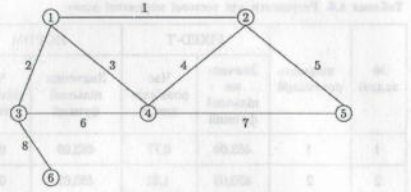


Рис. 4.3. Приклад тестової мережі

невиконані вимоги було вибрано $q_i^* = 100$ для всіх пар точок. Результати для тестової мережевої задачі наведено в табл. 4.6. Для порівняння було взято широко відому програму HOPDM [99] для розв'язання задач лінійного програмування методом внутрішніх точок.

Результати показали, що всі тестові мережеві задачі було розв'язано з достатньою точністю. При порівнянні часу розв'язування з програмою HOPDM, призначеною для розв'язання задач лінійного програмування, видно, що для задач з кількістю реалізацій 256 та вище (рис. 4.4) програма FIXED-T показала кращі результати: час розв'язування для програми HOPDM зростає при збільшенні кількості реалізацій значно швидше, ніж для програми FIXED-T. Таким чином, програму FIXED-T можна ефективно використовувати для розв'язування цього типу задач з кількістю реалізацій близько 1000.

Таблиця 4.5. Початкові пропускні здатності y_j^0 та параметри c_j і q_j для тестової мережі

Параметри	Ребра j							
	1	2	3	4	5	6	7	8
y_j^0	2	2	4	4	2	4	3	1
c_j	1	1	1	1	1	1	1	1
q_j	1.5	1	1	1	1	1	1	1

Таблиця 4.6. Результати для тестової мережевої задачі

№ задачі	кількість реалізацій	FIXED-T		HOPDM	
		Значення цільової функції	Час розв'язання, с	Значення цільової функції	Час розв'язання, с
1	1	463,00	0,77	463,00	0,17
2	2	450,03	1,32	450,02	0,06
3	4	440,55	2,36	440,55	0,11
4	8	431,07	4,62	431,07	0,16
5	16	408,09	8,67	408,09	0,39
6	32	388,43	14,88	388,44	1,32
7	64	362,09	29,77	362,096	8,19
8	128	349,23	78,38	349,23	19,28
9	256	338,46	114,02	338,46	181,59
10	512	303,07	230,41	303,07	332,8
11	1024	290,82	381,24	290,82	2410,4

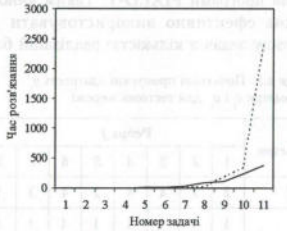


Рис. 4.4. Залежність часу розв'язання для тестових задач:
"__" – FIXED-T; "....." – HOPDM

Розділ 5. Перспективне планування перевезень (залізничних, автомобільних), знаходження оптимальної номенклатури рухомого складу

Математична модель, наведена в даному розділі, призначена для розв'язання задач перспективного планування перевезень та раціонального розподілу коштів на реконструкцію транспортних мереж (залізничних, автомобільних, авіаційних). Реалізація моделі дає можливість одержати чисельну інформацію з питань перспективного планування, що стосуються: критичних за пропускними здатностями вузлів та раціонального розподілу коштів на їх реконструкцію; раціональної схеми потоків транспортних засобів із врахуванням мінімізації експлуатаційних затрат і обсягів порожніх потоків; знаходження оптимального складу парка транспортних засобів та раціонального використання коштів для його розширення. Наведено характеристику відповідних задач оптимізації й принципи побудови методів їх розв'язання. Для визначеності термінології модель побудовано для задачі перспективного планування вантажних залізничних перевезень та модернізації залізничної транспортної системи, проте її можна використовувати і для інших областей застосування.

5.1. Постановка задачі

Математична модель призначена для розв'язання задач перспективного планування вантажних залізничних перевезень і модернізації залізничної транспортної системи. Головна мета полягає в знаходженні раціональної схеми вагонопотоків

та розподілу капіталовкладень на реконструкцію станцій і поповнення рухомого складу.

На змістовному рівні розглянута задача перспективного планування описується так. Задано плановий період, що складається з окремих інтервалів. Кожен інтервал планового періоду характеризується відносно стабільним потоком вантажних перевезень на залізничній мережі. Як правило, вибір інтервалів визначається фактором сезонності. Тривалості інтервалів вважаються досить великими, тобто більшими, ніж тривалість виробничих циклів транспортної системи (наприклад, середнього часу обігу вагонів).

Для планового періоду визначені головні параметри здатності транспортної системи: парк транспортних засобів та пропускні здатності станцій з вантажних операцій. План реконструкції транспортної системи полягає у визначенні доцільного розподілу заданого обсягу фінансових вкладень, спрямованих на збільшення пропускних здатностей станцій щодо здійснення вантажних операцій та поповнення рухомого складу.

Для кожного інтервалу планового періоду відомі прогнозовані обсяги агрегованих вантажних перевезень (кореспонденції). Для кореспонденції визначені: вид агрегованого вантажу (зерно, мінеральні добрива, нафта тощо), станції відправлення та призначення, пріоритет реалізації та прогнозований максимальний обсяг перевезення.

Одним із головних понять, що використовуються з метою побудови моделі, є вагонопотік на дільниці мережі. Вагонопотік визначається для кожного інтервалу і кожного типу вагона. Інтенсивність вагонопотоку визначається загальною кількістю вагонів, які проходять через дільницю мережі в інтервалі планового періоду. Для кожного інтервалу виконуються умови балансу вагонопотоків: загальна кількість вагонів даного типу, що прибувають на станцію, дорівнює кількості таких вагонів, що відбувають із станції. Вагонопотік даного типу відповідає звичайному поняттю циркуляції в орієнтованих мережах [13, 35, 36].

Критерій оптимізації полягає в максимізації загального прибутку, одержаного в плановому періоді, за рахунок вантажних перевезень. В результаті реалізації математичної моделі можна одержати чисельну інформацію з таких головних питань перспективного планування:

- виявлення критичних за пропускною потужністю залізничних станцій та рекомендації з розподілу коштів на їх реконструкцію;
- визначення раціонального складу парку транспортних засобів та рекомендації по розподілу коштів для його поповнення;
- розробка раціональної схеми вагонопотоків із врахуванням мінімізації експлуатаційних витрат та обсягів безвантажних потоків;
- раціональний вибір замовлень на перевезення до їх реалізацій.

Ці дані можна використовувати для прийняття рішень перспективного розвитку транспортної системи та об'єктивного обгрунтування критичних факторів її функціонування. Схема вагонопотоків забезпечує важливу інформацію для розв'язання задачі визначення маршрутизації вантажних потягів та складання розкладу їх руху.

5.2. Математична модель

Модель описується на концептуальному рівні. Вона характеризується досить великим обсягом різнорідних даних. Вхідні дані моделі будуть мати позначку data, вихідні – result.

Далі наведено об'єкти, позначення та основні співвідношення моделі.

Плановий період. T (data) – множина всіх інтервалів планового періоду; t – символ ідентифікації інтервалів: $t \in T$;

τ_t (data) – тривалість інтервалу t планового періоду.

Транспортна мережа подається за допомогою орієнтованого графа *G*. Вершини графа відповідають станціям (вузлам)

залізниці. Орієнтовані ребра графа відповідають окремим дільницям (перегонам) залізниці. Введемо позначення:

I (data) – множина станцій; i – символ ідентифікації станцій залізниці, $i \in I$;

J (data) – множина дільниць; j – символ ідентифікації дільниць залізниці, $j \in J$;

$\Gamma(i)$ ($\Gamma^{-}(i)$) (data) – множина дільниць, які приходять на станцію (які виходять із станції) i;

$i^{-}(j)$ ($i^{+}(j)$) (data) – початкова (кінцева) станція дільниці j.

Транспортні засоби, транспортний потік. Під транспортними засобами розуміємо вагони вантажних перевезень (наявні на залізниці в плановому періоді). Позначимо: K (data) – множина типів вагонів (товарні вагони, насипні платформи, цистерни тощо); k – символ ідентифікації типів вагонів, $k \in K$

Одним із головних понять моделі є транспортний потік (вагонопотік) на дільниці мережі. Інтенсивність (величина) транспортного потоку $y_k(j,t)$ (result) визначається загальним числом вагонів типу k, які проходять по дільниці j в інтервалі t. При цьому ігноруються цільові призначення вагонів (до числа вагонів, які визначають величину транспортного потоку, включаються як вагони з вантажем, так і порожняк). Множина всіх змінних $y_k(j,t)$ буде позначатися Y.

Передбачається, що тривалість інтервалів планового періоду τ_t істотно більша за тривалість виробничого циклу залізниці. Тому для кожного інтервалу повинні виконуватися умови балансу транспортних потоків: загальна кількість вагонів даного типу, що прибувають на станцію, дорівнює кількості таких вагонів, що відбувають із станції:

$$\sum_{j \in \Gamma^{-}(i)} y_k(j,t) = \sum_{j \in \Gamma(i)} y_k(j,t),\ i \in I, k \in K, t \in T. \quad (5.1)$$

Відзначимо, що вагонопотік $y_k(j,t)$ відповідає звичайному поняттю циркуляції в орієнтованих мережах [13, 35, 36]. З транспортним потоком на дільниці пов'язані експлуатаційні витрати на його реалізацію: $c_j^{-}(k,t)$ (data) – експлуатаційні витрати на перегін одного вагона типу k на дільниці j в інтервалі t. Сумарні експлуатаційні витрати $F^{-}(Y)$(result) по реалізації транспортних потоків у плановому періоді визначаються формулою

$$F^{-}(Y) = \sum_{t \in T} \sum_{k \in K} \sum_{j \in J} c_j^{-}(k,t) y_k(j,t). \quad (5.2)$$

Кореспонденції перевезень. Для розв'язання розглянутого класу задач необхідна інформація щодо замовлень (попиту) на вантажні перевезення для інтервалів планового періоду. Вона, як правило, генерується на основі обробки статистичних даних щодо перевезень за минулий час та прогнозу замовлень на плановий період. Позначимо: L (data) – множина різних типів агрегованих вантажних перевезень (вантажів); l – символ ідентифікації типів вантажів. Для кожного типу вантажу задана множина типів вагонів K_l (data), якими цей вантаж можна перевозити, $K_l \subseteq K$.

Для скорочення замовлення на перевезення будемо називати кореспонденцією. Нехай Q (data) – множина кореспонденцій планового періоду, q – символ ідентифікації кореспонденцій. Для кореспонденції $q \in Q$ визначені такі дані: інтервал планового періоду t_q (data); тип вантажу l_q (data); максимальний прогнозований обсяг перевезень B_q (data). Обсяг перевезень кореспонденції задається в одиницях виміру вантажу типу l_q. Для визначення кількості вагонів типу k, необхідної для перевезення вантажу типу l, використовуються коефіцієнти γ_{kl}(data), що дорівнюють місткості вагона типу k в одиницях виміру вантажу типу l. Кореспонденція може мати кілька станцій призначення (відправлення), наприклад, така, що, пов'язана з доставкою вантажів у порти (з портів). Вантаж такої кореспонденції можна доставляти на припортові станції за визначеним списком.

В моделі буде використовуватися варіантна схема реалізації кореспонденцій, яка виходить з того, що для кожної кореспонденції q апріорі задана певна множина способів її реалізації (множина реалізацій) Ω_q(data). Для кожної реалізації $\omega \in \Omega_q$ визначено такі дані:

- маршрут $\Re_q(\omega)$ (data);
- тип вагона $k_q(\omega)$ (data);
- тариф вартості перевезення одиниці вантажу $c_q^+(\omega)$ (data);
- тривалість реалізації перевезення $\theta_q(\omega)$ (data).

Маршрут $\Re_q(\omega)$ подається послідовним списком дільниць мережі й відповідає звичайному визначенню маршрута на орієнтованому графі:

$$\Re_q(\omega) = \{j^1, j^2, ..., j^{n_q}\},$$

$$i^-(j^{k+1}) = i^+(j^k), \quad k = 1,2,...,n_q - 1,$$

де n_q – кількість дільниць маршрута $\Re_q(\omega)$.

Початкова станція $i_q^-(\omega) = i^-(j^1)$ маршрута $\Re_q(\omega)$ є однією із станцій відправлення кореспонденції, а кінцева станція $i_q^+(\omega) = i^+(j^{n_q})$ маршрута $\Re_q(\omega)$ – однією із станцій призначення кореспонденції.

До тривалості реалізації перевезення $\theta_q(\omega)$ зараховуються: час підготовки вагона до відправлення (навантаження вагона, формування поїзда), час доставки вагона з початкової станції на кінцеву станцію маршруту та час розвантаження вагона. Запропонована форма даних варіантів реалізацій кореспонденцій є інформаційно надлишковою: реалізації можуть відрізнятися тільки типом вагонів при однакових маршрутах; значення тарифів вартості перевезення одиниці вантажу $c_q^+(\omega)$ можуть визначатися тільки станціями відправлення та призначення кореспонденції й тому бути однаковими для різних варіантів її реалізації тощо. Однак така форма опису варіантів реалізацій кореспонденцій забезпечує компактність запису й простоту інтерпретації моделі. Крім того, вона досить загальна і враховує різні варіанти формулювання задачі (наприклад, за умови залежності тарифу вартості перевезення від типу вагонів).

Формально множина варіантів реалізації кореспонденції розглядається в моделі як підмножина вхідних даних. Як правило, ці дані генеруються програмно. При цьому як маршрути

кореспонденцій можуть розглядатися, наприклад, найкоротші за вибраними критеріями (відстанню, експлуатаційними витратами; часом реалізації) шляхи на мережі залізничних шляхів, що з'єднують станції відправлення та призначення кореспонденції. При генеруванні маршрутів кореспонденцій можуть враховуватися додаткові технологічні (наприклад, екологічні) вимоги на допустимі маршрути перевезень даної кореспонденції, які в явному вигляді в моделі не відображені.

Кожній реалізації ω кореспонденції q відповідає змінна моделі $x_q(\omega)$ (result), що характеризує обсяг перевезень кореспонденції q за варіантом її реалізації ω. Множину всіх змінних $x_q(\omega)$ позначимо X^q. Загальний обсяг реалізованих перевезень кореспонденції має бути не більшим, ніж прогнозований максимальний обсяг кореспонденції B_q.

Позначимо: x_q^- (result) – нереалізований обсяг кореспонденції q; c_q^- (data) – штраф за наявність одиниці нереалізованого обсягу кореспонденції q.

Зауважимо, що штрафні множники c_q^- відображають, як правило, не реальні фінансові збитки в результаті неповної реалізації кореспонденції, а є керуючими параметрами моделі, які дають можливість ранжувати кореспонденції за ступенем важливості реалізації їх перевезення. Крім того, введення змінних x_q^- забезпечує формальну сумісність обмежень моделі. Ненульове значення змінної вказує на нерентабельність реалізації кореспонденції чи на обмеженість потужності транспортної системи.

Змінні $x_q(\omega)$ та x_q^- задовольняють балансовим співвідношенням:

$$\sum_{\omega \in \Omega_q} x_q(\omega) + x_q^- = B_q, \quad x_q(\omega) \geq 0, \quad x_q^- \geq 0, \quad q \in Q. \quad (5.3)$$

Загальний "прибуток" $F^+(X^q)$ від реалізацій перевезень всіх кореспонденцій визначається за формулою

$$F^+(X^q) = \sum_{q \in Q} \sum_{\omega \in \Omega_q} c_q^+(\omega) x_q(\omega). \quad (5.4)$$

Загальні "втрати" $F^-(X^q)$ у зв'язку з наявністю нереалізованих перевезень всіх кореспонденцій визначаються рівністю

$$F^-(X^q) = \sum_{q \in Q} c_q^- x_q^- . \qquad (5.5)$$

Нехай $w_k^q(j,t)$ (result) – кількість вагонів типу k, які проходять через дільницю j в інтервалі t, при реалізації кореспонденції q. Значення $w_k^q(j,t)$ ($t \in T$; $j \in J$; $k \in K$; $q \in Q$) однозначно визначаються значеннями змінних $x_q(\omega)$:

$$w_k^q(j,t) = \sum_{\omega \in \Omega} \{x_q(\omega)/\gamma_{u_q} \mid \ t_q = t,\ k_q(\omega) = k,\ j \in \Re_q(\omega)\} . \qquad (5.6)$$

Кількість $w_k(j,t)$ (result) вагонів типу k, що проходять через дільницю j в інтервалі t при реалізації всіх кореспонденцій, визначається за формулою

$$w_k(j,t) = \sum_{q \in Q} w_k^q(j,t),\ t \in T;\ j \in J;\ k \in K . \qquad (5.7)$$

Умова узгодження величин вагонопотоків $y_k(j,t)$, які треба визначити, з реалізацією перевезень кореспонденцій визначається обмеженнями

$$y_k(j,t) \geq w_k(j,t),\ \ t \in T,\ \ k \in K,\ \ j \in J. \qquad (5.8)$$

Значення змінних $y_k(j,t)$, $w_k(j,t)$ однозначно визначають величини $w_k^0(j,t)$, що позначають число вагонів типу k, які проводяться через дільницю j в інтервалі t без вантажів ("порожняком"):

$$w_k^0(j,t) = y_k(j,t) - w_k(j,t),\ \ t \in T, k \in K, j \in J . \qquad (5.9)$$

Пропускна здатність вузла з обробки вантажів щодо обробки вантажів буде враховуватися в моделі в агрегованому вигляді шляхом визначення для вузла транспортної мережі потужності з обробки вантажів. (Для скорочення в моделі розглядатиметься лише один тип робіт з обробки вантажів. У програмному забезпеченні враховується наявність різних типів таких робіт: обробка нафтопродуктів, сипучих вантажів тощо). Позначимо β_i (data) потужність станції i з обробки вантажів.

Тоді обсяг робіт з обробки вантажів, який можна виконати на станції i в інтервалі t, дорівнює $\beta_i \tau_t$, де τ_t – тривалість інтервалу t.

Введемо такі допоміжні змінні моделі:

$z_l^+(i,t)$ (result) – обсяг вантажу типу l, який приймається на станцію i в інтервалі t;

$z_l^-(i,t)$ (result) – обсяг вантажу типу l, який відправляється із станції i в інтервалі t.

Значення змінних $x_q(\omega)$ однозначно визначають обсяги й тип вантажів, які приймаються (відправляються) на станціях:

$$z_l^+(i,t) = \sum_{q \in Q} \sum_{\omega \in \Omega} \{x_q(\omega) \mid \ l_q = l,\ t_q = t,\ i_q^+(\omega) = i\}, \qquad (5.10)$$

$$z_l^-(i,t) = \sum_{q \in Q} \sum_{\omega \in \Omega} \{x_q(\omega) \mid \ l_q = l,\ t_q = t,\ i_q^-(\omega) = i\}. \qquad (5.11)$$

Умови обмеженості пропускної потужності вузлів транспортної мережі з врахуванням їх реконструкції визначаються такими нерівностями:

$$\sum_{l \in L} \{\kappa_l^+ z_l^+(i,t) + \kappa_l^- z_l^-(i,t)\} \leq (\beta_i + x_i^\beta)\tau_t,\ \ i \in I,\ t \in T, \qquad (5.12)$$

де $\kappa_l^+(\kappa_l^-)$ (data) – коефіцієнт трудомісткості по розвантаженню (навантаженню) вантажу типу l;

x_i^β (result) – додаткова потужність станції i щодо обробки вантажів у результаті її реконструкції;

X_β – множина всіх змінних x_i^β.

Коефіцієнти трудомісткості κ_l^+, κ_l^- вимірюються середнім часом, необхідним на обробку вантажу відповідного типу.

Загальні витрати на реконструкцію вузлів транспортної мережі $F^-(X_\beta)$ (result) визначаються виразом

$$F^-(X_\beta) = \sum_{i \in I} c_i^\beta x_i^\beta , \qquad (5.13)$$

де c_i^β (data) – витрати по збільшенню потужності станції i щодо обробки вантажів на одну одиницю.

Відзначимо, що з умови обмеженості пропускної потуж-

ності вузла (5.12) випливає, що в моделі передбачається проведення його реконструкції до початку планового періоду.

Потужність транспортних засобів визначається за допомогою величини P_k (data) – кількості вагонів типу k, що є в наявності на залізниці.

Умови обмеженості потужності транспортних засобів будуть визначатися в термінах "вагоно-годин" за допомогою величини $P_k\tau_t$ – кількості вагоно-годин типу k в інтервалі t, яка є в наявності на залізниці (τ_t – тривалість інтервалу t планового періоду).

Позначимо $v_k(t)$, $(t \in T;\ k \in K)$ (result) кількість вагоно-годин типу k, що треба витратити на реалізацію кореспонденцій в інтервалі t. З означення кореспонденції випливає, що значення $v_k(t)$, $(t \in T;\ k \in K)$ однозначно визначаються змінними обсягів реалізацій кореспонденцій $x_q(\omega)$:

$$v_k(t) = \sum_{q \in Q} \sum_{\omega \in \Omega} |(x_q(\omega) / \gamma_{k_q}) \theta_q(\omega)| \quad t_q = t,\ k_q(\omega) = k\}. \quad (5.14)$$

Нехай $v_k^0(t)$ (result) – кількість вагоно-годин типу k, які витрачаються на транспортування вагонів типу k без вантажів в інтервалі t:

$$v_k^0(t) = \sum_{j \in J} \tau_j w_k^0(j,t),\ (t \in T;\ \ k \in K), \quad (5.15)$$

де τ_j (data) – середній час подолання перегону на ділянці j.

