МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
УКРАЇНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ НАУКИ І ТЕХНОЛОГІЙ

М. В. МИХАЙЛОВСЬКИЙ, О. П. ЄГОРОВ

ДОСЛІДЖЕННЯ ОПЕРАЦІЙ

Затверджено Вченою радою УДУНТ як навчальний
посібник Протокол № 1 від 26.09.2022

Дніпро 2022

УДК 004 : 65.011.8 (075.8)

Михайловський М. В., Єгоров О. П. Дослідження операцій : навчальний посібник. – Дніпро : Україн. держ. ун-т науки і технол., 2022. – 80 с.

Посібник призначений для студентів спеціальності 151 – Автоматизація та комп’ютерно-інтегровані технології.

Послідовно викладені основи дослідження операцій – науки, яка ставить своєю метою оптимізацію рішень, що приймаються в системах організаційно-технічного управління. Для постановки задачі оптимізації розглянуті етапи операційного проєкту, критерії оптимальності, види математичних моделей дослідження операцій та класи операційних задач.

Велика увага приділена викладенню сучасного математичного апарату дослідження операцій – методам лінійного, дискретного і нелінійного програмування. Опис теоретичних методів ілюструється численними практичними прикладами.

Іл.: 23. Бібліогр.: 5 найм.

Друкується за авторською редакцією.

Відповідальний за випуск М. В. Михайловський, канд. техн. наук, доц.

Рецензенти: В. М. Куваєв, д-р техн. наук, професор, головний науковий співробітник НТУ «Дніпровська політехніка» (м. Дніпро)
В. С. Ткачов, канд. техн. наук, доцент, доцент кафедри автоматизації та електротехніки ДВНЗ «Придніпровська державна академія будівництва та архітектури» (м. Дніпро)

ЗМІСТ

ВСТУП

Дослідження операцій є невід'ємною частиною системного аналізу, пов'язаною з прийняттям рішень. Сутність дослідження операцій полягає у пошуку найбільш ефективних варіантів досягнення мети шляхом комплексного аналізу проблемної ситуації, розробкою можливих рішень та оцінки наслідків їх реалізації.

Дослідження операцій – це наука, яка займається розробкою та практичним використанням методів ефективного (оптимального) управління. Предмет дослідження операцій – організаційні системи, що складаються з великої кількості взаємодіючих підрозділів, інтереси яких не завжди узгоджуються (і навіть суперечливі).

Метою дослідження операцій є кількісне обґрунтуванням рішень щодо управління організаціями. Рішення, яке виявляється найбільш корисним для всієї організації, називається оптимальним, а рішення, яке корисне лише одному або кільком підрозділам, є субоптимальним (покращуючим).

Таким чином, вивчення операцій є наука про кількісне обґрунтування оптимальних рішень.

Дослідження операцій базується на наступних поняттях:

1. Операція (захід) – сукупність дій, спрямованих на досягнення мети.

2. Мета операції – кількісний критерій ефективності прийнятих рішень.

2. Операційна сторона – особа або група осіб, які переслідують певну ціль. У складних операціях операційна сторона складається з ЛПР – людини, яка приймає рішення, та аналітиків – фахівців з методів дослідження операцій.

4. Активні засоби – ресурси, що використовуються для досягнення мети.

5. Методи дії, поведінки або використання активних засобів (ресурсів). Залежно від типу операції вони називаються рішеннями, альтернативами або стратегіями.

6. Результат – стан системи після операції.

7. Тип зв'язку між рішеннями (стратегіями) та результатами операції, що залежить від умов операції.

Прикладом операції візьмемо підготовку до модульного контролю вивчаємої студентами дисципліни.

Мета операції – успішна здача іспиту.

*Операційна сторон*а – студент.

Активні засоби – час, призначений для підготовки до іспиту, конспекти, підручники, консультації викладача тощо.

Можливі результати продиктовано системою оцінювання, що використовуються: зараховано / незазараховано, 4-х або 12-ти бальна оцінка.

Результат цієї операції залежить від багатьох чинників: неможливості освоєння всього матеріалу дисципліни за призначений час підготовки, варіанта тестових питань, самопочуття студента тощо.

Тому не існує повністю детермінованого зв'язку між стратегіями підготовки до іспиту та його результатами.

Іншим прикладом стохастичної взаємодії між рішеннями та результатами операцій можуть служити спортивні ігри (наприклад, футбол, гольф, шахи).

В дослідженні операцій відмічають 3 особливості:

1. Системний підхід – комплексна методологія вивчення складних систем або проблем. У цій методології визначаючим чинником є переважання цілей системи над цілями її підсистем.