В моделі передбачено можливість поповнення рухомого складу. У зв'язку з цим введено такі позначення:

c_k^P (data) – витрати на придбання одного вагона типу k;

x_k^P (result) – кількість придбаних вагонів типу k;

X_p – множина змінних x_k^P;

$F^P(X_p)$ (result) – загальні витрати на поповнення рухомого складу:

$$F^-(X_p) = \sum_{k \in K} c_k^P x_k^P. \quad (5.16)$$

Умови обмеженості потужності транспортних засобів із рахуванням поповнення рухомого складу будуть визначатися такими обмеженнями моделі:

$$v_k(t) + v_k^0 \le (P_k + x_k^P)\tau_t, \quad k \in K,\ t \in T. \quad (5.17)$$

Фінансові ресурси. Цільова функція. Позначимо B^f (data) обсяг фінансових ресурсів, які заплановані для використання з метою реконструкції вузлів транспортної мережі та поповнення рухомого складу. Обмеження на обсяг фінансових ресурсів, що використовуються на вказані цілі, подамо нерівністю

$$F^-(X_\beta) + F^-(X_p) \le B^f. \quad (5.18)$$

Цільова функція $F(X^q, Y)$ задачі визначається загальним прибутком у плановому періоді, а саме:

$$F(X^q, Y) = F^+(X^q) - F^-(X^q) - F^-(Y). \quad (5.19)$$

Критерій оптимізації задачі полягає в максимізації функції (5.19).

5.3. Характеристика моделі

Наведемо стислу характеристику головних співвідношень моделі.

Критерій оптимізації полягає в максимізації цільової функції (5.19), значення якої дорівнює сумі доходу, одержаного в результаті реалізації перевезень кореспонденції, за винятком штрафу за наявність нереалізованих обсягів кореспонденцій та експлуатаційних витрат, пов'язаних із реалізацією транспортних потоків.

Головні обмеження задачі:

- умови балансу вагонопотоків (5.1);
- балансові співвідношення обсягів реалізації кореспонденції (5.3);
- умови узгодження вагонопотоків із реалізацією перевезень кореспонденцій (5.8);
- умови обмеженості потужностей вузлів (5.12);
- умови обмеженості потужності транспортних засобів (5.17);

- умови обмеженості обсягу фінансових ресурсів, які використовуються на реконструкцію вузлів мережі та поповнення рухомого складу (5.18).

Головними змінними задачі є:

- змінні реалізацій варіантів перевезень кореспонденцій $x_q(\omega)$;
- змінні вагонопотоків $y_k(j,t)$;
- змінні реконструкції вузлів мережі x_i^β та поповнення рухомого складу x_k^β.

Інші змінні й обмеження моделі є допоміжними і введені нами для компактності записів та простоти інтерпретації моделі. Задача (5.1)–(5.19) є задачею лінійного програмування. Головною особливістю реальних задач розглянутого класу є їх велика вимірність. Наведемо типові значення параметрів моделі, що визначають її вимірність, для залізничних перевезень України:

$|T| \approx 4$ – кількість інтервалів планового періоду;

$|I| \approx 1000$ – кількість станцій;

$|J| \approx 1500$ – кількість дільниць залізниці;

$|K| \approx 20$ – кількість типів вагонів;

$|L| \approx 20$ – кількість типів агрегованих вантажних перевезень;

$|Q| \approx 4000$ – кількість кореспонденцій;

$|\Omega_q| \approx 3$ – середня кількість різних варіантів реалізації кореспонденції.

Тоді вимірність задачі (не враховуючи кількості допоміжних змінних, допоміжних обмежень і обмежень, що забезпечують несуперечність змінних) визначається співвідношеннями:

кількість змінних $\approx |T||K||J|+|\Omega_q||Q| \approx 132000$;

кількість обмежень $\approx |T||K||J|+|Q||J||I| \approx 200000$;

Таким чином, реальні задачі розглянутого класу можуть бути задачами істотно великої вимірності. В даний час існують потужні програмні засоби розв'язання задач лінійного програмування. Проте використання цього сучасного програмного забезпечення без врахування специфічних особливостей задач

недоцільно. Це пояснюється такими причинами. По-перше, задача, що досліджується, є задачею перспективного планування, багато її вихідних даних визначаються приблизно, на основі статистичного аналізу й прогнозу. Тому немає необхідності одержання її точного (в математичному розумінні) розв'язку. Головною метою є не одержання оптимальних варіантів реалізацій прогнозованих перевезень, а рекомендації з розвитку транспортної системи (щодо збільшення потужностей станцій і транспортних засобів). По-друге, задача характеризується яскраво вираженою блоковою структурою її змінних і обмежень. Більшість обмежень задачі можна віднести до так званих мережних обмежень. Зазначені особливості задачі використовуються в запропонованому далі підході до її розв'язання.

5.4. Метод розв'язання

Пропонуються два методи розв'язування задачі. Перший метод (Method_E_1) призначений для розв'язування задачі, яка повністю відповідає описаній вище моделі. Method_E_1 забезпечує формально точний розв'язок задачі. Його доцільно використовувати при наявності потужної обчислювальної системи чи при розв'язанні задач порівняно невеликої вимірності. Другий метод (Method_E_2) призначений для наближеного розв'язання задачі та заснований на суттєвій модифікації моделі шляхом введення так званих коефіцієнтів обігу транспортних засобів.

Method_E_1 заснований на використанні схеми декомпозиції за множиною змінних [45, 153]. Змінними декомпозиції є змінні реконструкції транспортної системи: x_i^β, x_t^β. Вони є зв'язуючими змінними задачі: розв'язання задачі при фіксованих значеннях x_i^β, x_t^β зводиться до розв'язання ряду незалежних задач для окремих інтервалів планованого періоду. Обчислення субградієнтів цільової функції за такою схемою декомпозиції зводиться до розв'язання двоїстої задачі при обмеженнях (5.12), (5.17).

Для розв'язання задачі верхнього рівня схеми декомпозиції пропонується застосувати r-алгоритм [45, 153]. Кількість змінних n цієї задачі дорівнює кількості зв'язуючих змінних обмежень: $n = |I'| + |K|$. Тут через I' позначено множину станцій, для якої задано обмеження (5.12). (Для стислості опису моделі ці обмеження визначалися для всіх станцій. Однак при розв'язуванні реальних задач змінні реконструкції станцій визначаються лише для деякої підмножини станцій із критичними потужностями щодо обробки вантажів. Як правило, кількість таких станцій, що підлягають реконструкції, становить не більше 10% від їх загальної кількості: $|I'| \approx 0{,}1 \cdot |I|$. Таким чином, число змінних двоїстої задачі $n \approx 120$. Як показує досвід, r-алгоритм забезпечує розв'язання задач у схемах декомпозиції лінійного програмування за ~$5n$ ітерацій. При цьому основний час виконання однієї ітерації r-алгоритму визначається часом розв'язання внутрішніх підзадач схеми декомпозиції. Зауважимо, що при розв'язанні внутрішніх підзадач (наприклад, симплекс алгоритмом) використовується результат їх розв'язання на попередній ітерації r-алгоритму. Крім того, немає необхідності в точному розв'язанні цих підзадач. Точність розв'язку може адаптивно збільшуватися в процесі роботи r-алгоритму. Це приводить до істотного зменшення трудомісткості підзадач схеми декомпозиції.

Method_E_2 призначений для наближеного розв'язання задачі і заснований на суттєвій модифікації моделі шляхом введення так званих коефіцієнтів обігу транспортних засобів. Коефіцієнт обігу $\eta_k(t)$ визначається для кожних типу вагона k та інтервалу планового періоду t із співвідношення

$$v_k^0(t) = \eta_k(t) v_k(t).$$

Таким чином, коефіцієнт обігу дорівнює відношенню вагоно-годин для завантажених вагонів до вагоно-годин для вагонів без вантажу (див. (5.14), (5.15)).

Якщо розв'язок задачі знайдений, то коефіцієнти обігу визначаються тривіально. З іншого боку, якщо апріорі визначити значення цих коефіцієнтів $\tilde{\eta}_k(t)$, то розв'язання задачі

істотно спрощується. У цьому випадку з моделі можна виключити змінні вагонопотоків $y_k(j,t)$, а також рівняння, що узгоджують вагонопотоки з реалізацією перевезень кореспонденцій (5.8). При цьому вимірність задачі істотно зменшується. Після розв'язання отриманої таким способом задачі, пошук значень вагонопотоків $y_k(j,t)$ зводиться до розв'язання задач по визначенню потоків вагонів без вантажу (незалежно від типів вагонів та інтервалів планового періоду). А кожна з них зводиться до класичної (однопродуктової) транспортної задачі.

Використовуючи таким способом знайдений розв'язок, визначаємо $\tilde{\tilde{\eta}}_k(t)$ – значення коефіцієнта обороту, що відповідає цьому розв'язку. Якщо значення $\tilde{\tilde{\eta}}_k(t) \approx \tilde{\eta}_k(t)$ то процес розв'язування задачі закінчений. У протилежному випадку змінюємо значення коефіцієнтів $\tilde{\eta}_k(t)$ (наприклад, $\tilde{\eta}_k(t) := (\tilde{\eta}_k(t) + \tilde{\tilde{\eta}}_k(t))/2$) і повторюємо процедуру розв'язання задачі.

Евристичне обгрунтування такого підходу до розв'язання задачі полягає в тому, що коефіцієнти обороту вагонів досить стабільні і визначаються загальним характером структури перевезень і транспортної мережі.

Розділ 6. Задачі проектування мереж для випадку, коли ребра, ізоморфні заданому графу, можуть вийти з ладу

Задачі проектування зв'язних мереж природним чином виникають при аналізі питань зв'язності мереж, які потрібно спроектувати таким чином, щоб при цьому можна було забезпечити не тільки передачу заданих обсягів при нормальній роботі мережі, але й виконати це при можливому виході з ладу окремих її елементів (ребер чи вузлів).

Зауважимо, що оптимізаційні задачі, які виникають при вивченні питань, пов'язаних зі збереженням зв'язності мережі після видалення з неї окремих компонентів, є надзвичайно складними. Подібні питання розглядалися, наприклад, у праці [138], де було розглянуто задачу проектування мережі мінімальної вартості за умови, що між деякими вузлами мережі мають існувати реберно-непересічні шляхи. Для розв'язання такої задачі спочатку будується допустима мережа, а потім за допомогою локального пошуку зменшується загальна вартість спроектованої мережі. В статті [98] розглянуто задачу побудови "живучої" мережі мінімальної вартості, в якій реберні зв'язності між окремими парами вершин дорівнюють максимальному з параметрів, що задані для кожної з вершин цієї пари. При цьому функції вартості ребер лінійно залежать від величини потоку на них. В [98] запропоновано евристичні алгоритми побудови деревовидної мережі та потрібні процедури локального пошуку. Зауважимо, що коли у всіх вершин ці параметри однакові і дорівнюють k, задача називається синтезом k-зв'язних мереж.

Ефективним підходом до задач синтезу надійних мереж виявилося застосування так званих розрізних нерівностей (cut inequalities), бо саме вони в багатьох випадках задають окремі грані многогранника допустимих розв'язків (див., наприклад, [74, 130, 120, 129, 58, 86, 85, 103, 102]). Незважаючи на те що початкова задача є *NP*-повною, задача перевірки розрізних нерівностей при певних умовах розв'язується за поліноміальний час за допомогою алгоритма Гоморі та Ху [36], а цим можна скористатися для відсічення деяких нецілочисельних допустимих розв'язків. Такий підхід було застосовано в [103] до розв'язання задачі синтезу за умови, що побудована мережа повинна залишатися зв'язною після видалення з неї заданої кількості будь-яких ребер. Цим методом в працях [85, 86] знайдено розв'язання задачі синтезу 2-зв'язної мережі з обмеженою довжиною її циклів. У процесі розв'язування згаданих задач, якщо після заданої кількості ітерацій покращення значення цільової функції не відбувається, робота алгоритмів завершується, тому оцінити точність знайденого розв'язку неможливо.

В даному розділі буде розглянуто задачу проектування надійної та зв'язної мережі (ЗПНЗМ), окремими випадками якої є добре відомі задача комівояжера (ЗК), задача Штейнера на графі (ЗШГ), задача побудови надійної мережі (ЗПНМ), задача синтезу мережі з одним джерелом (ЗСМ). Передбачається, що читач знайомий із термінологією теорії графів, наприклад, згідно з [13].

Щоб сформулювати ЗПНЗМ, розглянемо неорієнтований граф $G = (V, E)$, де V – множина його вершин, а E – множина ребер. Для кожної пари вершин s, t з множини V шляхом між цими (зв'язаними) вершинами називається послідовність вершин та ребер $s = v_0, e_1, v_1, ..., v_{l-1}, e_l, v_l = t$ де кожне ребро e_i з'єднує вершини v_{i-1}, v_i, $i = 1, ..., l$ і жодні вершина чи ребро не зустрічаються в цій послідовності більше, ніж один раз. Якщо вершини s і t співпадають, тобто $s = t$, такий шлях називається циклом на графі G. Коли говорять, що граф $G' = (V', E')$ є підграфом графа G, це означає, що $V' \subset V$, а множина E' скла-

дається з ребер, що належать E, причому їх кінцеві вершини знаходяться в V'.

Припустимо, що задано множину термінальних вузлів N ($N \subset V$) та невід'ємні вартості c_e для кожного з ребер e множини E. Задано також неорієнтований граф $H = (V_h, E_h)$, де V_h та E_h позначено відповідно множини вершин та ребер графа H. Іноді термінальні вузли з N будемо називати просто вузлами. Вартість будь-якого підграфа $G' = (V', E')$ графа G дорівнює сумі вартостей ребер у множині E'.

ЗПНЗМ полягає в знаходженні такого підграфа G_* графа G, в якому існує принаймні один шлях, що зв'язує довільну пару вузлів з N навіть при видаленні ребер будь-якого підграфа G, ізоморфного графу H, а вартість підграфа G_*, обчислена як сума вартостей ребер, є мінімальною.

В підрозділі 6.1 формулюється модель задачі ЗПМЗ у термінах потоку на мережі, а в підрозділі 6.2 (зокрема, в п. 6.2.1–6.2.4) ЗПНЗМ вивчається для випадку, коли граф H складається з одного ребра, тобто він має лише одне ребро та дві вершини. Цей випадок розглядається детально, бо вихід з ладу одночасно кількох елементів менш ймовірний, і тому така задача має великий практичний інтерес. Коли граф H складається з одного ребра, то ЗПНЗМ еквівалентна задачі Штейнера для знаходження підграфа, реберна зв'язність якого дорівнює 2 (2-зв'язність). В роботах [136, 138, 98, 102, 86, 130, 19, 39] з проектування телекомунікаційних мереж автори зосереджуються на k-зв'язності ($k \geq 0$) в задачах проектування мережі з мінімальним наперед заданим числом реберно-непересічних шляхів для будь-яких пар вершин. Многогранники, пов'язані з цими задачами, вивчалися в [138, 56, 102] та [87]. Зауважимо, що додаткову вимогу зберегти зв'язність графа G_* після видалення будь-якої вершини з V можна звести до задачі з видаленням ребра (причому вона матиме меншу вимірність). Тому ми не будемо тут розглядати умови існування шляху між будь-якими парами вузлів з N після видалення будя-якої вершини з V.

Коли граф H є тривіальним, тобто множина E_h є пустою, то ЗПНЗМ є задачею Штейнера на графі G. Евристичний

алгоритм знаходження дерева Штейнера $T(N)$, яке перевищує оптимальне не більше, ніж у $2(1-1/|N|)$ рази, запропоновано у [166] та [150]. В [132] запропоновано алгоритм знаходження дерева Штейнера $T(N)$, вартість якого перевищує вартість оптимального дерева T_* не більше, ніж у $2(1-1/l)$ рази, причому знайти його можна за час $O(n^2 + |N| \cdot \log|N|)$, де l – кількість листків у дереві $T_*(N)$. Ми можемо знайти таке дерево $T(N)$, а далі за допомогою додавання до нього додаткового ребра та видалення з нього окремих ребер ми побудуємо допустимий розв'язок ЗПНЗМ.

В підрозділах 6.3 та 6.4 вивчається, чи існує розв'язок ЗПНЗМ для різних типів графа H. В підрозділі 6.3 розглянуто ЗПНЗМ для випадку, коли граф H є паросполученням або деревом з фіксованою кількістю ребер. У цих випадках проблему перевірки, чи має ЗПНЗМ розв'язок, можна вирішити за допомогою поліноміального алгоритму. В підрозділі 6.4 розглянуто ЗПНЗМ для випадку, коли граф H є паросполученням або лісом, і кількість його ребер не фіксована. Для цих випадків показано, що проблема перевірки, чи існує розв'язок у ЗПНЗМ, є NP-повною.

У деяких застосуваннях задачі проектування телекомунікаціних та транспортних мереж взаємодія інформаційних потоків різних типів відбувається складним чином, а вартість збільшення обсягу потоку на ребрі нелінійно залежить від значення потоку та пропускної здатності цього ребра. Як вказується в [102], ці фактори також є важливими, але їх слід враховувати в математичній моделі оптимізаційної задачі на другому етапі – після того, як мережу оптимальної вартості спроектовано.

6.1. Модель ЗПНЗМ в термінах потокових змінних

Нехай $\overline{H}$ – множина підграфів графа G, ізоморфних графу H. Для довільного підграфа Г графа G будемо позначати множину його вершин $V(\Gamma)$, а множину його ребер – $E(\Gamma)$. Для того щоб сформулювати математичну модель ЗПНЗМ, відзначимо спочатку просту властивість підграфа G_*

(оптимального розв'язку ЗПНЗМ): якщо видалимо з підграфа G_* ребра графа $\Gamma \in \overline{H}$, і шлях, що з'єднує один вузол (наприклад, s) з іншими вузлами N існує, то в підграфі G_* існує, принаймні, один шлях між будь-якими парами вузлів із N.

Беручи до уваги цю властивість, математичну модель ЗПНЗМ можна сформулювати як задачу знаходження однопродуктового потоку мінімальної вартості на мережі з одним джерелом та кількома стоками за умови, що після видалення з мережі всіх ребер підграфа, ізоморфного H, в утвореній мережі буде існувати принаймні один шлях між кожною парою термінальних вузлів з N для передачі одиничного потоку.

Нехай один вузол (скажімо, s) зафіксовано як джерело, а інші вузли з N вважаються стоками. Тоді математичну модель ЗПНЗМ можна записати таким чином:

знайти

$$\min \sum_{e \in E} c_e x_e \tag{6.1}$$

при обмеженнях

$$\sum_{j \in \Gamma(i)} x^r_{ij} - \sum_{j \in \Gamma(i)} x^r_{ji} = \begin{cases} 1, & i = s, \\ 0, & i \neq s, r, \\ -1, & i = r, i \in V, r \in N_0, \Gamma \in \overline{H}, \end{cases} \tag{6.2}$$

$$0 \le x^r_{ij} \le x_e, \quad r \in N_0, \quad (i,j) = e \in E, \tag{6.3}$$

$$x_e = 0 \vee 1, \ e \in E, \tag{6.4}$$

де $N_0 = N \backslash \{s\}$; $\Gamma(i)$ – множина ребер з однією вершиною i на графі, що утворений з графа G шляхом видалення ребер підграфа Γ.

Через велику кількість змінних та кількість обмежень, що зростає експоненційно, безпосереднє розв'язання ЗПНЗМ як задачі цілочисельного програмування має помірний практичний інтерес. Спробуємо визначити мінімальну кількість

обмежень μ (6.2), за яких оптимальний розв'язок ЗПНЗМ залишається незмінним. У наступному твердженні показано, що складність визначення кількості μ є практично такою ж, як і складність розв'язання самої задачі (6.1)–(6.4). Будемо називати величину μ стійкою кількістю обмежень (the tight number of constraints).

Твердження 6.1. *Задача визначення стійкої кількості обмежень є NP-повною.*

Доведення. Розглянемо ЗПНЗМ для випадку, коли граф H має дві вершини та одне ребро. Нехай ми зафіксували ребро g та розглядаємо задачу (6.1)–(6.4) для випадку, коли з мережі G може бути вилучено лише одне ребро g. Тоді ЗПНЗМ еквівалентна задачі про дерево Штейнера на графі $G(g) = (V, E\backslash\{g\})$ з множиною $V\backslash N$ вершин Штейнера та множиною N спеціальних вершин. Таким чином, якщо ребро g не є ребром оптимального дерева Штейнера, це дерево буде оптимальним розв'язком задачі (6.1)–(6.4), і задача (6.1)–(6.4) не є еквівалентною ЗПНЗМ. Тому для визначення стійкої кількості обмежень в (6.4), слід визначити, чи належить ребро g оптимальному дереву Штейнера. Оскільки задача знаходження оптимального дерева Штейнера є *NP*-повною, твердження доведено.

Нехай дерево T_* є оптимальним розв'язком задачі Штейнера на графі G з множиною вершин Штейнера $V\backslash N$ та множиною N спеціальних вершин. Як відомо, дерево T_* має два листки. Тому можна за поліноміальний час знайти деякі із зайвих обмежень у (6.2), а також знайти приблизне значення μ_0 величини μ, таке що $\mu \le \mu_0 < |N|$. Спочатку знаходимо дерево $T(N)$ за допомогою поліноміального алгоритму (див. [132]). Далі, як відомо, виконується нерівність [132]

$$c(T(N)) \le 2(1 - 1/l)c(T_*),$$

де $c(T(N))$ – це повна вартість дерева $T(N)$; l – кількість листків на дереві T_*. Однак відомо, що $l = 2$, тому дерева $T(N)$ та T_* співпадають. Тепер за допомогою того самого поліноміального алгоритму знаходимо дерево $T_g(N)$ на графі $G(g) = (V, E\backslash\{g\})$,

де g – довільне ребро з E. Якщо $T_g(N)$ та T_* співпадають, це означає, що ребро g не є ребром дерева T_*. Тому обмеження в (6.2), введене для ребра g, є надлишковим. Таким чином знаходимо приблизне значення μ_0, повторюючи процедуру $|E|$ разів. Оскільки кількість ребер дерева T_* менша за $|V|$, одержуємо, що $\mu \le \mu_0 < |V|$.

6.2. Випадок, коли граф в ЗПНЗМ є окремим ребром

Нехай граф H має дві вершини та одне ребро. Інакше кажучи, граф G_* має залишитися зв'язним після видалення з графа G будь-якого ребра e з множини E.

Якщо $N = \{s, r\}$ ($|N| = 2$), ЗПНЗМ еквівалентна знаходженню двох шляхів, що з'єднують вузли s та r і не мають спільних ребер, а сума вартостей їх ребер є мінімальною. Цю задачу можна розв'язати за час $O(n^3)$ (n – кількість вершин мережі G), обчисливши поток мінімальної вартості в мережі G з джерела s до стоку r за умови, що ребра G мають одиничні пропускні спроможності, а величина потоку дорівнює 2. У цьому випадку, якщо спробувати розв'язати цю задачу за допомогою LP-релаксації задачі (6.1)–(6.4) (де обмеження цілочисельності для x_e релаксовані), то складність розв'язання буде оцінюватися як $O(n^2|N|)$ (див. [39]), що більше за складність розв'язання ЗПНЗМ як задачі про мінімальний потік $O(n^3)$ (див. [54]).

У випадку $N = V$ ЗПНЗМ еквівалентна задачі побудови 2-зв'язної мережі мінімальної вартості [98]. Якщо ж граф G є повним, а вартості його ребер задовольняють нерівності трикутника, то ЗПНЗМ еквівалентна LP-релаксації задачі комівояжера (ЗК), тобто для розв'язання ЗПНЗМ можна використовувати методи знаходження нижніх границь для ЗК, описані в працях [98, 74, 125].

Властивості многогранника допустимих розв'язків для ЗПНЗМ та для задачі Штейнера вивчено в працях [102, 129]. Ці властивості та твердження (доведені в [102]) можна

використовувати для зменшення розриву двоїстості між оптимальними значеннями в задачах проектування зв'язних мереж та Штейнера і відповідних задач, одержаних як їх LP-релаксація.

Ми можемо розглянути окремий випадок цієї задачі, коли граф G_* залишається зв'язним після видалення фіксованого ребра g з E. В цьому випадку ЗПНЗМ є задачею Штейнера на графі $G(g) = (V, E\backslash\{g\})$ (знайти дерево T_* мінімальної вартості, що з'єднує всі вузли в N). Добре відомо, що задача Штейнера є NP-повною, і тому ЗПНЗМ також є NP-повною. В працях [78, 61] та в багатьох інших для розв'язання задачі Штейнера на графі було запропоновано багато точних алгоритмів (див. оглядові статті [114, 166]).

В багатьох практичних задачах достатньо знайти найближений розв'язок задачі Штейнера $T(N)$. Евристичні поліноміальні алгоритми для знаходження дерева $T(N)$, вартість якого не більше, ніж в $2(1-1/|N|)$ перевищує вартість оптимального дерева, запропоновано в [150, 166]. В [132] запропоновано алгоритм знаходження дерева $T(N)$, вартість якого не більше, ніж в $2(1-1/l)$ перевищує вартість оптимального дерева, за час $O(n^2 + |N| \cdot \log|N|)$, де l – кількість листків в оптимальному дереві Штейнера T_*. За допомогою цих алгоритмів можемо знайти дерево $T(N)$, а далі додати до дерева $T(N)$ додаткове ребро або видалити з нього деякі ребра та знайти таким чином допустимий розв'язок ЗПНЗМ (див. підрозділ 6.4).

Розглянемо тепер іншу модель ЗПНЗМ для випадку, коли граф H складається з одного ребра. Нехай $\delta(W) = \{(i,j) \in E, i \in W, j \in V\backslash W\}$ є розріз, який заданий підмножиною $W \subset V$ множини вершин V, а $R(N)$ – множина розрізів $\delta(W)$, таких, що $W \cup N$ та $N \cap (V\backslash W)$ непусті. Легко побачити, що ЗПНЗМ можна сформулювати в термінах розрізів таким чином:

знайти

$$\min \sum_{e \in E} c_e x_e \qquad (6.5)$$

при обмеженнях

$$x(\delta(W)) \geq 2 \text{ для всіх } \delta(W) \in R(N), \tag{6.6}$$

$$x_e = 0 \vee 1, \quad e \in E. \tag{6.7}$$

Якщо ми додамо до задачі (6.5)–(6.7) такі обмеження:

$$x(\delta(v)) = 2 \text{ для всіх } v \in N, \tag{6.8}$$

то вона зведеться до задачі про комівояжера (ЗК) у випадку, коли $N = V$. В [98] відзначається, що нижня границя Хелда-Карпа (Held-Karp lower bound) у багатьох випадках успішно використовувалася дослідниками для розв'язання ЗК методом гілок та границь. Нижню границю Хелда-Карпа можна подати як задачу лінійного програмування:

$$\min \sum_{e \in E} c_e x_e,$$

$$x(\delta(W)) \geq 2, \ \delta(W) \in R(N),$$

$$x(\delta(v)) = 2, \ v \in V,$$

$$x_e \geq 0, \ e \in E.$$

Ця модель називається 1-деревною релаксацією задачі про комівояжера. Нехай граф $G = (V, E)$ є повним, а вартості ребер задовольняють нерівності трикутника. Для цього випадку в [98] було показано, що 1-деревна релаксація ЗК еквівалентна такій задачі лінійного програмування:

$$\min \sum_{e \in E} c_e x_e, \tag{6.9}$$

$$x(\delta(W)) \geq 2, \ \delta(W) \in R(N), \tag{6.10}$$

$$x_e \geq 0, \ e \in E. \tag{6.11}$$

Нескладно впевнитися, що задача (6.9)–(6.11) є *LP*-релаксацією задачі (6.5)–(6.8). Тому, коли граф є повним, $N = V$, а вартості ребер задовольняють нерівності трикутника, можемо

знайти нижню оцінку в ЗПНЗМ за допомогою будь-якого алгоритму для знаходження нижньої границі для ЗК та в деяких випадках розв'язати ЗПНЗМ методом гілок та границь.

Легко впевнитися, що якщо граф G не є повним, то цей розгляд неприйнятний. Для того щоб розв'язати ЗПНЗМ для довільного графа G, наведемо тепер іншу модель ЗПНЗМ, що має меншу кількість обмежень, ніж попередня. Для побудови цієї моделі відзначимо одну корисну властивість оптимального розв'язку G_* ЗПНЗМ: оптимальний розв'язок G_* задачі є 2-зв'язним. Таким чином, вважається, що реберна зв'язність графа G не менша за 2. При цьому, коли граф H є одним ребром, ЗПНЗМ можна сформулювати так:

$$\min \sum_{e \in E} c_e x_e, \tag{6.12}$$

$$\sum_{j \in \delta(i)} x_{ij}^r - \sum_{j \in \delta(i)} x_{ji}^r = \begin{cases} 2, & i = s, \\ 0, & i \neq s, r, \\ -2, & i = r, \ i \in V, \end{cases} \quad r \in N_0, \tag{6.13}$$

$$0 \leq x_{ij}^r \leq x_{ij} \leq 1, \ r \in N_0, \quad e = (i, j) \in E, \tag{6.14}$$

$$x_e = 0 \vee 1, \ e \in E. \tag{6.15}$$

Цю модель ЗПНЗМ можна розглядати як штейнерову версію задачі проектування 2-зв'язної мережі мінімальної вартості, так звану задачу Штейнера про 2-зв'язний підграф. В п. 6.3.2 опишемо алгоритм знаходження приблизного розв'язку *LP*-релаксації цієї задачі для визначення нижньої границі по функціоналу ЗПНЗМ. В п. 6.3.3 надається евристичний алгоритм для знаходження допустимого розв'язку задачі (6.12)–(6.15), а таким чином і верхньої границі для ЗПНЗМ. У цій моделі число 2 в обмеженні (6.13) може розглядатися як величина потоку, що перетікає з джерела s у сток r. Якщо величина цього потоку дорівнює 1, отримуємо двоїсту задачу для

LP-релаксації задачі про дерево Штейнера (ДШ) на графі G. В наступному підрозділі ми покажемо, що розрив між цією задачею та її *LP*-релаксацією, а також між ЗПНЗМ та її *LP*-релаксацією, можна зменшити.

6.2.1. Многогранник обмежень спеціальної ЗПНЗМ

Як уже відзначалося, якщо в ЗПНЗМ розглянути відмову лише одного ребра e, то ЗПНЗМ еквівалентна відомій задачі Штейнера на графі (ЗШГ) з множиною основних вершин. В даному підрозділі наведемо деякі властивості ЗШГ, які вивчені в [102]. В наступному підрозділі подібні властивості доведено для ЗПНЗМ.

Нехай задано граф $G = (V, E)$, N – підмножина множини вершин V, де $|N| \geq 2$, а G – зв'язний граф. Вершини N з називають основними, а вершини з $V \backslash N$ – вершинами Штейнера. Довільне дерево, що з'єднує основні вершини, називається деревом Штейнера. Вершини $V \backslash N$ з можуть бути чи не бути вершинами дерева Штейнера.

Для $F \subseteq E$ позначимо $x^F = \{x_e^F, e \in E\}$ вектор інцидентності F, де $x_e^F = 1$, якщо $e \in F$, і $x_e^F = 1$ в протилежному випадку. Розглянемо опуклу оболонку $\mathrm{Conv}(T(N))$ векторів інцидентності всіх $F \subseteq E$, які містять дерево Штейнера. Для довільних ваг $e_l \geq 0$ ребер e з E ЗШГ на G еквівалентна такій задачі:

$$\min\{cx : \ x \in \mathrm{Conv}(T(N))\}, \tag{6.16}$$

де cx – скалярний добуток векторів c, $x \in R^E$. Ребро e назвемо мостом відносно N, якщо після видалення e з графа G, в G не існує шляху, що з'єднує деякі вершини s, r з N. Для довільного $W \subseteq V$, $G(W)$ – це граф с множиною вершин W та множиною ребер $E(W) = \{(i, j), \ i, j \in W\}$.

Теорема 6.1 [98]. *Нехай $V_1, ..., V_p$, $p \geq 2$ – таке розбиття множини V, що $V_i \cap N$ не є пустим для всіх $i = 1, ..., p$. Тоді нерівність*

$$1/2 \sum_{i=1}^{p} x(\delta(V_i)) \geq p - 1$$

визначає грань для $Conv(T(V))$ тоді й тільки тоді, коли:

(a) *$G(V_i)$ є зв'язним графом для $i = 1, ..., p$;*

(b) *граф $G(V_i)$ не містить моста відносно $N_i = N \cap V_i$ для $i = 1, ..., p$;*

(c) *при V_i стягуванні в одну вершину одержаний граф $\overline{G} = (\overline{V}, \overline{E})$ має вершинну зв'язність, яка дорівнює 2.*

При цьому, якщо $N_i = 1$, то вважається, що $G(V(V_i))$ не містить моста відносно N_i і в $\overline{G}$ можуть бути паралельні ребра.

Нехай A – множина ребер, що містяться в множинах $\delta(V_1)$, ..., $\delta(V_p)$, $k(A)$ – число додаткових компонентів зв'язності графа G, отриманих після видалення з нього всіх ребер з A. У цих позначеннях умови теореми 6.1 еквівалентні тому, що нерівність

$$x(A)k(A)^{-1} \geq 1 \tag{6.17}$$

визначає грань $Conv(T(V))$ для тоді і тільки тоді, коли підграфи $G(V_1)$, ..., $G(V_p)$ задовольняють умови (a), (b), (c).

Ліва частина нерівності (6.17) – це міцність графа G з вагами (міцністю) x_e ребра e для всіх $e \in E$.

Тепер покажемо, що використовуючи алгоритми [33, 75] для розв'язання задачі знаходження міцності графа можемо поліпшити нижню оцінку, знайдену шляхом розв'язання задачі *LP*-релаксації (6.16), і алгоритми для розв'язання задачі посилення міцності графа можна застосувати при обчисленні верхньої оцінки:

$$\min \sum_{e \in E} c_e x_e, \tag{6.18}$$

$$x(\delta(W)) \geq 1, \ \delta(W) \in R(N), \tag{6.19}$$

$$0 \leq x_e \leq 1, \ e \in E. \tag{6.20}$$

Розглянемо таку задачу класифікації обмежень[1] задачі (6.19)–(6.20): для заданого вектора $\overline{x} \in R^E$ потрібно перекона-

[1] Англійською мовою ця задача називається *separation problem* (див., наприклад, [101]), в російських виданнях часто зустрічається варіант перекладу *задача отделения*.

тися, що або обмеження (6.19)–(6.20) виконуються, або знайти максимально порушене обмеження.

Виконання $0 \le x_e \le 1$ обмежень для $\bar{x}$ перевіряється простим способом за час $|E|$. Тому припустимо, що $x_e \ge 0$ для всіх $e \in E$. Нехай $G(\bar{x})$ – граф G з вагами (пропускними здатностями) $\bar{x}_e$ для всіх ребер $e \in E$. Алгоритмом Гоморі та Ху, за час $O(|V|^3)$ знайдемо величини максимального потоку між усіма парами $|N|$ основних вершин. Якщо для всіх пар вершин з N величина максимального потоку більша за 1, то очевидно, що $\bar{x}_e$ задовольняє обмеженням задачі (6.18)–(6.20), у протилежному випадку знаходимо максимально порушене обмеження.

Таким чином, розв'язок задачі (6.18)–(6.20) можна знайти методом еліпсоїдів [50] за поліноміальний час. При цьому трудомісткість методу виражається як поліном від числа змінних $|E|$ і від інформаційної довжини задачі, тобто трудомісткість методу не залежить від кількості обмежень задачі.

Припустимо, що знайдено оптимальний розв'язок $\bar{x}_e$ задачі (6.18)–(6.20) і на мережі $G(\bar{x})$ визначені величини максимальних потоків між усіма парами $|N|$ основних вершин. Таким чином, побудовано дерево з псевдовершинами v_i, $i = 1,..., p$, кожній з яких відповідає підмножина $V_i \subset V$, де V_i, $i = 1,..., p$, такі, що $V_i \cap V_j$ – пуста множина, $V = \bigcup_{i=1}^{p} V_i$, а також:

(a1) підграфи $G(V_1)$, ..., $G(V_p)$ є компонентами зв'язності графа G;

(b1) $|V_i \cap N| = 1 \quad \forall i = 1,..., p$ тобто V_i, $i = 1,..., p$, є розбиттям множини V.

За допомогою стягування підмножини V_i в одну вершину для всіх $i = 1,..., p$ з графа $G(\bar{x})$ одержимо граф $\overline{G}(\bar{x})$. При цьому $\overline{G}(\bar{x})$ може мати паралельні ребра. Розмістимо нову вершину на ребрі (v, w) і замінимо його двома ребрами (v, u) і (u, w). Покладемо, що вага ребра дорівнює x_e, а вага (u, w) – дорівнює M (M – досить велике число).

Нехай $\sigma(\bar{x}) = \min\{\bar{x}(A)k(A)^{-1}$, множина $A \subseteq E(\bar{x})$ є непустою},

$E(\bar{x})$ – множина ребер графа. Припустимо, що за допомогою одного з поліноміальних алгоритмів [33, 75] визначено значення $\sigma(\bar{x})$. Можливі два випадки.

1. $\sigma(\bar{x}) < 1$. У цьому випадку розглянемо задачу (6.18)–(6.20) з додатковими обмеженнями $\min\{\bar{x}(A)k(A)^{-1} \ge 1$, множина $A \subseteq E(\bar{x})$ є непустою}. Задача класифікації цих обмежень розв'язується знову за допомогою згаданих вище алгоритмів. Тому методом еліпсоїдів за поліноміальний час знаходиться розв'язок x_* останньої задачі. При цьому легко показати, що $c\bar{x} > cx_*$, тобто поліпшується нижня оцінка для ЗШГ.

2. У випадку $\sigma(\bar{x}) \ge 1$ знайдена нижня оцінка $c\bar{x}$ досить близька до оптимального значення ЗШГ у такому розумінні. Ясно, що для довільного вектора $\mathrm{Conv}(T(N))$ з виконуються нерівності

$$1/2\sum_{i=1}^{p} x(\sigma(V_i)) \ge p - 1,$$

для всіх варіантів розбиття $V_1,..., V_p$, $p \ge 2$, множини V, таких, що $|V_i \cap N| \ge 1$ для всіх $i = 1,..., p$ та $0 \le x_e \le 1$ для всіх $e \in E$. У [165] доведено, що ці нерівності недостатні для повного опису многогранника $\mathrm{Conv}(T(N))$. Якщо $\sigma(\bar{x}) \ge 1$, то вони вірні для $\bar{x}$ і для будь-якого розбиття $V_1, ..., V_p$, $p \ge 2$, множини V. Крім того, V_i – зв'язні графи для всіх $i = 1,..., p$, тобто можуть не виконуватися лише умови (b) і (c) теореми 6.1.

6.2.2. Многогранник ЗПНЗМ

Для того, щоб знайти розв'язок задачі проектування мережі, в ряді праць використовується метод гілок та відсічень, для чого, виходячи із задачі цілочисельного чи змішаного лінійного програмування, знаходять вірні нерівності чи грані та розглядають їх як додаткові обмеження задачі – таким способом відсікаються деякі нецілочисленні розв'язки початкової задачі [86, 56]. При такому підході визначення того, чи належить заданий вектор многограннику, пов'язаному з цією

задачею, потребує розв'язання задачі класифікації знайдених нерівностей.

Покажемо, що грань многогранника, пов'язаного із ЗПНЗМ, для випадку, коли граф H складається з одного ребра, можна визначити аналогічно грані, яка задається нерівностями в [102].

Нехай C – цикл на графі G. Будемо говорити, що цикл C покриває вершину v, якщо v є вершиною цикла C.

Для будь-якого $F \subseteq E$ $|E|$-вектор x^F називається вектором інциденції для F, якщо $x_e^F = 1$ для будь-яких $e \in F$ та $x_e^F = 1$ при $e \notin F$.

В попередньому підрозділі ми відзначали, що оптимальний розв'язок G_* ЗПНЗМ є 2-реберно-зв'язним. Тому підграф G_* містить один чи більше циклів, що покривають всі термінальні вузли з N. Нехай $\mathrm{Conv}(C(N))$ – це опукла оболонка всіх векторів інциденції x^F, де F містить ребра циклів, що покривають всі термінальні вузли з N.

Таким чином, для заданих вартостей ребер c_e, $e \in E$, ЗПНЗМ можна записати у вигляді

$$\min\{cx: \quad x \in \mathrm{Conv}(C(N))\}. \tag{6.21}$$

Для заданого розбиття $V_1, \ldots, V_p$ множини V на p непустих підмножин, таких, що $V_i \cap N$ не пусте, кожний допустимий розв'язок містить не менше p ребер, кінцеві вершини яких знаходяться в V_i та V_j для різних i та j. Тому для многогранника Conv(ЗПЗМ) виконується нерівність

$$1/2 \sum_{i=1}^{p} x(\delta(V_i)) \geq p. \tag{6.22}$$

Однак наступний приклад показує, що задача лінійного програмування

$$\min\{cx : 1/2 \sum_{i=1}^{p} x(\delta(V_i)) \geq p,\ V = \bigcup_{i=1}^{p} V_i,\ V_i \cap N - \text{непусте},\ V_i \neq V_j\}$$

не дає допустимого розв'язку задачі (6.21).

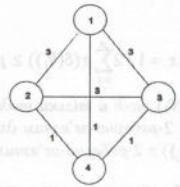


Рис. 6.1. Приклад, коли задача лінійного програмування не дає допустимого розв'язку

На рис. 6.1 номери вершин показано в кружках, а числа біля ребер вказують їх вартість; вершини з номерами 1, 2 та 3 є термінальними вузлами. Неважко впевнитися, що оптимальним розв'язком останньої задачі лінійного програмування є $\bar{x}_{12} = 1/2$, $\bar{x}_{23} = 1/2$, $\bar{x}_{13} = 1/2$, $\bar{x}_{24} = 1$, $\bar{x}_{34} = 1$, $\bar{x}_{14} = 1$, а оптимальне значення цільової функції дорівнює 7,5. Можна перевірити, що нерівність (6.22) виконується для оптимального розв'язку цієї задачі лінійного програмування для будь-якого розбиття $V_1, \ldots, V_p$ множини V на p непустих підмножин, таких що не пусте.

Єдиним оптимальним розв'язком ЗПНЗМ у цьому випадку є $x_{12}^* = 1$, $x_{24}^* = 1$, $x_{13}^* = 1$, $x_{34}^* = 1$, а оптимальне значення цільової функції дорівнює 8. Таким чином, існує розрив, пов'язаний із цілочисельністю, тому що оптимальний розв'язок останньої задачі лінійного програмування виходить за межі $\mathrm{Conv}(C(N))$.

Тепер покажемо, що при деяких додаткових умовах нерівність (6.22) задає грань $\mathrm{Conv}(C(N))$. Без втрати узагальненості можна вважати, що вершинна зв'язність графа не менша за 2. В протилежному випадку легко показати, що ЗПНЗМ можна розкласти на менші задачі на підграфах графа G, вершинна зв'язність яких не менша за 2.

Теорема 6.2. *Нехай $V_1, \ldots, V_p$, $(p \geq 2)$ – розбиття V, таке що $|V_i \cap N| \geq 1$, $i = 1, \ldots, p$.*

Нерівність

$$a^t x = 1/2\sum_{i=1}^{p} x(\delta(V_i)) \geq p$$

задає грань $\mathrm{Conv}(C(N))$ *тоді й тільки тоді, коли:*

(a) $G_i = (V_i, E_i)$ *є 2-реберно-зв'язним для* $i = 1, ..., p$;

(b) $G_i = (V_i, E_i \backslash \{e_i\})$ *є 2-реберно-зв'язним для всіх* $i = 1, ..., p$, *де* $e_i \in E_i$;

(c) *граф* $\overline{G} = (\overline{V}, \overline{E})$, *одержаний з* G *за допомогою стягування до одного вузла, є 2-вершинно-зв'язним і містить у собі реберно-неперетинні цикли, що покривають всі вузли в* $\overline{V}$.

У випадку, коли $|N_i| = 1$, *вважається, що виконуються умови* (a) *й* (b) *та дозволено, щоб у графа* $\overline{G}$ *існували паралельні ребра.*

Доведення. Ця теорема доводиться аналогічно теоремі 6.6 в [102]. Припустимо, що один з графів $G(V_i)$, скажімо граф $G(V_1)$, не є зв'язним графом. Нехай V_1' – множина вершин однієї з компонент $G(V_1)$, така, що $(V_1 \backslash V_1') \cap N'$ не є пустим. Оскільки G – зв'язний граф, то існує ребро, що з'єднує V_1' з однією з V_i, $i = 1, ..., p$, наприклад з V_2. У цьому випадку

$$a^t x = 1/2 x(\delta(V_1 \setminus V_1')) + x(V_1', V_2) + 1/2 x(\delta(V_1' \cup V_2)) + \sum_{i=1}^{p} x(\delta(V_i)).$$

Права частина цієї рівності є сумою нерівностей:

$$1/2 x(\delta(V_1 \setminus V_1')) + 1/2 x(\delta(V_1' \cup V_2)) + \sum_{i=1}^{p} x(\delta(V_i)) \geq p,$$

$$x_e \geq 0 \text{ для } e = (i, j),$$

де $i \in V_1'$, $j \in V_2'$. Як було відзначено вище, ця система нерівностей не визначає грань для $\mathrm{Conv}(C(N))$.

Припустимо, що ребро $g \in$ міст для графа $G_e(V_i)$ для деякого $1 \leq i \leq p$ і $e \in E(V_i)$, тобто ребра g та e з'єднують два підграфа $G(V_i^1)$ і $G(V_i^2)$ графа $G(V_i)$, причому $G(V_i^1) \cap N_i$ та $G(V_i^2) \cap N_i$ не є пустими. Тому якщо для $F \subseteq C(N)$ виконується $a^t x^F = p$, то $e, g \in F$.

Можливі такі два випадки.

1. Існують тільки два реберно-неперетинні шляхи p_1, p_2, що з'єднують вершини $v_1 \in V_i^1 \cap N_i$ і $v_2 \in V_i^2 \cap N_i$, такі що ребра цих шляхів належать $E(V_i)$. Із врахуванням того, що $e, g \in F$ для будь-якого F з $C(N)$, маємо $x_e, x_g > 0$. Звідси випливає, що $a^t x^F = p$ не визначає грань для $\mathrm{Conv}(C(N))$.

2. Припустимо, що кількість шляхів, що з'єднують v_1 і v_2, більше, ніж 2. У цьому випадку не всі ребра хоча б одного шляху (нехай це шлях P_0) належать $E(V_i)$. Нехай C – такий цикл у $\overline{G}$, який покриває всі вершини з $\overline{V}$, а $C \cap P_0$ не є пустою множиною. Якщо такий цикл C не існує, тоді $x_e, x_g > 0$ (випадок 1). Покладемо $F = C \cap E(V_i)$. Очевидно, $a^t x^F = p$. Тепер нехай $F' = (C \cup P_1 \cup P_2) \backslash (P_0 \cap \delta(V_i))$. Ясно, що $x^F \in \mathrm{Conv}(C(N))$. Але $a^t x^F < p$, що суперечить умові $a^t x^F \geq p$.

Припустимо, що в $\overline{G}$ не існують два реберно-неперетинні цикли, що покривають усі вершини з $\overline{V}$. Тоді в $\overline{G}$ має бути хоча б один цикл C, що покриває всі вершини з $\overline{V}$. У цьому випадку $x_e = 1$ для всіх ребер e з C. Тому задача розпадається на підзадачі на графах $G(V_i)$, $i = 1, ..., p$, і $a^t x^F \geq p$ не є гранню для $\mathrm{Conv}(C(N))$.

Припустимо, що вершинна зв'язність $\overline{G}$ менша, ніж 2. Тоді серед вершин $\overline{x}_1, ..., \overline{x}_p$, отриманих при стягуванні V_i в одну вершину $i = 1, ..., p$, є така, скажімо v_1, після видалення якої отриманий граф $G_{V_1}(V_i)$ розпадається на дві компоненти. Нехай $\{v_2, ..., v_k\}$ – вершини однієї компоненти, а $\{v_{k+1}, ..., v_p\}$ – іншої. У цьому випадку $a^t x^F \geq p$ є сумою таких нерівностей

$$1/2\sum_{i=1}^{p} x(\delta(V_i)) + 1/2\sum_{i=1}^{p} x(\delta(W_1)) \geq k,$$

$$1/2\sum_{i=1}^{p} x(\delta(V_i)) + 1/2\sum_{i=1}^{p} x(\delta(W_2)) \geq p - k,$$

де $W_1 = V_1 \cup \{\overline{v}_2, ..., \overline{v}_k\}$ і $W_2 = V_1 \cup \{\overline{v}_{k+1}, ..., \overline{v}_p\}$. Тому $a^t x \geq p$ не визначає грань для $\mathrm{Conv}(C(N))$.

Припустимо, що $F_b = \{x \in \mathrm{Conv}(C(N)),\ b'x = \beta\}$ – деяка грань $\mathrm{Conv}(C(N))$ і $x \in \mathrm{Conv}(C(N)), a'x = p\} = F_a \subseteq F_b$. Кожному циклові C у $\overline{G}$, що покриває всі вершини з $\overline{V}$, відповідає деякий цикл $C(G)$ у G, для якого $C \in C(G)$. Нехай $B = \bigcup_{i=1}^{p} E(V_i)$. Очевидно, що $a'x^B = p$. Оскільки $G_e(V_i)$ не містить міст відносно N_i для $e \in E(V_i)$, то існують два реберно-непересічні шляхи P_1 і P_2, що з'єднують вершини $v_1, v_2 \in N_i$, такі, що для $F' = (C \cup P_1 \cup P_2)$ вірно $a'x^F = p$. Тепер нехай $e = (u, v)$ є ребром P_0. Згідно з умовами (а), (b) у $G(V_i)$ можна знайти шлях P_0, який з'єднує вершини u і v, для якого $P_0 \cap P_1$ і $P_0 \cap P_2$ є пустими множинами. Знову одержуємо, що для $F_1 = C \cup (P_1 \backslash e) \cup P_2 \cup P_0$ виконується рівність $a'x^{F_1} = p$, отже, $a'x^{F_1} = a'x^F = p$. Звідси випливає, що $b_e = 0$ для $e \in A$. Оскільки e – довільне ребро з A, то $b_e = 0$ для всіх $e \in A$.

Нехай e – ребро циклу C у $\overline{G}$, що покриває всі вершини з $\overline{V}$. Згідно з умовою (с) існує цикл C_1, який покриває всі вершини з $\overline{V}$, такий, що $C \backslash C_1 = \{e\}$ і $C \backslash C_1 = \{g\}$. Тоді для довільних $B \subseteq A$, $F = B \cup C$ та $F_1 = B \cup C_1$ маємо $a'x^F = p$ і $a'x^{F_1} = p$, тому $b_e = b_g$. Це означає, що $b'x = \beta$ є рівнянням $a'x = p$, помноженим на деяку константу, тобто $a'x \geq p$ визначає грань для $\mathrm{Conv}(N)$, що й потрібно було довести.

Припустимо, що $\bar{x} = \{\bar{x}_e,\ e \in E\}$ – оптимальний розв'язок задачі (6.9)–(6.11), який можна знайти за поліноміальний час методом еліпсоїдів, оскільки для заданого $|E|$-вектора x задачу класифікації для лінійної системи (6.9)–(6.11) можна розв'язати за поліноміальний час алгоритмом Гоморі та Ху, поданим у [36]. Таким чином, можемо за поліноміальний час знайти нижню границю для ЗПНЗМ.

Позначимо $G(\bar{x})$ граф G з вагами ребер $\bar{x}_e$ для всіх ребер e з E. За допомогою алгоритма Гоморі та Ху можемо знайти максимальний потік між всіма парами термінальних вузлів з N, а отже, і розбиття множини вершин V на $V_1, ..., V_p$, $p = |N|$ таке, що $|V_i \cap N| = 1$ для $i = 1,..., p$. Тепер, стягуючи кожну множину вузлів V_i в один вузол, одержимо з графа $G(\bar{x})$ граф

$\overline{G}(\bar{x})$. Розташуємо нові вузли на паралельних ребрах для того, щоб граф $\overline{G}(\bar{x})$ не мав паралельних ребер чи петель. Тепер розглянемо таку задачу на графі $\overline{G}(\bar{x})$: знайти

$$\sigma = \min\left\{\frac{\bar{x}(W)}{k(W)+1},\ W \subseteq E \ -\ \text{непуста множина}\right\}, \quad (6.23)$$

де $\bar{x}(W) = \sum(x_e,\ e \in W)$; $k(W)+1$ – кількість компонент, одержаних після виходу з ладу W ребер графа $\overline{G}(\bar{x})$. Очевидно, що коли $\sigma \leq 1$, то для будь-якого розбиття $\overline{V}_1, ..., \overline{V}_p$, $(p \geq 2)$ множини вершин $\overline{V}$ виконується нерівність $1/2\sum_{i=1}^{p} \bar{x}(\delta(V_i)) \geq p$. Якщо $\sigma < 1$, то існує таке розбиття множини V, що для цього розбиття та $\overline{G}(\bar{x})$ нерівність (6.17) не виконується. В цьому випадку можемо знайти кращу нижню границю, розв'язавши задачу лінійного програмування (6.9)–(6.11), (6.22), оскільки задачу класифікації для лінійної системи (6.22) можна розв'язати за поліноміальний час. Для будь-якої вершини $v \in \overline{V}$ додаємо до графа $\overline{G}(\bar{x})$ вершину w та ребра (v, w). Позначимо одержаний граф G_w. Ваги нових ребер (v, w) дорівнюють нулю, а ваги інших ребер графа G_w залишаються такими, якими вони були в графі $\overline{G}(\bar{x})$. Неважко довести, що задача (6.23) еквівалентна задачі знаходження міцності графа G_w, і ми можемо скористатися для цього поліноміальним алгоритмом з праць [33, 75]. Якщо $\sigma \leq 1$, то нижня границя, знайдена як оптимальне значення в задачі (6.9)–(6.11), буде задовільною, оскільки для будь-якого розбиття множини $\overline{V}$ та $\bar{x}$ нерівність (6.22) та умови (а)–(в) теореми 6.1 виконуються для графа.

Описаний вище метод можна застосовувати для ЗПНЗМ помірних розмірів. Для розв'язання великих задач в наступних підрозділах буде запропоновано схему методу гілок та границь для знаходження розв'язку ЗПНЗМ.

6.2.3. Чисельні методи розв'язання ЗПНЗМ

Для розв'язання ЗПНЗМ застосовується загальна схема методу гілок та границь. З метою поліпшення ефективності методу гілок та границь у наступних підрозділах буде запропо-

новано ефективні алгоритми побудови нижньої F_{lower} та верхньої F_{upper} оцінок для цільової функції ЗПНЗМ. Нижня оцінка F_{lower} знаходиться шляхом розв'язання задачі (6.12)–(6.14), а верхня оцінка F_{upper} визначається на основі інформації, одержаної при обчисленні F_{lower}.

У методі гілок та границь при розв'язанні подібних задач використовуються різні правила розгалуження, для того щоб відкинути велику кількість безперспективних розгалужень дерева розв'язку. У попередніх розділах показано, що ЗПНЗМ, ЗШГ, ЗК і задача синтезу мереж з одним джерелом (ЗСМ) є схожими задачами. Тому при застосуванні методу гілок та границь для розв'язання ЗПНЗМ доцільно застосувати правило розгалуження, в якому використовуються ідеї, розглянуті в працях [150, 74, 18] при розв'язанні ЗШГ, ЗК і ЗСМ.

Опишемо процедуру для визначення наближеного розв'язку ЗПНЗМ. У графі G шляхом застосування алгоритму з [132] за поліноміальний час знаходимо дерево Штейнера T, вартість якого не більше ніж вартість оптимального дерева Штейнера, помноженої на $2(1-1/l)$, де l – число висячих вершин в оптимальному дереві Штейнера. Зрозуміло, що всі висячі вершини T дерева належать N.

Припустимо, що степені вершин $v_1, ..., v_k$ дерева T – непарні числа. Очевидно, що кількість цих вершин повинна бути парною, тобто $k = 2m$, де m – деяке ціле число, не більше, ніж $(n-1)/2$. Розглянемо повний граф $G_k = (V_k, E_k)$ з множиною вершин $V_k = \{v_1, ..., v_k\}$ з вагами ребер q_{rt}, де q_{rt} – довжина найкоротших шляхів у графі $G(T) = (V, E\backslash T)$, що з'єднують вершини v, t з V_k. Нехай визначено досконале паросполучення M з мінімальною вагою в графі $G_k = (V_k, E_k)$. Кожному ребру паросполучення відповідає шлях у графі $G(T)$. Множину ребер, що лежать на шляхах, які відповідають ребрам паросполучення, позначимо $E(M)$. Оскільки $c_{ij} \geq 0$ для всіх ребер (i, j), легко показати, що будь-які два шляхи не мають загальних ребер. Тому довільне ребро $\overline{E} = (V, E\backslash T)$ з є ребром або одного, або двох циклів у графі $\overline{G}$, що визначається множиною $\overline{E}$.

Для визначення хорд у цьому графі розв'язуємо задачу знаходження $\mu_{ij} = \min\{\mu_i, \mu_j\}$ – кількості реберно-неперетинних шляхів між вершинами i, j, для яких $\mu_i > 2$, $\mu_j > 2$. Якщо всі вершини (не враховуючи вершин i, j) одного з цих шляхів не належать множині N, то знайдено N-хорду для деякого циклу графа $\overline{G}$. У цьому випадку всі ребра цього шляху видаляються з графа $\overline{G}$, і таким способом знаходиться множина ребер $\overline{E}$, а визначений цією множиною підграф $\overline{G}$ не містить хорд. Поклавши $\overline{x}_{ij} = 1$ для ребер i, j графа $\overline{G}$ і $\overline{x}_{ij} = 0$ для інших ребер графа G, визначаємо допустимий розв'язок $\overline{x}$ ЗПНЗМ.

Нехай $\overline{f} = c\overline{x}$, $f^* = cx_*$ і $E_* = \{(i, j), x^*_{ij} = 1\}$, де x_* – оптимальний розв'язок ЗПНЗМ. Згідно з властивостями оптимального розв'язку зрозуміло, що $E_* = M_* + T_*$, де T_* – деяке дерево Штейнера, а M_* – деяке паросполучення в графі G_* з множиною ребер E_*. Тому $f_* = c(M_*) + c(T_*)$, де $c(T_*)$ – вага T_*, а $c(M_*)$ – вага M_*. Так само ясно, що $\overline{f} \leq c(M) + c(T)$. Із врахуванням того, що $c(T) \leq 2(1-1/l)c(T_*(N))$ [132] і $c(T_*(N)) \leq c(T_*)$, де T_* – оптимальне дерево Штейнера, одержимо, що $\overline{f} \leq 2(1-1/l)f_*$ при $c(M) \leq 2(1-1/l)c(M_*)$.

Нехай (i, j) – деяке ребро графа $\overline{G}$. Зрозуміло, що після видалення ребра (i, j) з $\overline{G}$, цей граф залишається зв'язним. У ньому визначимо спочатку остовне дерево мінімальної ваги, після видалення ребер цього остовного дерева знайдемо паросполучення M_{ij} на $\overline{G}$. Повторюючи цю процедуру для кожного ребра (i, j) графа $\overline{G}$, вибираємо для розгалуження змінну $\overline{x}_{ij}$, для якої паросполучення M_{ij} має мінімальну вагу.

Можна застосовувати і правило розгалуження, в якому фіксується змінна $\overline{x}_{ij}$, для якої хоча б одна з вершин i, j має максимальний степінь у графі з множиною ребер $\{(i, j); \overline{x}_{ij} > 0\}$. У цьому випадку часто виходить, що поточне значення F_{lower} стає більшим, ніж знайдене рекордне значення цільової функції, і подальше розгалуження задачі з нульовим значенням змінної $\overline{x}_{ij}$ безперспективне. Таке правило розгалуження узгоджується з методом штрафування вершин, який застосовується для знаходження наближеного розв'язку ЗК у [74], і правилом

розгалуження, яке використовується в [18] при розв'язанні ЗПНЗМ.

6.2.4. Алгоритми для знаходження наближеного значення нижньої границі

Для того щоб знайти оптимальний розв'язок ЗПНЗМ у випадку, коли граф H складається з одного ребра, пропонуємо алгоритм знаходження наближеного розв'язку двоїстої задачі LP-релаксації (6.12)–(6.15). Цей алгоритм оснований на алгоритмі знаходження потоку мінімальної вартості.

Двоїсту задачу можна сформулювати як задачу лінійного програмування:

знайти

$$\max\Big\{2\sum_{r\in N_0}(u_r^r - u_s^r) - \sum_{(i,j)\in E} z_{ij}\Big\} \qquad (6.24)$$

при обмеженнях

$$u_j^r - u_i^r \le w_{ij}^r,\ \ (i,j)\in E,\ \ r\in N_0, \qquad (6.25)$$

$$\sum_{r\in N_0} w_{ij}^r \le c_{ij} + z_{ij},\ \ (i,j)\in E, \qquad (6.26)$$

$$w_{ij}^r \ge 0,\ \ z_{ij}\ge 0,\ \ (i,j)\in E,\ \ r\in N_0. \qquad (6.27)$$

Припустимо, що знайдено допустимий розв'язок цієї задачі. З теорії двоїстості випливає, що значення цільової функції (6.24), яке відповідає цьому розв'язку, задає нижню границю F_{lower} для ЗПНЗМ. Далі використовуємо інформацію, одержану під час обчислення F_{lower} для того, щоб знайти F_{upper}. Значення F_{lower} та F_{upper} використовуються в схемі методу гілок та границь для розв'язання ЗПНЗМ. Аналогічна схема алгоритму гілок та границь описана в [18], і розв'язки багатьох задач проектування мереж та задач розміщення було знайдено за цією схемою. Відзначимо просту й необхідну властивість наведеної вище двоїстої задачі.

Твердження 6.2. *Існує такий оптимальний розв'язок* u_v^r, w_{ij}^r, $v\in V$, $r\in N_0$, $(i,j)\in E$ *задачі* (6.24)–(6.27), *що виконується рівність*

$$w_{ij}^r = \max\{0,\ u_j^r - u_i^r\},\ \ r\in N_0,\ \ (i,j)\in E. \qquad (6.28)$$

Доведення. Припустимо, що для деякого $(i,j)\in E$, $r\in N_0$ виконується нерівність $0 < u_j^r - u_i^r < w_{ij}^r$. В цьому випадку визначаємо $w_{ij}^r = u_j^r - u_i^r$ для будь-яких $(i,j)\in E$, так що $0 < u_j^r - u_i^r < w_{ij}^r$. Ясно, що знайшли інший допустимий розв'язок цієї задачі. Оскільки значення w_{ij}^r могло тільки зменшитися, то оптимальне значення в (6.24) залишається незмінним. Таким чином, твердження для цього випадку виконується. Якщо $u_j^r - u_i^r \le 0 < w_{ij}^r$, вважаємо $w_{ij}^r = 0$, і в цьому випадку твердження також виконується.

Із врахуванням останнього твердження розв'язок задачі (6.24)–(6.27) можна знайти за допомогою алгоритму, що досить близький до алгоритмів, застосованих при розв'язанні ЗПНЗМ [41], [56] і задач розміщення, розглянутих у [34]. Алгоритм розв'язання задачі складається з двох етапів. На початку першого етапу роботи алгоритму беремо $\overline{w}_{ij}^r = 0$ та $\overline{z}_{ij} = 0$ для всіх $(i,j)\in E$ і $r\in N_0$. Довільним чином фіксуємо вершину r з N_0.

Нехай $\overline{w}_{ij}^r$ визначено значення змінних для всіх $(i,j)\in E$ і для розглянутої вершини r з N_0. Позначимо G^r підграф G, що відповідає вершинам і дугам, які лежать на найкоротших шляхах, що з'єднують вершину s з вершиною r, з вагами ребер $\overline{w}_{ij}^r$. Спочатку G^r складається з однієї вершини s. У G^r виділяється деякий розріз R_r^r (на початку $R_r = \delta(s)$), що відокремлює джерело від вершини і має таку властивість, що

$$c_{ij} := c_{ij} - \sum_{r\in N_0} \overline{w}_{ij}^r > 0 \ \text{ для } (i,j)\in E$$

на його ребрах (R_r^r). Якщо для даної вершини не існує такого розрізу, то вершина r видаляється з множини N_0. Далі довільним чином фіксується інша вершина r з N_0.

Нехай $\delta_1 = \min\{c_{ij}: (i,j) \in R_r\}$, а l_r – довжина найкоротшого шляху з s у r. Покладемо $\delta_2 = \overline{l}_r - l_r$, де $\overline{l}_r$ – довжина найкоротшого шляху в графі $\overline{G}(T) = (V, E \backslash T)$ з модифікованими вагами ребер c_{ij}. Нехай $\delta = \min\{\delta_1, \delta_2\}$. Знайдемо нові значення

$$\overline{w}'_{ij} = \begin{cases} \overline{w}'_{ij} + \delta & \text{для } (i,j) \in R_r, \\ \overline{w}'_{ij} & \text{для } (i,j) \notin R_r \end{cases}$$

і покладемо $F_{lower} = F_{lower} + \delta$. На початку роботи алгоритму $F_{lower} = 0$. Якщо всі вершини виключені з множини N_0, то перший етап алгоритму на цьому завершується. У протилежному випадку фіксується раніше не фіксована вершина t з N_0, і цей процес продовжується для розрізу R_t і модифікованих ваг для ребер графа G. Нехай G^0 – підграф G з множиною ребер $E^0 = \{(i,j): c(i,j) = 0\}$. На другому етапі алгоритму знаходимо два реберно-непересічні мінімальні шляхи між вершинами s і r для всіх r з N_0 (їх довжину позначимо $l_r(1)$, $l_r(2)$). Очевидно, що $l_r(1) = 0$. Якщо $l_r(2) > 0$, то не всі ребра другого шляху належать до E^0. У цьому випадку беремо $c_{ij} = z_{ij} = l_r(2)$, де (i,j) – деяке ребро розрізу R_t, а далі повторюємо перший етап алгоритму для всіх r з N_0.

Оскільки після кожного кроку одержуємо, що $c_{ij} = 0$ хоча б для одного ребра $(i,j) \in E$, то число кроків алгоритму, в яких визначаються значення δ, не може бути більшим за n^2. Якщо після кожного кроку на графі G залишаються ребра $(i,j) \in E$, для яких $c_{ij} > 0$, то розріз R_t можна побудувати за n^2 кроків, тобто складність описаного алгоритму оцінюється як $O(n^4)$.

Опишемо тепер алгорим для знаходження F_{lower}. Він простіший за описаний вище. За допомогою твердження 2 знайдемо спочатку значення u'_v для $v \in V$, а далі скористаємося (6.28) для визначення величин w'_{ij}, $r \in N_0$ та $(i,j) \in E$ для $r \in N_0$. Для того щоб знайти оптимальне значення u'_v для $v \in V$ та $r \in N_0$, зафіксуємо спочатку деякий сток r_0 в N_0 та розв'яжемо таку задачу:

знайти

$$\max\{2(u'_{r_0} - u'_s) - \sum_{(ij)\in E} z_{ij}\} \tag{6.29}$$

при обмеженнях

$$u'_j - u'_i \le c_{ij} + z_{ij}, \quad (i,j) \in E, \tag{6.30}$$

$$z_{ij} \ge 0, \quad (i,j) \in E. \tag{6.31}$$

Ця задача є двоїстою для задачі знаходження потоку мінімальної вартості на мережі G, на якій джерелом є вузол s, стоком – вузол r_0, всі ребра мають одиничну пропускну здатність, вартість ребра e дорівнює c_e для $e \in E$, а задана величина потоку дорівнює 2.

Припустимо, що оптимальний розв'язок задачі про мінімальний потік отримано поліноміальними алгоритмами [162, 141] і що таким способом було знайдено оптимальне значення u'^0_v для $v \in V$ та z^0_{ij}, $(i,j) \in E$. Для зручності оптимальне значення z^0_{ij} позначимо z_{ij}.

На наступній ітерації фіксуємо новий сток r_1 в N_0, який відрізняється від r_0. Беремо

$$u_v = u'^0_v \quad \text{якщо} \quad u_v > u'^0_v,$$

$$c_{ij} := \max\{0,\ c_{ij} - u'^0_j + u'^0_i\} \text{ для } (i,j) \in E, \tag{6.32}$$

знову розв'язуємо задачу (6.29)–(6.31) з цими новими параметрами.

Повторюємо цю процедуру доти, поки в N_0 не залишиться фіксованих вузлів. Таким чином, розв'язуємо задачу про мінімальний потік $|N|-1$ разів. Відзначимо, що на кожній ітерації, якщо визначаємо величину w'_{ij}, як це було показано в (6.28), то згідно з (6.32) стверджуємо, що за допомогою цього алгоритму можна знайти величини w'_{ij}, для яких обмеження (6.26) задовольняються. Наприкінці визначаємо z_{ij} за формулою

$$z_{ij} = \sum_{k=1}^{|N_0|} z^k_{ij} .$$

Оскільки обмеження (6.26) задовольняються для w^r_{ij}, можна зробити висновок, що u^r_v, z^r_{ij}, w^r_{ij}, знайдені описаним вище алгоритмом, є допустимим розв'язком задачі (6.24)–(6.27), а значення (6.24), що відповідає цьому розв'язку, є нижньою границею для ЗПНЗМ. Описаний алгоритм знаходить точне оптимальне значення (6.24) для деяких класів задачі (6.24)–(6.27) (див., наприклад, три перші задачі з табл. 6.1, поданої нижче). У загальному випадку за допомогою цього алгоритму визначається наближений розв'язок задачі (6.24)–(6.27).

Для знаходження границі обчислень у цьому алгоритмі відзначимо деякі особливості, важливі для оцінки нижньої границі. Оскільки всі ребра мають одиничну пропускну здатність, а величина потоку дорівнює 2, то задача (6.29)–(6.31) є двоїстою до задачі знаходження двох шляхів між вузлами s та r_0, що не перетинаються та мають мінімальну сумарну вартість своїх ребер. Тому для знаходження $u^{r_0}_v$, $z^{r_0}_{ij}$, $w^{r_0}_{ij}$, для $v \in V$ та $(i,j) \in E$ можна скористатися алгоритмом доповнюючих шляхів мінімальної вартості (див. [74]), за допомогою якого ці шляхи можна знайти за $O(1)$ ітерацій. Таким чином, оскільки $c_{ij} \geq 0$, то один із них (скажімо, P_1) можна знайти алгоритмом Дейкстри.

Оскільки пропускна здатність ребер обмежена зверху одиницею, при застосуванні алгоритму найменших ланцюгів (див. [20]) на другій ітерації ребро (v,w) шляху P_1 замінюється на дві дуги (v,w) та (w,v) з вартостями $c_{wv} = M$ та $c_{vw} = -M$, де M – досить велике число. Тому можемо скористатися алгоритмом Форда для отримання другого шляху на мережі, що залишилася, а значення $u^{r_0}_v$ знайдемо як потенціал вершини $v \in V$.

З теорії задачі про потік мінімальної вартості добре відомо, що для деяких ребер (i,j) та потенціалів $u^{r_0}_i$ і $u^{r_0}_j$ виконується нерівність

$$u^{r_0}_j - u^{r_0}_i > c_{ij}.$$

З простих спостережень за алгоритмом найменших ланцюгів випливає, що ця нерівність буде виконуватись для ребер шляху P_1. Таким чином, лише для ребер шляху P_1 можна одержати, що $z^0_{ij} = c_{ij} - u^{r_0}_j + u^{r_0}_i$ та $z^0_{ij}=0$ для всіх інших ребер (i,j) з E. Оскільки два реберно-непересічних шляхи можна знайти за час $O(|V^3|)$, то описаним вище алгоритмом можна одержати наближений розв'язок задачі (6.24)–(6.27) за час $O(|V^3|\cdot|N|)$.

Зробимо кілька зауважень. В описаному вище алгоритмі вибір джерела серед термінальних вузлів та фіксування стоку серед вершин, що залишилися, проводилися в довільному порядку. Однак існують класи задач проектування зв'язних мереж, для яких доцільно діяти таким чином. За джерело вибираються вузли v або s таким чином, що найкоротший шлях, який їх з'єднує, дорівнює найдовшому з найкоротших шляхів між будь-якими парами вузлів з N. При цьому за допомогою алгоритма знаходиться краща нижня границя порівняно з іншими варіантами вибору джерела.

Це показує, що при фіксуванні джерела серед вузлів, що залишилися в N, важливо яким-небудь алгоритмом попереднього впорядкування розташувати найкоротші шляхи, які з'єднують джерело s з вузлами множини N, що залишилися, в порядку зменшення, й лише після цього фіксувати вузли в N_0, враховуючи це розташування.

6.2.5. Алгоритм для знаходження верхньої границі

В даному підрозділі описано евристичні алгоритми побудови початкової допустимої мережі та евристичні засоби локальних поліпшень для зменшення вартості спроектованої мережі при збереженні її допустимості.

Перший алгоритм базується на так званому "жадібному" підході для побудови 2-реберно-зв'язної мережі $\overline{G}$ та подальшому вилученні з неї окремих ребер таким чином, щоб ця мережа залишалася допустимою. Далі (якщо можливо) вартість мережі $\overline{G}$ зменшується за допомогою евристичного алгоритму локального пошуку.

Нехай E_0 – множина таких ребер (i,j), що обмеження (6.26) для апроксимації розв'язку двоїстої задачі (див. п. 6.2.3) задовольняється як рівність, а граф G_0 задається множиною ребер

E_0. Неважко показати, що граф $G_0 = (V_0, V_0)$ є 2-реберно-зв'язним. Опишемо алгоритм побудови такої 2-реберно-зв'язної мережі $\overline{G} = (\overline{V}, \overline{E})$, що граф $\overline{G}$, який її представляє, є підграфом графа G_0.

На початку роботи алгоритму вважаємо, що множини $\overline{V}$ та $\overline{E}$ є пустими. Далі на мережі G_0 алгоритмом Дейкстри знаходимо найкоротші шляхи між кожною парою термінальних вузлів. Нехай L – список, в якому зберігаються всі ці шляхи. З цього списку вибирається шлях мінімальної вартості. Позначимо його P_1, а вузли, які він з'єднує, – v_1 та v_2. Цей шлях видаляється із списку L, а всі ребра та всі вершини включаються до множин ребер $\overline{E}$ та вершин $\overline{V}$ відповідно. В цьому списку ми знову вибираємо шлях мінімальної вартості. Позначимо його P_2; нехай він з'єднує вузли v_3 та v_4. Якщо степені вузлів v_3 та v_4 (по відношенню до поточного графа $\overline{G} = (\overline{V}, \overline{E})$) більші чи дорівнюють 2, видаляємо цей шлях із списку L, а вершини та ребра цього шляху не включаються до $\overline{V}$ та $\overline{E}$. В протилежному випадку включаємо вершини та ребра шляху P_2 відповідно до $\overline{V}$ та $\overline{E}$. Виконання процедури продовжується, поки список L не стане пустим. Нехай $\overline{G}$ – граф, побудований таким чином. Оскільки $d_f \geq 2$ для всіх вершин f з $\overline{V} \backslash N$, то для перевірки, чи є цей граф 2-реберно-зв'язним, спочатку вибирається вузол v з N степені $d_v = 1$ (якщо такий існує). Припустимо, що шлях $P(v, w)$ є мінімальним порівняно з найкоротшими шляхами між вузлом v та рештою вершин з $\overline{V}$ на графі, що задається множиною ребер $E_0 \backslash \overline{E}$. Далі до множин $\overline{V}$ та $\overline{E}$ включаємо всі вершини та ребра шляху $P(v, w)$, в результаті чого одержуємо новий граф $\overline{G} = (\overline{V}, \overline{E})$). Надалі ця процедура виконується не більше $O(|N|)$ разів.

Для впевненості застосовуємо алгоритм Гоморі та Ху для знаходження мінімальних розрізів, що роз'єднують будь-яку пару вузлів з N на графі $\overline{G}$ з одиничними пропускними здатностями всіх його ребер. Якщо мінімальний розріз, що розділяє вузли v_1 та v_2, менший за 1, то знайдемо найкоротший шлях між v_1 та v_2 на графі, який визначається множиною ребер $E_0 \backslash \overline{E}$,

і включимо всі ребра та вершини цього шляху відповідно до множин $\overline{V}$ та $\overline{E}$.

Однак може статися, що граф $\overline{G}$ містить такі ребра, що при їх видаленні з $\overline{G}$ одержимо граф (також будемо позначати його $\overline{G}$), який залишиться 2-реберно-зв'язним.

Нехай d_{v_1}, d_{v_2} – степені вершин v_1 та v_2, такі, що $d_{v_1} > 2$, $d_{v_2} > 2$. Тоді на графі $\overline{G}$ знаходимо найкоротший шлях $P_1(v_1, v_2)$ між вершинами v_1 та v_2, а далі назначаємо всім ребрам шляху $P_1 = P_1(v_1, v_2)$ досить великі вартості й знову знаходимо найкоротший шлях $P_2 = P_1(v_1, v_2)$ між вершинами v_1 та v_2. Припустимо, що ми знайшли реберно-непересічні шляхи $P_1, P_2, ..., P_k$ між вершинами v_1 та v_2, де $k \geq \min\{d_{v_1}, d_{v_2}\}$. У списку $P_1, P_2, ..., P_k$ зберігаються лише ті шляхи, вершини яких належать $\overline{V} \backslash N$ за виключенням вершин v_1 та v_2. Якщо $k > 2$, то всі ребра шляху P_k видаляються з $\overline{G}$.

Далі встановлюємо $k = k - 1$, і якщо $k > 2$, то всі ребра шляху P_k знову видаляються з $\overline{G}$. Ця процедура повторюється, поки k залишається більшим за 2. Таким способом знаходимо допустимий розв'язок $\overline{G}$ ЗПНЗМ, і на цьому опис першого алгоритму завершується.

Наступним кроком з метою знаходження 2-реберно-зв'язаного підграфа графа $\overline{G}$ (розв'язок) мінімальної вартості застосуємо евристичну процедуру покращення.

Степенева евристична процедура є локальним пошуком покращення для вершини, степінь якої більше 4. Припустимо, що степінь d_v вершини v більша 4. Тоді в графі $\overline{G}$ існує принаймні 4 ребра, що виходять з вершини v, скажімо, (v, v_1), (v, v_2), (v, v_3), (v, v_4) . Якщо

$$c_{vv_1} + c_{vv_2} + c_{vv_3} + c_{vv_4} > \min\{c_{vv_1} + c_{vv_2} + c_{v_3v_4},\quad c_{vv_3} + c_{vv_4} + c_{v_1v_2}\},$$

то можемо видалити з графа $\overline{G}$ або пару ребер (v, v_1), (v, v_2), або ж пару (v, v_3), (v, v_4), і додати в граф $\overline{G}$ або ребро (v_1, v_2), або ребро (v_2, v_3), для того щоб одержати розв'язок з меншою вартістю, ніж вартість графа $\overline{G}$.

Евристична процедура для циклу дає локальне поліпшення

для циклів, довжина яких (тобто кількість ребер циклу) більша за 4. При цьому робиться спроба замінити два несуміжних ребра (v_1, v_2) та (v_3, v_4) з циклу C на два ребра (v_1, v_4) та (v_2, v_3), щоб одержати цикл C'.

Розглянемо цикл C як простий граф. Якщо ребра (v_1, v_2) та (v_3, v_4) видалені з циклу C, то він розділяється на два шляхи P_1 та P_2, що зв'язують v_1 з v_3 та v_2 з v_4, відповідно. Якщо тепер з'єднати v_1 з v_4, а v_2 з v_3, то з циклу C утвориться новий цикл C'. Евристична процедура такого типу була використана для покращення знайденого допустимого розв'язку в задачі проектування стійкої мережі з обмеженням на її 2-зв'язність [138].

Для того щоб скористатися евристичними процедурами локального пошуку для циклів за степенями, виберемо будь-яку вершину v, для якої $d_v > 4$, і цикл C, що має довжину, більшу за 4, та одержимо новий допустимий розв'язок нижчої вартості за допомогою описаних вище перетворень графа $\overline{G}$. Ця процедура повторюється, поки подальші покращення можливі.

Крім евристичних процедур вказаного типу, можна було б застосувати до графа $\overline{G}$ інші цікаві евристичні процедури локального поліпшення, описані в [138]. Однак час роботи цих евристичних алгоритмів локального пошуку пропорційний часу виконання одного кроку в алгоритмі гілок та границь, так що наперед незрозуміло, чи буде знайдено допустимий розв'язок нижчої вартості цими евристичними алгоритмами локального пошуку.

Опишемо тепер другий евристичний алгоритм для побудови допустимого розв'язку ЗПНЗМ. Цей алгоритм можна розглядати як комбінацію якого-небудь з евристичних алгоритмів для побудови дерева Штейнера $T(N)$ з алгоритмом знаходження найкоротшого шляху. Спершу одним з евристичних алгоритмів (див. [138, 150]) знаходимо дерево Штейнера $T(N)$, яке охоплює всі термінальні вузли в N. Нехай степені вершин $v_1, ..., v_k$ дерева $T(N)$ дорівнюють 1. Ясно, що всі вершини $v_1, ..., v_k$ лежать в N. Тоді, як і в першому алгоритмі, знаходимо найко-

ротший шлях, що з'єднує вершини v_i, $i = 1, ..., k$ з іншими вершинами дерева $T(N)$. Таким чином, будуємо 2-реберно-зв'язний граф $\overline{G} = (\overline{V}, \overline{E})$, а далі, якщо це можливо, зменшуємо вартість мережі $\overline{G}$ евристичними алгоритмами локального покращення для степенів та циклів.

Другий евристичний алгоритм цікавий тим, що він дозволяє одержати ряд результатів, що зв'язують оптимальне дерево Штейнера з оптимальним розв'язком ЗПНЗМ. Припустимо, що в другому евристичному алгоритмі дерево $T(N)$ визначено за допомогою алгоритму, описаного в праці [132] . Тоді маємо (див. [132])

$$c(T(N)) \le 2(1 - 1/l)c(T_*),$$

де $c(T(N))$ – повна вартість дерева $T(N)$; l – кількість листків на дереві T_*. Припустимо, що P_k – множина ребер шляху, що з'єднує лист v_i, $i = 1, ..., k$, з іншою вершиною дерева $T(N)$, а $c(P_k)$ – сумарна вартість ребер P_k. Тоді маємо $c(\overline{G}) \le$ $\le c(T(N) + c(P_k))$, де $c(\overline{G})$ – вартість $\overline{G}$. Таким чином, одержуємо

$$c(\overline{G}) \le 2(1 - 1/l)c(T_*) + c(P_k).$$

Нехай G_* – оптимальний розв'язок ЗПНЗМ. Ясно, що за допомогою вилучення деякої множини ребер E_* з графа G_* одержимо дерево Шнейнера T_0. Тоді можна записати

$$c(G_*) = c(T_0) + c(E_*) \ge c(T_*) + c(E_*),$$

звідки випливає, що

$$c(G_1) \le 2(1 - 1/l)c(G_*) + c(P_k) - c(E_*).$$

А з цієї нерівності можна зробити висновок, що розв'язок ЗПНЗМ можна одержати як розв'язок задачі: знайти таке дерево Штейнера T, на якому досягається мінімум $c(P_k)$ при обмеженнях $c(T) \le c(T(N))$, де $c(T(N))$ обчислено раніше поліноміальним алгоритмом з [132].

6.2.6. Результати обчислень

Для оцінювання запропонованих евристичних підходів було розв'язано кілька задач – три з них є практичними зада-

чами проектування мережі, а сім було сгенеровано випадковим чином. Для розв'язання тестових задач використовувалась модель (6.12)–(6.15). У всіх випадках для знаходження розв'язку ЗПНЗМ було застосовано схему алгоритму гілок та границь. Для обчислення нижньої границі F_{lower} було використано апроксимаційний алгоритм з п. 6.2.4, а для обчислення верхньої границі F_{upper} – евристичний алгоритм з п. 6.2.5. Ці алгоритми були запрограмовані на C++.

Спочатку знаходимо наближений розв'язок задачі (6.24)–(6.27) та обчислюємо F^0_{lower}.

Нехай граф G_0 задається множиною ребер (i, j), для яких обмеження (6.26) виконуються як рівності. Очевидно, що реберна зв'язність графа G_0 не менша за 2. Знаходимо допустимий розв'язок $\overline{G}$ на графі G_0 та відповідне значення (6.12), яке дорівнює F_{upper}. Якщо для заданого ε виконується нерівність

$$F_{upper} - F_{lower} \le \varepsilon F^0_{lower},$$

то обчислення завершуються. В цьому випадку ми знайшли ε-оптимальний розв'язок. У протилежному випадку фіксуємо $x_e = 0$ чи $x_e = 1$ для ребра e з графа $\overline{G}$ (поточний розв'язок), вартість якого є найбільшою серед нефіксованих ребер цього графа, і виконуємо стандартну ітерацію методу гілок та границь. Після обчислення F_{lower} та F_{upper} перевіряється нерівність (6.33) для поточного значення верхньої границі.

Результати обчислень для ε = 0,05 наведено в табл. 6.1. Граф, що являє собою мережу в задачах 1, 2, 3, є графом доріг для трьох областей України, а термінальним вузлам відповідають великі міста. В інших задачах мережа задавалася повним графом, а ваги ребер прирівнювалися випадковим числам від 0 до 100.

В табл. 6.1 використано такі позначення:

n – кількість вершин мережі;

m – кількість ребер мережі;

node – кількість вузлів (терміналів) мережі;

F_{lower} – нижня оцінка, знайдена на першому кроці алгоритму;

Таблиця 6.1. Результати обчислень для ε = 0,05

N	n	m	*node*	F_{lower}	F_{upper}	F_r	nF_r	Час (с.)
1	8	12	4	20	20	20	0	0,1
2	17	29	5	40	44	44	0	0,2
3	38	65	8	552	553	553	0	0,2
4	10	45	10	140	190	170	5	1,1
5	15	105	10	113	118	116	1	9,2
6	15	105	15	122	296	182	4	14,8
7	20	190	10	54	112	104	5	647,8
8	30	435	10	60	100	95	3	1841,9
9	40	780	10	32	63	50	3	429,8
10	50	1225	10	30	55	52	2	124,9

F_{upper} – верхня оцінка, знайдена на першому кроці алгоритму;

F_r – знайдене рекордне значення для цільової функції;

nF_r – кількість змін рекордного значення цільової функції (6.12) при роботі алгоритму гілок та границь.

Всі наведені тут результати обчислень було одержано на ПК-486 (30 Мгц). Хоча алгоритм, описаний в п. 6.2.4., знаходить досить "слабке" значення початкової нижньої границі для випадково сгенерованих задач (6.4)–(6.10), порівняно незначний час отримання розв'язку пояснюється тим, що коли ми приписали досить великі вартості шести-десяти ребрам графа G_0 (поточний розв'язок ЗПНЗМ), значення нижньої границі різко зросло і стало більшим за значення поточного рекорду для (6.12). Приписування досить великої вартості ребру e графа G_0 є еквівалентним ситуації, коли покладається, що значення відповідної змінної x_e на поточній ітерації алгоритму гілок та границь дорівнює нулю. Для того щоб зафіксувати

$x_e = 0$, вибиралося ребро графа G_0, вартість якого максимальна порівняно з іншими ребрами G_0.

В наступних підрозділах буде досліджуватися існування розв'язку ЗПНЗМ для окремих типів графа H.

6.3. Окремі випадки ЗПНЗМ, коли граф є деревом або паросполученням

У даному підрозділі розглядається ЗПНЗМ для випадку, коли всі ребра підграфа, ізоморфного заданому, можуть бути видаленими або зруйнованими. Нехай задано граф $H = (V_h, E_h)$, V_h – множина його вершин, E_h – множина його ребер. Розглянемо задачу проектування зв'язної мережі для випадку, коли всі ребра підграфа, ізоморфного H, можуть бути видаленими з початкового графа, тобто потрібно знайти підграф G_* графа G мінімальної вартості, що містить хоча б один шлях для кожної пари вузлів з N навіть після видалення з G_* ребер підграфа, ізоморфного H. Цю задачу будемо також скорочено позначати ЗПНЗМ.

Тут розглянемо кілька окремих випадків ЗПНЗМ для спеціальних типів графа $H = (V_h, E_h)$, обговоримо алгоритми розв'язання та доведемо ряд теорем про існування розв'язків цих задач.

1. Випадок, коли граф H є паросполученням. Нехай граф H має $2p$ вершин та p ребер, тобто він є паросполученням з p ребрами. Спробуємо розібратися, для яких графів G ЗПНЗМ має розв'язок.

Відомо, що максимальна кількість p_{sr} шляхів, що з'єднують вершини s та r у графі G та не мають спільних вершин, дорівнює мінімальній кількості вершин, необхідних для того, щоб граф G став незв'язним. Тому можемо знайти p_{sr}, розв'язавши задачу про максимальний потік та задачу про мінімальний розріз на орграфі G', побудованому таким чином. Вводимо в графі G копії v' всіх вершин $v \in V$, крім джерела s та стоку r. Вершини v', v з'єднуємо ребром (v', v) одиничної пропускної здатності. Всі ребра (w, v) замінюються на пари ребер (w, v')

та (v, w') з досить великими пропускними здатностями. Відомо, що потужність розрізу з мінімальним числом елементів на G', що розділяє джерело s та сток r, дорівнює максимальному числу шляхів p_{sr}, що з'єднують вершини s та r та не мають спільних вершин. За допомогою алгоритму Гоморі та Ху число p_{sr} можна знайти для будь-якої пари вузлів $s, r \in N$. Таким чином, якщо

$$\min\{p_{sr} : s, r \in N\} \geq p + 1, \qquad (6.34)$$

то ЗПНЗМ має розв'язок на графі G.

Якщо $|N| = 2$ ($N = \{s, r\}$), то ЗПНЗМ можна розв'язати одним із поліноміальних алгоритмів [54, 162, 141], за допомогою яких знаходиться потік мінімальної вартості.

2. Випадок, коли граф H є деревом. Нехай H – дерево з q вершинами, а граф G, пропускні здатності ребер якого дорівнюють 1 для всіх $e \in E$, позначимо $G(1)$. Для будь-якого $W \neq 0$, $W \subset V$ множина ребер

$$\delta(W) = \{(i, j);\ \ i \in W,\ \ j \in V \setminus W\} \qquad (6.35)$$

становить розріз, що відділяє вершини з W та $\overline{W} = V/W$.

Нехай $R(N)$ – клас розрізів $\delta(W)$, для яких множини $W \cap V$ та $\overline{W} \cap V$ не є пустими. Тут також можемо знайти розріз $\delta_{\min}(W)$ мінімальної потужності $R(N)$ в шляхом визначення максимальних потоків між всіма парами вузлів з N алгоритмом Гоморі та Ху.

Теорема 6.2. *Якщо потужність розрізу $\delta_{\min}(W)$ є більшою за $q - 1$, то ЗПНЗМ має розв'язок.*

Доведення. Припустимо, що розріз $\delta_{\min}(W)$ розділяє деякі вершини $s, r \in N$. З теорії потоків на мережах випливає, що будь-який потік з s до r, в тому числі й максимальний, можна розкласти на суму потоків уздовж простих шляхів та уздовж циклів. В той же час величина максимального потоку не залежить від величини потоків уздовж циклів. Тому можна вважати, що величини потоків уздовж циклів дорівнюють нулю. Для всіх ребер графа $G(1)$, які мають одиничні пропускні здатності, можемо розкласти максимальний потік на потоки

вздовж простих шляхів, що не мають спільних ребер. Ясно, що кількість таких шляхів дорівнює потужності розрізу $\delta_{min}(W)$. З формулювання теореми випливає, що потужність розрізу $\delta_{min}(W)$ є більшим за кількість ребер у дереві H. Тому після вилучення всіх ребер якого-небудь дерева, ізоморфного H, джерело та сток можна буде з'єднати принаймні одним шляхом,що й треба було довести.

У випадку графа H, розглянутого в даному підрозділі, задачу існування розв'язку в ЗПНЗМ можна розв'язати за час $O(|V^3|\cdot|N|)$ строго поліноміальним алгоритмом Гоморі та Ху.

6.4. Окремі випадки ЗПНЗМ, коли граф H є лісом або паросполученням

В даному підрозділі буде розглянуто задачу ПЗМ для двох типів графів H, коли не існує поліноміального алгоритму для з'ясування питання, чи існує розв'язок в ЗПНЗМ, якщо $P \neq NP$.

1. Випадок, коли граф H є лісом. Нехай $|N| = 2$, $N = \{s, r\}$ і H є лісом з піддерев з двома висячими вершинами, тобто кожне дерево є простим ланцюгом. У цьому випадку проблема, чи існує розв'язок в ЗПНЗМ, є еквівалентною такій грі Шеннона для двох гравців на графі [66]. Обидва гравця за домовленістю виділяють дві вершини (вузли s та r) як термінальні. Далі вони по черзі роблять ходи.

Згідно з правилами один гравець у свій хід видаляє одне з ребер графа й намагається розбити всі ланцюги, що з'єднують вершини s та r. Другий гравець за свій хід може відмітити одне з ребер, і таке відмічене ребро вже не можна вилучити з графа. Його мета – зберегти хоча б один ланцюг, що з'єднує вершини s та r. Гра, в якій може виграти другий гравець, називається грою Шеннона 2-го типу.

Теорема 6.3. *Задача перевірки, чи існує розв'язок ЗПНЗМ, коли* $|N| = 2$, $N = \{s, r\}$, *а* H *є лісом, що складається з ланцюгів, еквівалентна грі Шеннона 2-го типу.*

Д о в е д е н н я. Для доведення теореми покажемо, що степені вершин підграфа G_{first}, що містить ребра, вилучені першим гравцем, не більше 2. У [66] доведено, що гра Шеннона належить 2-му типові тоді й тільки тоді, коли граф G складається з двох дерев, що не мають спільних ребер з однаковими вершинами та має у своєму складі обидві вершини s і r. Відповідно до цього факту хід другого гравця полягає в тому, щоб відмітити ребро, яке зв'язує два компоненти одного з дерев, що утворюються в результаті видалення ребра першим гравцем. Оскільки обидва дерева мають однакове число ребер, другий гравець завжди зможе з'єднати дві частини, що утворилися з одного дерева, ребром, яке належить іншому дереву.

Припустимо, що степінь деякої вершини v підграфа G_{first} не менше, ніж 3. Тоді підграф G_{first} містить хоча б 3 ребра (v, w_1), (v, w_2), (v, w_3). Нехай ці ребра в цьому ж порядку вилучені першим гравцем. Позначимо T_v, T_{w_1} (де $v \in T_v$, $w \in T_{w_1}$) нові компоненти дерева, що утворилися після видалення (v, w_1). З видаленням ребра (v, w_2) піддерева T_v воно розпадається на два піддерева, скажімо, T_{vv}, T_{vw_2}, де $v \in T_{vv}$, $w_2 \in T_{vw_2}$. Нарешті, після видалення ребра (v, w_3) піддерева воно розпадається на два піддерева, скажімо, T_{vvv} і T_{vvw_3}, де $v \in T_{vvv}$, $w_3 \in T_{vvw_3}$. Таким чином, у результаті видалення першим гравцем за три ходи ребер (v, w_1), (v, w_2), (v, w_3), утворяться такі піддерева:

$$T_{w_1}, T_{vw_2}, T_{vvv}, T_{vvw_3}.$$

Розглянемо всі можливі випадки. Нехай $s \in T_{w_1}$, $r \in T_{vvw_3}$. Якщо перший гравець видаляє ребро (v, w_2), це буде марним ходом, бо згідно з вказаною стратегією другий гравець може за два ходи з'єднати T_{vvv} з ребром T_{vvw_3} другого піддерева. Аналогічно показується, що видалення першим гравцем одного з ребер (v, w_i), $i = 1, 2, 3$ є зайвим ходом в інших випадках.

У випадку, коли s і $r \in$ вершинами одного піддерева, те ж саме показується тривіально. Таким чином, у графі G_{first} не існує вершини v степеня більше, ніж 2. У загальному випадку

граф G_{first} може бути незв'язним. Отже, граф G_{first} є лісом з ланцюгів, що і було потрібно довести.

Покажемо, що задача про два дерева: чи містить даний граф два дерева, що не мають спільних ребер, а всі вершини яких однакові, є *NP*-повною.

Розглянемо задачу про два ланцюги: чи містить даний граф два ланцюги, що не мають спільних ребер, усі вершини яких однакові. Коли число вершин ланцюгів оцінюється як $O(|V|)$, ця задача еквівалентна *NP*-повній задачі про Гамільтонов шлях [19], і тому вона теж *NP*-повна.

Зрозуміло, що задача про два ланцюги є окремим випадком задачі про два дерева. Тому задача про два дерева є *NP*-повною.

2. Випадок, коли граф H є паросполученням. Нехай $|N| \geq 2$, а граф $H = (V_h, E_h)$ є паросполученням, тобто E_h складається з ребер, що попарно не мають спільних кінцевих вершин. Цей випадок відрізняється від попереднього тим, що тут кількість ребер паросполучення невідоме.

Теорема 6.4. *Якщо граф G містить деякий планарний підграф, для якого всі внутрішні грані – трикутники, а множина вершин має у своєму складі N, то існує рішення ЗПНЗМ на графі G для довільного паросполучення H, $N \subseteq V$.*

Доведення. Нехай граф G містить зв'язний планарний підграф G_p з множиною вершин $V(G_p) \subseteq N$. Покажемо, що підграф G_p залишиться зв'язним після видалення з нього ребер будь-якого паросполучення, що міститься в G_p. Припустимо, що в підграфі G_p існує паросполучення M, після видалення ребер якого підграф G_p розпадається на k компонент зв'язності $G_{p(i)}$ з множинами вершин $V(G_{p(i)})$ і множинами ребер $E(G_{p(i)})$ для $i = 1, ..., k$.

Нехай $n_i = |V(G_{p(i)})|$, $m_i = |E(G_{p(i)})|$ для $i = 1, ..., k$, $q = |M|$. Тепер розглянемо кожний підграф $G_{p(i)}$ як окремий граф і позначимо f_i число його граней. Оскільки ребра, вилучені з підграфа G_p, є ребрами паросполучення M, то з кожної грані підграфа G_p може бути вилучене тільки одне ребро.

Нехай $e = (v, w)$ – таке ребро паросполучення M, що $v \in V(G_{p(i_1)})$, $w \in V(G_{p(i_2)})$ для деяких $1 \leq i_1, i_2 \leq k$. Ясно, що цим ребром можна з'єднати графи $G_{p(i_1)}$ та $G_{p(i_2)}$, і в результаті такого з'єднання одержимо новий граф з множиною вершин $V(G_{p(i_1)}) \cup V(G_{p(i_2)})$ і множиною ребер $E(G_{p(i_1)}) \cup E(G_{p(i_2)}) \cup \{e\}$ з числом граней $f_{i_1} + f_{i_2} - 1$. Таким чином, шляхом додавання ребра з паросполучення M до цього графу ми одержимо сам підграф G_p з числом граней $f_1 + f_1 + ... + f_k + q - k - 1$. Звідси повинна виконуватися рівність

$$f_1 + f_1 + ... + f_k + q - (k - 1) = f,$$

де f – число граней підграфа G_p. Зрозуміло, що для розбиття $V(G_{p(1)})$, ..., $V(G_{p(k)})$ множини $V(G_p)$ має місце рівняння

$$m_1 + m_2 + ... + m_k + k = m.$$

За формулою Ейлера $m_i - f_i = n_i - 2$, $m - f = n - 2$, де n, m – кількості вершин та ребер подграфа G_p. Віднімаючи з останнього рівняння попереднє, одержуємо

$$n_1 + n_2 + ... + n_k - 2k + k - 1 = n - 2.$$

Беручи до уваги, що $n_1 + n_2 + ... + n_k = n$, отримуємо, що $k = 1$. Це означає, що підграф G_p не може розпадатися на $k \geq 2$ частин після вилучення з нього ребер довільного паросполучення, що й потрібно було довести.

Якщо вартості $c_e = 1$ для всіх ребер графа G, то ЗПНЗМ у цьому випадку еквівалентна знаходженню такого підграфа G_* у графі G з мінімальним числом ребер, множина вершин якого містить N, а всі його грані є трикутниками.

Наприклад, нехай всі грані підграфа G_*, що складається з двох ланцюгів з однаковими вершинами і без спільних ребер – трикутники. У цьому випадку, при $N = V$, знаходження підграфа G_* еквівалентно перевірці, чи містить граф G Гамільтонів шлях. Отже, задача знаходження планарного підграфа графа G, для якого всі внутрішні грані – трикутники, а множина вершин містить N, є *NP*-повною.

Підсумуємо результати, подані в розділі 6. Теореми 6.2, 6.3, 6.4 про існування розв'язку в задачі побудови зв'язної мережі мінімальної вартості доведені для випадку, коли оптимальна мережа залишається зв'язною навіть коли всі ребра графа, ізоморфного заданому, видалено з нього. Ці теореми не тільки задають достатні умови існування розв'язку в задачі побудови зв'язної мережі мінімальної вартості, але є також корисними для розробки евристичних алгоритмів для розв'язання ЗПНЗМ у випадках, розглянутих у підрозділах 6.3 та 6.4. Для розв'язання ЗПНЗМ у випадку, коли H є паросполученням і $|N| \geq 2$, ми можемо знайти 2-реберно-зв'язний граф Г алгоритмами, описаними в підрозділі 6.2.5, коли ми додаємо ребра до Г таким чином, щоб будь-який розріз $\delta(W)$ на Г не створював паросполучення. Якщо Г є циклом, то ми повинні додавати вершини до Г таким чином, щоб всі підцикли одержаного підграфа були трикутниками за виключенням одного, який може складатися з 4 вершин.

Розглянуті моделі мають широке коло застосувань і можуть бути використаними при формулюванні та аналізі генетичної задачі (див. [66]). Зокрема, ці результати для ЗПНЗМ на двочастинних графах можна застосовувати в теорії проектування надійних схем.

Розділ 7. Програмне забезпечення для розв'язування задач оптимального проектування надійних мереж

У даному розділі подано стислий опис програмного забезпечення, розробленого для розв'язування розглянутих задач оптимального проектування надійних мереж. Оскільки технічні характеристики цих задач ставлять різні вимоги до інтерфейсу користувача, це змусило створити три програмних блоки з окремими інтерфейсами. Перший блок об'єднує програмні модулі для розв'язання задач таких типів.

1. *Задача знаходження пропускних здатностей дуг надійної орієнтованої мережі при одиничних відмовах її елементів у випадку передачі потоків за довільними шляхами.* Зокрема, для задач цього типу передбачено два формулювання: одне для випадку проектування нової мережі з суворими вимогами до її надійності (модель 1), а друге – для випадку, коли потрібно спроектувати нові пропускні здатності, що доповнюють вже існуючі, щоб мережа стала надійною (модель 2).

2. *Задача знаходження пропускних здатностей дуг надійної орієнтованої мережі при одиничних відмовах її елементів у випадку передачі потоків по заданій множині допустимих шляхів.* Це дозволяє покращити надійність мережі чи знайти критичні вузли.

3. *Задача оптимізації мереж з урахуванням неповної інформації.*

Другий програмний блок було спеціально розроблено для задач проектування оптимальної логічної структури мережі. Він включає розвинуті засоби для роботи з картою, необхідні для розв'язування практичних задач цього типу.

Третій програмний блок було розроблено для задач перс-

пективного планування перевезень та раціонального розподілу коштів на реконструкцію транспортних мереж (залізничних, автомобільних, авіаційних). Він бере до уваги потреби користувача при довгостроковому плануванні перевезень та при знаходженні оптимальної номенклатури рухомого складу. Всі ці блоки мають зручну систему меню та довідку (help).

7.1. Система для розв'язання задач 1–3

В систему із спільним інтерфейсом включено оптимізаційні модулі для розв'язування таких задач проектування надійних мереж.

1. Модель 1 – проектування мережі мінімальної вартості за умови виходу з ладу окремих ланок (програму написано мовою С, див. розд. 6); модель 2 – знаходження оптимальних пропускних здатностей дуг надійної орієнтованої мережі з передачею потоків за довільними шляхами (програму написано мовою ФОРТРАН, див. п. 4.1.1).

2. Знаходження оптимальних пропускних здатностей дуг надійної орієнтованої мережі з передачею потоків по заданій множині допустимих шляхів (програму написано мовою ФОРТРАН, див. п. 4.1.2).

3. Задача оптимізації мереж з урахуванням неповної інформації (програму написано мовою С++, див. підрозд. 4.4). Цей модуль дозволяє розв'язувати мережеві оптимізаційні задачі зі стохастичними параметрами у вигляді двоетапних моделей з фіксованою рекурсією, в яких стохастичними є праві частини (вимоги). Передбачається, що вимоги є випадковими величинами зі скінченним дискретним розподілом, які задані у вигляді таблиці реалізацій та їх ймовірностей. Включена до системи програма є реалізацією декомпозиційної схеми по змінних, яка використовує r-алгоритм для розв'язання координуючої задачі.

Система складається з бази даних, сукупності оптимізаційних модулів і керуючої підсистеми. Керуюча підсистема містить засоби, що підтримують роботу з багатьма варіантами

моделей для кожної задачі. Підсистема реалізована на основі СУБД INTERBASE засобами системи швидкої розробки програм С++BUILDER. Одночасно з системою можуть працювати кілька користувачів (при наявності ліцензії на відповідний режим роботи із СУБД INTERBASE), можливості розподілу доступу до даних не використовуються, усі користувачі мають рівні права доступу. Кожен користувач повинен працювати з окремим варіантом моделі.

Засобами СУБД підтримується цілісність бази: видалення зв'язаних записів у дочірніх таблицях, якщо видаляється деякий запис із батьківської таблиці; блокування додавання записів у дочірню таблицю, якщо відсутній відповідний запис у батьківській таблиці. Більш глибокий аналіз коректності даних проводиться при роботі прикладних модулів при розв'язуванні конкретної задачі. Усі зміни даних у базі мають проводитися засобами цієї системи. У протилежному випадку можливі порушення цілісності інформації.

Користувач спілкується із системою тільки через керуючу підсистему, що забезпечує інтерфейс користувача за допомогою меню та сукупності екранних форм, формування в автоматичному режимі вхідних файлів виконуваних модулів для вказаних типів задач, запуск заданого користувачем програмного модуля, завантаження в базу даних результатів розрахунків.

Інсталяція системи. Для роботи з системою на комп'ютері повинні бути встановлені:

С++BUILDER 6 Professional;

INTERBASE 7.0 Server (при наявності ліцензії на режим роботи з багатьма користувачами сервер може бути встановлений на іншому комп'ютері локальної мережі);

INTERBASE 7.0 Client.

Система поставляється у вигляді двох файлів:

Reliable_network.rar – архівний файл, що включає частину кліента у складі: керуюча підсистема, програмні модулі по кожному типу задач, система допомоги (Help),

Reliable_network_Gdb.rar – архівний файл, що включає файл бази даних Reliable_network.gdb.

Файл Reliable_network.rar має таку структуру підкаталогів:

- кореневий каталог;
- підкаталог MODEL A1;
- підкаталог MODEL A2;
- підкаталог MODEL C;
- підкаталог MODEL D.

Файл Reliable_network.rar слід розархівувати із збереженням структури підкаталогів у будь-який каталог на жорскому диску. Файл Reliable_network.gdb також слід розмістити на жорсткому диску. Для цього файла засобами BDE Administrator-а слід створити аліас RELIABLE_NETWORK з параметрами: User Name=SYSDBA, Password = masterkey.

Після цього система готова до роботи. Запуск системи проводиться шляхом запуску модуля Net_project.exe.

7.2. Система для розв'язання задачі проектування оптимальної логічної структури мережі

Математичне формулювання задачі знаходження оптимальної логічної структури мережі докладно описано в розд. 4.2.

Розроблені програмні модулі реалізовані на базі сучасних програмних засобів C++ BUILDER і працюють під Windows 2000/XP. Комплекс програм є багатовіконним та призначеним для користувача, що не є програмістом. Програмні модулі мають інтуїтивно зрозумілий інтерфейс. Керування ходом виконання програмних модулів здійснюється за допомогою розвиненої системи меню та маніпулятора "миші". У довільний момент роботи користувач має можливість одержати контекстно-орієнтовану довідкову інформацію.

Для відображення даних за базові беруться карти України та Європи. Прив'язка спеціалізованих об'єктів до карти може виконуватись або автоматизовано (якщо для них є паперова карта або відомі їх географічні координати), або вручну, нанесенням об'єкта на карту "на око". Довільний об'єкт у процесі

роботи може бути перенесений на нове місце, а в лінійних та площинних об'єктів може бути також змінена форма.

Користувач за допомогою клацання лівою кнопкою "миші" на зображенні об'єкта може одержати всю інформацію, що його цікавить, яка пов'язана з цим об'єктом та знаходиться в основній базі даних, а при бажанні – змінити її.

Розроблено засоби для знаходження найкоротших шляхів у заданій мережі. Існує також можливість ввести пошкодження ребра або в одному, або в обох напрямках. Це дозволяє в оперативному режимі обчислювати необхідну корекцію інформаційних потоків.

Розроблено програми, що спрощують роботу з географічною картою. Зокрема, вони дають можливість зчитувати необхідні географічні координати з баз даних або дисплея; позиціювати карту згідно з цими координатами; змінювати масштаб, щоб необхідні об'єкти можна було побачити на дисплеї; програмно зображати на дисплеї необхідну інформацію згідно з розробленими сценаріями.

Для визначення координат географічних об'єктів при побудові нової структури комутаційної мережі розроблено програмні засоби, які дозволяють:

- одержувати інформацію про створені об'єкти (географічні координати, вид, тощо);
- обчислювати необхідні показники цих об'єктів (довжину, площу, тощо);
- записувати одержану інформацію в бази даних.

Після розв'язування задачі треба відтворити отримані результати на дисплеї. Для цього треба створити нові графічні об'єкти – вузли побудованої комутаційної мережі та лінії зв'язку, здійснити їх прив'язку до географічної карти та відобразити їх на дисплеї. Для виконання цих задач розроблені програмні засоби, які дозволяють:

- створювати графічні об'єкти (точкові, лінійні, площинні);
- прив'язувати побудовані об'єкти в необхідних місцях до географічної карти;

- редагувати положення графічних об'єктів на географічній карті;
- записувати інформацію про графічні об'єкти в бази даних.

7.3. Система для розв'язання задачі перспективного планування вантажних перевезень та розвитку транспортної системи

Програмне забезпечення призначене для визначення оптимальної схеми вантажних перевезень і раціонального розподілу капіталовкладень на реконструкцію станцій і поповнення вагонопарку транспортної мережі.

Вхідні дані визначають інформацію про такі об'єкти:

- транспортна мережа (граф залізниць, станції, ділянки);
- плановий період (кількість інтервалів планового періоду, їх тривалість);
- рухомий склад (чисельність, функціональні й вартісні характеристики вантажних вагонів);
- види агрегованих перевезень (множина різних типів вантажів);
- термінали вантажних операцій (станції формування складів та їх потужності по обробці вантажів);
- прогнозований обсяг та структура вантажних перевезень (кореспонденцій) по планових інтервалах;
- обсяг капітальних фінансових ресурсів, призначених для реконструкції елементів транспортної системи (нарощування потужностей терміналів і ділянок);
- правила і схеми визначення плати та експлуатаційних витрат при реалізації вантажних перевезень.

Вихідні дані програмного забезпечення визначають числову інформацію з головних питань розглянутого класу задач перспективного планування:

- критичні по пропускній здатності елементи залізничної мережі і рекомендації їх раціонального розвитку (обсяги фінансових ресурсів, необхідних для проведення їх реконструкції);

- раціональна структура парку вантажних вагонів та обсяги фінансових ресурсів, виділених на його поповнення;
- реалізація замовлень на перевезення (реалізація кореспонденцій);
- раціональна структура парку вантажних вагонів та обсяги фінансових ресурсів, виділених на його поповнення;
- реалізація замовлень на перевезення (реалізація кореспонденцій);
- раціональна схема вагонопотоків на мережі залізниць, що забезпечує реалізацію замовлень на перевезення.

Ці дані можуть бути використані для прийняття рішень перспективного розвитку транспортної системи й об'єктивного обґрунтування критичних факторів її функціонування. Схема вагонопотоків забезпечує важливу інформацію для розв'язання задачі визначення маршрутів вантажних потягів та складання розкладу руху.

Критерій оптимізації полягає в максимізації загального доходу, одержаного в результаті реалізації вантажних перевезень у плановому періоді (значення цільової функції дорівнює обсягові фінансових ресурсів, одержаних за виконання перевезень кореспонденцій за винятком експлуатаційних витрат по їх реалізації).

Основні обмеження задачі:

- умови балансу вагонопотоків;
- балансові співвідношення обсягів реалізації кореспонденцій;
- умови узгодження вагонопотоків з реалізацією перевезень кореспонденцій;
- умови обмеженості потужностей вузлів;
- умови обмеженості робочого парку рухомого складу;
- умова обмеженості засобів на реконструкцію терміналів мережі та рухомого складу.

Як базові елементи розроблені методи розв'язання задач використовують такі алгоритми [13, 35, 36, 45)]:

- алгоритм побудови найкоротших шляхів на графі;

- алгоритм розв'язання транспортної задачі на графі;
- алгоритм негладкої оптимізації в схемах декомпозиції блокових задач математичного програмування.

Програмне забезпечення математичної моделі розроблено мовою C++ у стилі об'єктно-орієнтованого програмування [2]. Розроблені класи відображають концепції об'єктів математичної моделі. Інтерфейс користувача обслуговування бази даних розроблений у середовищі C++Builder.

Програмне забезпечення включає:

- спеціалізовану базу даних інформаційного забезпечення задач (СУБД InterBase);
- модулі алгоритмів розв'язання задачі;
- сервісні засоби діалогового режиму постановки, розв'язання й аналізу результатів;
- сервісні засоби графічного зображення результатів розв'язання задачі.

Програмне забезпечення призначене для розв'язання широкого спектра задач розглянутого класу, має розвинутий інтерфейс для розгляду різних варіантів моделі та проведення аналізу розв'язку.

Список літератури

1. *Ахо А., Хопкрофт Дж., Ульман Дж.* Построение и анализ вычислительных алгоритмов. – М.: Мир, 1979. – 535 с.
2. *Буч Г.* Объектно-ориентированный анализ и проектирование с примерами приложений на C++. – М.: Бином, 1999. – 564 с.
3. *Демьянов В.Ф., Васильев Л.В.* Недифференцируемая оптимизация. – М.: Наука, 1981. – 384 с.
4. *Дикин И.И.* Итеративное решение задач линейного и квадратичного программирования // ДАН СССР. – 1967. – 174. – С. 747–748.
5. *Дикин И.И. Зоркальцев В.И.* Итеративное решение задач математического программирования. – Новосибирск: Наука, 1980. – 127 с.
6. *Ермольев Ю.М.* Методы стохастического программирования. – М.: Наука, 1976. – 239 с.
7. *Журбенко Н.Г.* Об одном классе методов минимизации с преобразованием пространства // Методы решения экстремальных задач. – К.: Ин-т кибернетики им. В.М. Глушкова НАН Украины, 1996. 68–80.
8. *Журбенко Н.Г.* Оценка эффективности одного класса ε-субградиентных методов минимизации с преобразованием пространства // Оптимизация и ее приложения. – К.: Ин-т кибернетики им. В.М. Глушкова НАН Украины, 1997. – С. 49–54.
9. *Журбенко Н.Г.* Про один ε-субградієнтний алгоритм мінімізації // Теорія оптимальних рішень. – К.: Ін-т кібернетики ім. В.М. Глушкова НАН України, 2002. – С. 111–118.
10. *Зоркальцев В.И.* Метод наименьших квадратов. – Новосибирск : Наука, 1995. – 220 с.
11. *Кнут Д.* Искусство программирования на ЭВМ. Т.3. Сортировка и поиск. – М.:Мир, 1978. – 848 с.

12. *Крамер Г.* Математические методы статистики.–М.: Мир, 1975. – 648 с.
13. *Кристофидес Н.* Теория графов. Алгоритмический подход. – М. : Мир, 1978. – 432 с.
14. *Лаптин Ю.П., Журбенко Н.Г.* Разработка программных средств оптимизации сложных технических объектов // Теорія оптимальних рішень. – К.: Ін-т кібернетики ім. В.М.Глушкова НАН України, 2002. – С. 3–12.
15. *Лаптин Ю.П.* Декомпозиция по переменным для некоторых задач оптимизации // Кибернетика и системный анализ. – 2004. – № 1. – С. 98–104.
16. *Лаптин Ю.П.* ε-Субградиенты в методах декомпозиции по переменным для некоторых задач оптимизации // Теорія оптимальних рішень. – К.: Ін-т кібернетики ім. В.М. Глушкова НАН України, – 2003. – № 2. – С. 75–82.
17. *Лебедева Т.Т., Сергиенко И.В.* Метод направляющих окрестностей // Докл. АН УССР. Сер.А. – 1978. – № 6. – С. 241–244.
18. *Михалевич В.С., Трубин В.А., Шор Н.З.* Оптимизационные задачи производственно-транспортного планирования. Модели, методы, алгоритмы. – М.: Наука, 1986. – 260 с.
19. *Пападимитриу Х., Стайглиц К.* Комбинаторная оптимизация. – М.: Мир, 1985. (Перев. с англ. *Papadimitriou C.H., Steiglitz K.* Combinatorial optimization: algorithms and complexity. – New Jersey: Prentice-Hall, 1982.)
20. *Результаты* экспериментального исследования эффективности методов, включенных в пакет прикладных программ ДИСПРО / В.С. Михалевич, И.В. Сергиенко, А.И. Кукса, В.А. Рощин, В.А Трубин. – Киев, 1980. – 68 с. – (Препр. / АН УССР. Ин-т кибернетики; 80–47).
21. *Ржевський С.В.* Монотонні методи опуклого програмування. – К.: Наук. думка, 1993. – 316 с.
22. *Сергиенко И.В.* Вопросы разработки одного подхода к решению задач оптимизации в системах обработки данных и в автоматизированных системах управления // Упр. системы и машины. – 1974. – № 6. – С.107–115.
24. *Сергиенко И.В., Шило В.П., Стецюк П.И.* Приближенный алгоритм решения задачи нахождения максимального независимого множества // Компьютерная математика. – К.: Ин-т кибернетики им. В.М. Глушкова НАН Украины, 2000. – С. 4–20.
25. *Сергієнко И.В.* Один метод розв'язування задач на відшукання екстремальних значень // Автоматика. – 1964. – № 5. – С. 15–21.
26. *Сергієнко І.В.* Інформатика в Україні: становлення, розвиток, проблеми. – К.: Наук. думка, 1999. – 354 с.
27. *Сергиенко И.В.* Математические модели и методы решения задач дискретной оптимизации. – К: Наук. думка, 1988. – 471 с.
28. *Сергиенко И.В., Шило В.П., Боярчук Д.А.* Задача о надежности коммуникационной сети при отказах ее компонент // Теория оптимальных решений. – К.: Ин-т кибернетики им. В.М. Глушкова НАН Украины, 2001. – С. 54–61.
29. *Сергиенко И.В., Шило В.П., Боярчук Д.А.* Нахождение надежной логической структуры коммуникационной сети, минимизирующей эксплуатационные затраты // Комп'ютерна математика. Оптимізація обчислень. – К.: Ін-т кібернетики ім. В.М. Глушкова НАН України, – 2001. – № 2. – С. 368–376.
30. *Стецюк П.И.* Линейная модель для нахождения пропускных способностей компонент надежной сети // Там же. – 2001. – № 1. – С. 376–384.
31. *Стецюк П.И.* Приближенный метод эллипсоидов // Кибернетика и системный анализ. – 2003. – № 3. – С. 141–146.
32. *Танаев В.С.* Декомпозиция и агрегирование в задачах математического программирования. – Минск: Наука и техника, 1987. – 183 с.
33. *Трубин В.А.* Прочность графа и упаковка деревьев и ветвлений // Кибернетика и системный анализ. – 1993. – № 3. – С. 94–99.

34. *Трубин В.А., Шарифов Ф.А.* Общая задача размещения // Теория и вычислительные проблемы оптимизации. – К.: Ин-т кибернетики им. В.М. Глушкова НАН Украины, 1993. – С. 16–20.
35. *Филлипс Д., Гарсиа-Диас А.* Методы анализа сетей. – М.: Мир, 1984. – 496 с.
36. *Ху Т.* Целочисленное программирование и потоки в сетях. – М.: Мир, 1974. – 519 с.
37. *Цурков В.И.* Декомпозиция в задачах большой размерности. – М.: Наука, 1981. – 351 с.
38. *Шарифов Ф.А.* Задача синтеза сети с независимыми источниками // Оптимизация и её приложения: Сб. научных трудов. – К.: Ин-т кибернетики им. В.М. Глушкова НАН Украины, 1997. – С. 26–31.
39. *Шарифов Ф.А.* Задача синтеза надежных сетей // Кибернетика и системный анализ. – 2000. – № 4. – С. 145–156.
40. *Шарифов Ф.А.* Задача синтеза надежных сетей с многими источниками и стоками // Теория оптимальных решений. – К.: Ин-т кибернетики им. В.М. Глушкова НАН Украины, 2001. – С. 16–22.
41. *Шарифов Ф.А. и др.* Задача синтеза интегрированных цифровых сетей // Методы решения экстремальных задач. – К.: Ин-т кибернетики им. В.М. Глушкова НАН Украины, 1996. – С. 10–14.
42. *Шило В.П.* Метод глобального равновесного поиска // Кибернетика и системный анализ. – 1999. – № 1. – С. 74–81.
43. *Шило В.П.* Результаты экспериментального исследования эффективности метода глобального равновесного поиска // Там же. – № 2. – С. 93–102.
44. *Шор Н.З.* Метод отсечения с растяжением пространства в задачах выпуклого программирования // Кибернетика. – 1977. – № 1. – С. 94–95. (English transl.: Cybernetics. – 1977. – 13. – P. 94–96).
45. *Шор Н.З.* Методы минимизации недифференцируемых функций и их приложения. – К.: Наук. думка, – 1979. –

200с. (English transl.: Shor N.Z. Minimization Methods for Non-Differentiable Functions. – Berlin: Springer-Verlag, 1985. – 178 p.)
46. *Шор Н.З.* Монотонные модификации *r*-алгоритмов и их приложения // Кибернетика и системный анализ. – 2002. – №6. – С. 74–96.
47. *Шор Н.З., Стеценко С.И.* Квадратичные экстремальные задачи и недифференцируемая оптимизация. – К.: Наук. думка, 1989. – 208 с.
48. *Шор Н.З., Шабашова Л.П.* О решении минимаксных задач методом обобщенного градиентного спуска с растяжением пространства // Кибернетика. – 1972. – № 1. – С. 82–88.
49. *Юдин Д.Б., Гольштейн Е.М.* Линейное программирование. Теория, методы и приложения. – М.: Наука, 1969. – 424 с.
50. *Юдин Д.Б., Немировский А.С.* Информационная сложность и эффективные методы решения выпуклых экстремальных задач // Экономика и мат. методы. – 1976. – № 12. – С. 357–369. (English transl.: Matekon. – 1977. – 13 (3). – P. 25–45).
51. *Aarts E. H. L., Lenstra J.K.* Local Search in Combinatorial Optimization. – Chichester: J. Wiley and Sons, 1997.
52. *Aarts E., Korst J.* Simulated annealing and Boltzmann machines. – New York: J. Wiley and Sons, 1989.
53. *Ahuja R.K., Magnanti T.L., Orlin J.B.* Network flows: Theory, algorithms, and applications. – Prentice-Hall, Upper Saddle River, New Jersey, 1993.
54. *Ahuja R.K., Thomas L., Orlin J.B.* Some resent advances in network flows // SIAM Review. – 1991. – **33**, N 2. – P. 175–219.
55. *Arnott R., de Palma A., Lindsey R.* Does providing information to drivers reduce traffic congestion? // Trans. Res. – 1991. – Part A, 25. – P. 309–318.
56. *Barahona F.* Network design using cut inequalities // SIAM.J.Optimization. – 1996. – **6**, N 3. – P. 823–837.
57. *Barahona F., Grötschel M., Mahjoub A.* Facets of the bipartite

subgraph polytope // Math. and Operations Research. – 1985. – N 10. – P. 340–358.
58. *Barahona F., Mahjoub A.* On the cut polytope // Math. Programming. – 1986. – **36**. – P. 157–173.
59. *Barnes E.R.* A variation on Karmarkar's algorithm for solving linear programming problems // Ibid. – P. 174–182.
60. *Benders J.F.* Partitioning procedures for solving mixed variables programming problems // Numer. Math. – 1962. – **4**, N 3. – P. 237–260.
61. *Bern M.W.* Faster exact algorithms for Steiner trees in planar networks // Networks. – 1990. – **20**. – P. 109–120.
62. *Birge J.R., Dempster M.A.H., Gassmann H.I., Gunn E., King A.J., Wallace S.W.* A standard input format for multiperiod stochastic linear programs. // Working Paper WP-87-118, IIASA. – Laxenburg, Austria, 1987.
63. *Bock F.* An algorithm for solving 'traveling-salesman' and related network optimization problems. Manuscript associated with talk presented at the Fourteenth National Meeting of the Oper. Res. Society of America, St Louis, Missouri // Abstract in Bulletin Fourteenth National Meeting of the Operations Research Society of America. – 1958. – P. 897.
64. *Boese K. D., Kahng A. B., Muddu S.* A new adaptive multistart technique for combinatorial global optimisations // Oper. Res. Lett. – 1994. – **16**. – P. 101–113.
65. *Bonomi E., Lutton J.L.* The *n*-sity travelling salesman problem: statistical mechanics and the Metropolis algorithm // SIAM Rev. – 1984. – **26**, N 4. – P. 551–568.
66. *Busaker R.G., Saaty T.L.* Finite graphs and networks. – New York: McGraw-Hill, 1965.
67. *Cerny V.* Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm // J. Optimiz. Theory and Appl. – 1985. – **45**. – P. 41–51.
68. *Chandra B., Karloff H., Tovey C.* A. New results on the old fc-opt algorithm for the traveling salesman problem // SIAM J. on Comput. – 1999. – **28**. – P. 1998–2029.

69. *Chao I.M., Golden B., Wasil E.* An improved heuristic for the period vehicle routing problem // Networks. – 1995. – **26**. – P. 25–44.
70. *Christofides N.*, Whitlock C.A. Network synthesis with connectivity constraints: a survey // Operational Research. – 1981. – **81**. – P. 705–723.
71. *Croes G. A.* A method for solving traveling-salesman problems // Oper. Res. – 1958. – **6**. – P. 791–812.
72. *Chu P.C., Beasley J.E.* A genetic algorithm for the multidimensional knapsack problem. – 1997. – 237 p.
73. *Clegg R.J., Clune A.J.* The MUSIC Project: Urban Control for Traffic Demand Management. To appear in the Transportation Research Record (available at *http://gridlock.york.ac.uk*).
74. *Cook W., Cunningham W.H., Pulleyblank W.R., Schrijver A.* Combinatorial Optimization. – New York: John Wiley & Sons, Inc., 1998.
75. *Cunningham W.H.* Optimal attack and reinforcement of network // J. ACM. – 1985. – **32**. – P. 549–561.
76. *Dorigo M., Gambardella L. M.* Ant colony system: a cooperative learning approach to the traveling salesman problem // IEEE Trans. Evolut. Comput. – 1997. – **1(1)**. – P. 53–66 (available at http://iridia.ulb.ac.be/~mdorigo/ACO/ACO.html).
77. *Dorigo M., Maniezzo V., Colomi A.* The ant system: optimization by a colony of cooperating agents // IEEE Trans. Sys., Man, Cybernetics. – 1996. – **26(1)**. – P. 29–41 (available at http://iridia.ulb.ac.be/~mdorigo/ACO/ACO.html).
78. *Dreyfus S.E., Wagner R.A.* The Steiner problem in graphs // Networks. – 1972. – **1**. – P. 195–207.
79. *Dueck G., Scheuer T.* Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing // J. Comput. Phys. – 1990. – **90**. – P. 161–175.
80. *Ericsson M., Resende M., Pardalos P.* A genetic algorithm for the wieght setting problem in OSPF routing // J. of Global Opt. – 2002. – **6**, N 3. – P. 299–333.

81. *Ermoliev Yu.M., Shor N.Z.* A Random Search Method for Two-Stage Problems of Stochastic Programming and its Generalization // Cybernetics. – 1968. – N 1. – P. 90–92.
82. *Ferreira A.G., Zerovnik J.* Bounding the probability of success of stochastic methods for global optimization // Comput. Math. Appl. – 1993. – **25**. – P. 1–8.
83. *Ferris M.C., Ruszczynski* A. Robust Path Choise in Networks with Failures // Networks. – 2000. – **35(3)**. – P. 181–194.
84. *Floudas C.A., Pardalos P.M.* (Eds.) Encyclopedia of optimization. 6 Volumes. – Dordrecht; Boston; London: Kluwer Academic Publishers, 2001.
85. *Fortz B., Labbe M.* Two Connected Networks with Rings of Bounded Cardinality. // Comput. Optimiz. and Applic. – 2004, – **27**. – P. 123–148.
86. *Fortz B., Labbe M.* Polyhedral results for two-connected networks with bounded rings // Math. Prog. Ser.A. – 2002. – **93**. – P. 27–54.
87. *Fortz B.* Design of Survivable Networks with Bounded Rings. –. Dordrecht; Boston; London: Kluwer Academic Publishers, 2000.
88. *Fourer R., Gay D.M., Kernighan B.W.* AMPL: A Modeling Language for Mathematical Programming. – Duxbury Press / Brooks / Cole Publishing Company, 1993. – 351 p.
89. *Frauendorfer K.* Stochastic two-stage programming // Lecture Notes in Economics and Mathematical Systems. – Berlin: Springer, 1992. – Vol. 8392.
90. *Glover F.* A template for scatter search and path relinking // Artificial Evolution, Lecture Notes in Computer Science, 1363 / Ed. by J.K. Hao, E. Lutton, E. Ronald, M. Schoenauer, D. Snyers. – Berlin: Springer-Verlag, 1998. – P.13–54.
91. *Glover F.* Future paths for integer programming and links to artificial intelligence // Comput. Oper. Res. – 1986. – **13**. – P. 533–549.
92. *Glover F.* Tabu Search. Part I // ORSA J. on Comput. – 1989. – N 1. – P. 190–206.
93. *Glover F.* Tabu Search. Part II // Ibid. – N 2. – P. 4–32.

94. *Glover F.* Tabu search and adaptive memory programming – advances, applications, and challenges // Interfaces in Computer Science and Operations Research: Advances in Metaheuristics, Optimization and Stochastic Modeling Techniques / Ed. by R. Barr, R. Helgason, J. Kennington. – Boston: Kluwer Academic Publishers, 1997. – P. 1–75.
95. *Glover F., Lokketangen A., Woodruff D.* Scatter search to generate diverse MIP solutions // Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research / Ed. by M. Laguna and J. L. Gonzalez-Velarde. – Boston: Kluwer Academic Publishers, 2000. – P. 299–317.
96. *Glover F., Laguna M.* Tabu Search. – Boston: Kluwer Academic Publishers, 1997.
97. *Glover F., Laguna M., Marti R.* Scatter search // Forthcoming in Theory and Applications of Evolutionary Computation: Recent Trends / Ed. by A. Ghosh and S. Tsutsui. – Berlin: Springer-Verlag, 2000 (available at http: // bus.colorado.edu /faculty/laguna).
98. *Goemans M.X., Bertsimas D.J.* Survivable networks, linear programming relaxations and the parsimonious property // Math. Prog. – 1993. – **60**. – P. 145–166.
99. *Gondzio J.* HOPDM (version 2.12) – A fast LP solver based on a primal-dual interior point method // Eur. J. Oper. Res. – 1995. – **85**. – P. 221–225.
100. *Gondzio J., Vial J.-P.* Warm Start and -Subgradients in a Cutting Plane Scheme for Block-Angular Linear Programs // Comp. Optimization and Appl. – 1999. – **14**. – P. 17–36.
101. *Grötschel M., Lyvasz L., Schrijver* A. Geometric Algotithms and Combinatorial Optimization. Ser. Algorithms and Combinatorics; 2. – Berlin: Springer-Verlag, 1988. – 364 p.
102. *Grötschel M., Monma C.L.* Integer polyhedra arising from certain network design problems with connectivity constraints // SIAM J. Disc. Math. – 1990. – N 4. – P. 502–522.

103. *Grötschel M., Monma C.L., Stoer M.* Design of Survivable Networks // Network Models. Ser. Handbooks in Operations Research and Management Science. – Amsterdam, North-Holland: Elsevier. – 1995. – Vol. 7. – Ch. 10. – P. 617–672.
104. *Hakimi S.L.* Steiner's problem in graphs and its implications // Networks. – 1970. – N 1. – P. 113–133.
105. *Hasegava S., Okanoue Y., Egava T., Sakauchi H.* Control Algorithms of SONET Integrated Self-Healing Networks // IEEE J. On Selected Areas in Communications. – 1994. – **12**, N 1. – P. 110–118.
106. *Higle J.L., Sen S.* Stochastic decomposition. A statistical method for large scale stochastic linear programming. – Dordrecht; Boston; London: Kluwer Academic Publishers, 1996.
107. *Holland J.* Adaptation in Natural and Artificial Systems. – Ann Arbor: University of Michigan Press, 1975.
108. *Holmberg K., Yuan D.* A Multicommodity Network-Flow Problem with Side Constraints on Paths Solved by Column Generation // Journal on Computing. – 2003. – **15**. – P. 42–57.
109. *Holmberg K.* Experiments with Primal-Dual Decomposition and Subgradient Methods for the Uncapacitated Facility Location Problem // Optimization. – 2001. – **49**. – P. 495–516.
110. *Holmberg K., Yuan D.* A Lagrangean Heuristic Based Branch-and-Bound Approach for the Capacitated Network Design Problem // Operations Research. – 2000. – **48**. – P. 461–481.
111. *Holmberg K., Yuan D.* A Lagrangean Approach to Network Design Problem // Int. Trans. Opt. Res. – 1998. – **5**, N 6. – P. 529–539.
112. *Holmberg K.* Efficient decomposition and linearisation methods for the stochastic transportation problem // Computational Optimisation and Applications. – 1995. – N 4. – P. 293–316.

113. *Hu T.C.* Integer programming and network flows. — California: Addison Wesley Publishing Company, 1970.
114. *Hwang F.K., Richards D.S.* Steiner tree problem // Networks. – 1992. – **22**. – P. 55–89.
115. *Johnson D. S.* Local optimization and the traveling salesman problem // Proc. of the 17th Colloquium on Automata, Languages and Programming. – 1990. – P. 446–461.
116. *Johnson D. S., McGeoch L.* A. The traveling salesman problem: a case study // Local Search in Combinatorial Optimization / Ed. by E.H.L. Aarts and J.K. Lenstra. – Chichester: John Wiley and Sons, 1997. – P. 215–310.
117. *Johnson D. S., Papadimitriou C. H., Yannakakis M.* How easy is local search? // Comput. Sys. Sci. – 1988. – **37**. – P. 79–100.
118. *Kall P., Wallace S.W.* Stochastic programming. – Chichester: Wiley, 1994.
119. *Karmarkar N.* A new polynomial time algorithm for linear programming // Combinatorica. – 1984. – **4**, N 4. – P. 373–395.
120. *Kerivin H., Mahjoub A.R.* Separation of partition inequalities for the (1,2)-survivable network design problem // Op. Res. Letters. – 2002. – **30**. – P. 265–268.
121. *Kernighan B. W., Lin S.* An efficient heuristic procedure for partitioning graphs // Bell Sys. Tech. – 1970. – **49**. – P. 291–307.
122. *Kirkpatrick S., Gelatti C.D., Vecchi M.P.* Optimization by simulated annealing // Science. – 1983. – **220**. – P. 671–680.
123. *Larsson T., Patriksson M.* An augmented Lagrangean dual algorithm for link capacity side constrained traffic assignment problems. // Trans. Res. – 1995. – Part B, **29**. – N 6. – P. 433–455.
124. *Larsson T., Yuan D.* An augmented Lagrangean Algorithm for Large Scale Multicommodity Routing // Computational Optimization and Application. – 2004. – **27**. – P. 187–215.
125. *Lawer E.L., Lenstra J.K., Rinnooy Kan A.H.G., Shmoys D., et al.* The traveling salesman problem. – New York: John Wiley & Sons Inc., 1985.

126. *Lemarechal C., Mifflin K.* Nonsmooth Optimization. – Oxford: Pergamon Press, 1978. – 180 p.
127. *Lin S.* Computer solutions of the traveling salesman problem // Bell Sys. Tech. J. – 1965. – **44**. – P. 2245–2269.
128. *Lin S., Kernighan B. W.* An effective heuristic algorithm for the traveling salesman problem // Oper. Res. – 1973. – **21**. – P. 498–516.
129. *Magnanti T., Mirchandani P., Vachani R.* The convex hull of two core capacitated network design problems // Math.Prog. – 1993. – **60**. – P. 233–250.
130. *Mahjoub A.R.* Two-edge connected spanning subgraphs and polyhedra // Math. Prog. – 1994. – **64**. – P. 199–208.
131. *Mayer J.* Stochastic Linear Programming Algorithms: A Comparison Based on a Model Manegement System. – Amsterdam: ODP, 1998. – 153 p.
132. *Mehlhorn K.* A faster approximation algorithm for the Steiner problem in graphs // Information Processing Letters. – 1988. – **27**. – P. 125–128.
133. *Migdalas A.* Bilevel programming in traffic planning: Models, methods and challenge // J. of Global Optimisation. – 1995. – N 7. – P. 381–405.
134. *Miller-Hooks E.* Adaptive least-expected time paths in stochastic, time-varying transportation and data networks // Networks. – 2001. – **37**, N 1.– P. 35–52.
135. *Minoux M.* Discrete Cost Multicommodity Network Optimization Problems and Exact Solution Methods // Annals of Operation Research. – 2001. – **106**. – P. 19–46.
136. *Minoux M.* Network Synthesis and Optimum Design Problems: Model, Solution Methods and Applications // Networks. – 1989. – **19**. – P. 313–360.
137. *Mladenovic N., Hansen P.* Variable neighborhood search // Comput. Oper. Res. – 1997. – **24(11)**. – P. 1097–1100.
138. *Monma C.L., Shallcross D.F.* Methods for designing communications networks with certain two-connected survivability constraints // Oper. Res. – 1989. – **3**, N 4. – P. 531–541.

139. *Optimization* by simulated annealing: an experimental evaluation. Part I / D.S. Johnson, C.R. Aragon, L.A. McGeoch and C. Schevon // Graph partitioning. Oper. Res. – 1989. – **37**. – P. 865–892.
140. *Optimization* by simulated annealing: an experimental evaluation. Part II / D.S. Johnson, C.R. Aragon, L.A. McGeoch and C. Schevon // Graph coloring and number partitioning. Oper. Res. – 1991. – **39**. – P. 378–406.
141. *Orlin J.B.* A faster strongly polynomial minimum cost flow algorithm // Oper. Res. – 1993. – **41**. – P. 338–350.
142. *Osman I., Kelly J.* Meta-heuristics: an overview // Meta-heuristics: Theory and Applications / Ed. by I. Osman and J. Kelly.-Boston: Kluwer Academic Publishers, 1996. – P. 1–21.
143. *Osman I., Laporte G. (*Eds.*)* Metaheuristics in Combinatorial Optimization, Annals of Operations Research (Volume 63). – The Netherlands: Baltzer. 1996.
144. *Prekopa A.* Stochastic programming. – Boston; Dordrecht; London: Kluwer Academic Publishers, 1995.
145. *Reeves C. (Ed.)* Modern Heuristic Techniques for Combinatorial Optimization. – New York: Halsted Press, 1993.
146. *Reiter S., Sherman G.* Discrete optimizing //. Soc. Ind. Appl. Math. – 1965. – **13**. – P. 864–889.
147. *Ruszczynski* A. Decomposition methods in stochastic programming // Math. Prog. – 1997. – **79**. – P. 333–353.
148. *Selman B., Kautz H.*A. Domain-independent extensions to GSAT: solving large structured satisfiability problems // Proc. of the 13th International Joint Conference on Artificial Intelligence. – 1993. – P. 290–295.
149. *Sen S., Doverspike R.D., Cosares S.* Network planning with random demand // Telecommunication systems. – 1994. – N 3. – P. 11–30.
150. *Shaohan M.* The Steiner trees on graph and its heuristic algorithm // Chin. J. Comput. – 1985. – N 8. – P. 237–239.

151. *Sharifov F.* Network design problems under edges of isomorphic subgraph are deleted // Proc. International Network Optimization Conference. Evry – Paris, France, 2003. – P. 521–525.
152. *Sheffi Y.* Urban transportation networks. – Englewood Cliffs, New Jersey: Prentice-Hall, 1985.
153. *Shor N.Z.* Nondifferentiable Optimization and Polynomial Problems. – Boston; Dordrecht; London: Kluwer Academic Publishers, 1998. – 412 p.
154. *Smith M.J.* A local traffic control policy which autimatically maximises the overall travel capasity of an urban road network // Traffic Engineering and Control. – **21**. – P. 298–302.
155. *Stoer M.* Design of survivable Networks. / (Ser. Lecture Notes in Mathematics, Vol. 1531). – Berlin: Springer-Verlag. 1992.
156. *Vanderbei R.J.* LOQO: An interior point code for quadratic programming // Tech. Rep. SOR-94-15, School of Engineering and Applied Science, Department of Civil Engineering and Operations Research, Princeton University, 1994.
157. *Väbrand P., Göthe Lundgren M., Jörnsten K.* On the nucleus of the basic vehicle routing game // Math. Prog. – 1996. – 72. – P. 83–100.
158. *Värbrand P., Tuy H., Ghannadan S., Migdalas A.* A strongly polynomial algorithm for a concave production-transportation problem with a fixed number of non-linear variables // Ibid. – P. 229–258.
159. *Värbrand P., Tuy H., Ghannadan S., Migdalas* A. Strongly polynomial algorithms for two special minimum concave cost network flow problems // Optimisation. – 1995. – **32**. – P. 23–43.
160. *Värbrand P., Tuy H., Ghannadan S., Migdalas A.* The minimum concave cost network flow problems with fixed number of sources and non-linear arc costs // J. of Global Optimisation. – 1995. – N 6. – P. 135–151.

161. *Voudouris C., Tsang E.* Partial constraint satisfaction problems and guided local search // Proc. of the Second International Conference on the Practical Application of Constraint Technology. – 1996. – P. 337–356.
162. *Vygen J.* On dual minimum cost flow algorithms // Math. Methods. Oper. Res. – 2002. – 56. – P. 101–126.
163. *Wets R.J.-B.* Challenges in Stochastic programming // Math. Prog. – 1993. – 75. – P. 115–135.
164. *Wets R.J.-B.* Large scale linear programming techniques in stochastic programming // Numerical techniques for stochastic optimization / Ed. by Ermoliev Yu.M. and Wets R.J.-B. – Berlin: Springer-Verlag, 1988. – P. 65–93.
165. *White K., Farber M., Pulleyblank W.* Steiner trees, Connected Domination and Strongly Chordal Graphs // Networks. – 1985. – **15**. – P. 109–124.
166. *Winter P.* Steiner problem in networks: A survey // Networks – 1987. – 170. – P. 129–167.
167. *Wynants C.* Network Synthesis Problems. – Boston; Dordrecht; London: Kluwer Academic Publishers, 2000.
168. *Yagiura M. T., Ibaraki, Glover F.* An ejection chain approach for the generalized assignment problem // Technical Report N 99013. Department of Applied Mathematics and Physics, Graduate School of Informatics. – Kyoto, Japan: Kyoto University, 1999.
169. *Yagiura M., Ibaraki T.* Analyses on the 2 and 3-flip neighborhoods for the MAX SAT // Combinatorial Optimiz. – 1999. – **3**. – P. 95–114.
170. *Yannakakiś M.* Computational complexity // Local Search in Combinatorial Optimization / Ed. by. E.H.L. Aarts and J.K. Lenstra. – Chichester, England: J. Wiley and Sons, 1997. – P. 19–55.
171. *Yuan D.* An Annotated Bibliorgaphy in Communication Network Design and Routing // Optimization Models and Methods for Communication Design and Routing, PhD thesis. – Linkoeping, Sweden: Linkoeping University, 2001.

Зміст

НАУКОВЕ ВИДАННЯ

Автори:

Н.З. Шор, І.В. Сергієнко, В.П. Шило, П.І. Стецюк,

І.М. Парасюк, Т.Т. Лебедева, Ю.П. Лаптін,

М.Г. Журбенко, Т.О. Бардадим, Ф.А. Шаріфов,

О.П. Лиховид, О.А. Березовський, В.М. Мірошниченко

Задачі оптимального проектування надійних мереж

Худ. оформлення *Т.С. Шеховцова*

Комп'ютерна верстка *Ю.Е. Дуріцький*

Підписано до друку 21.11.2005 р. Формат $60\times90/_{16}$.

Папір офс., 70 г/м^2. Гарн. літ. Друк офс.

Ум.-друк. арк. 14,08. Наклад 200 прим. Зам. № 2111-05.

Видавництво "Наукова думка"

Р. с. № 05417561 від 16.03.95

01601 Київ 1, вул. Терещенківська, 3

Оригінал-макет та друк:

Видавнича компанія "КИТ" 04080, м. Київ, вул. Фрунзе, 19–21,
тел.: 417-21-72, 462-48-51, 417-53-70

Свідоцтво про внесення до Державного реєстру суб'єктів видавничої справи ДК № 861 від 20.03.2002 р.

У книзі подано опис математичного і програмного забезпечення для ряду задач оптимального проектування та маршрутизації в мережах із врахуванням можливого виходу з ладу окремих компонент мережі та зміни вимог до потоків. Зокрема, розглянуто такі важливі з точки зору практичних потреб задачі на мережах: проектування мережі мінімальної вартості за умови виходу з ладу окремих ланок, знаходження пропускних здатностей дуг надійної орієнтованої мережі, проектування оптимальної логічної структури надійної мережі, модернізація надійної мережі, оптимізація мереж із врахуванням неповної інформації, перспективне планування перевезень, знаходження оптимальної номенклатури рухомого складу. Описано методи недиференційованої оптимізації, методи локального дискретного пошуку та схеми декомпозиції, використані для оптимізації надійних мереж.