2. Друга функція пов'язана з необхідністю вивчення та аналізу проблеми з різних точок зору. Тому група дослідників зазвичай складається з фахівців різних профілів (математиків, фізиків, психологів тощо), зусилля яких поєднані системною методологією. Така інтегрована команда дослідників дозволяє урізноманітнити альтернативи та знайти оптимальне рішення.

3. Використання наукових методів притамане будь-якій науці, але при дослідженні операцій вони мають свою специфіку, що пов'язано з кількісними критеріями оцінювання результатів операції.

1. ОСНОВНІ ПОНЯТТЯ ДОСЛІДЖЕННЯ ОПЕРАЦІЙ

1.1. Етапи операційного проєкту

Немає суворої регламентації процесу та змісту дослідження операцій, але в будь-якому завершеному проєкті можна виділити характерні етапи операційного проєкту.

1) *Постановка задачі*. Вона включає змістовний опис задачі: об'єкт та мета дослідження, внутрішні та зовнішні умови, ресурси, значення параметрів, можливі способи дії та можливі результати.

2) *Створення математичної моделі*. Характер задач дослідження операцій є таким, що їх рішення не може бути здійснене шляхом природного експерименту або фізичного моделювання. Дослідження здійснюються на математичних моделях, які містять:

– залежність критерію від керованих та некерованих змінних;

– рівняння, що відображають зв'язки між змінними, наприклад, рівняннями на основі матеріально-енергитичного балансу;

– обмеження, спричинені реальними умовами та вимогами до показників та змінних (ненегативність, цілочисленність, комплексність, допустимі та / або директивні значення тощо). У конкретних завдань не може бути окремих компонентів моделі. Однак у моделі повинна бути зформульована критеріальна функція.

3) *Перевірка адекватності моделі.* Математична модель є формалізованою гіпотезою щодо реальних відносин та поведінки системи. Тому, перш ніж використовувати модель для прогнозування наслідків та відбору рішень для реалізації, необхідно переконатися в адекватності системи або операції з точки зору поставленної цілі дослідження.

4) *Пошук оптимального рішення на моделі* – центральна стадія операційного дослідження (з математичної точки зору – це протилежна задача). Вона полягає у визначенні рішення, оптимального у сенсі прийнятого критерію. Для його пошуку використовуються методи оптимізації – математичне програмування.

5) *Аналіз оптимального рішення*. Він містить аналіз чутливості отриманого рішення до зміни умов операції, параметричний та варіантний аналіз, виявлення альтернативних оптимальних рішень тощо.

6) *Впровадження результатів досліджень*. Тут основна вимога полягає у забезпеченні прямої участі розробників на всіх етапах реалізації запропонованих рішень.

1.2. Критерій оптимальності

Термін «критерій», перекладений з давньогрецької мови означає міріло, оцінку. Критерієм є показник, який характеризує ефективність рішень з точки зору досягнення мети і, отже, дозволяє вибрати найкраще серед них. В дослідженні операцій використовуються еквівалентні терміни: критерій оптимальності, критерій ефективності, цільова функція.

Вибраний критерій повинен відповідати таким вимогам:

1. Правильно і повністю відображати поставлену мету.

4. Мати простий і зрозумілий фізичний сенс.

3. Мати кількісний детермінований вираз.

4. Бути чутливим до контрольованих змінних.

При вивченні існуючих систем до критерію можуть бути надані додаткові вимоги, такі як вимірюваність, статистична надійність, статистична ефективність тощо.

Багато показників, які використовуються як критерії, можуть бути умовно розділені на *соціальні* (середній дохід, забезпеченість житлом тощо), *економічні* (прибуток, рентабельність, вартість тощо), *техніко-економічні* (продуктивність, врожайність тощо), *техніко-технологічні* (міцність, чистота матеріалу, його фізичні або хімічні властивості) та ін.

Однак у багатьох випадках неможливо повністю відобразити ціль, встановлену одним критерієм, і тим більше неможливо, коли в операції багато цілей. У таких ситуаціях вводяться кілька показників, що характеризують досягнення мети. Задачі, в яких треба визначити найкраще рішення для декількох критеріїв, називаються *мультикритеріальними*.

1.3. Види математичних моделей дослідження операцій

Тип моделі визначається типом зв'язку між прийнятими рішеннями та отриманими результатами, що, у свою чергу, залежить від умов, в яких відбувається операція і приймається рішення.

1. *Детерміністичні моделі.* Якщо кожне рішення може бути внесено до відповідності певному результату, то маємо детерміністичний тип зв'язку. Такі моделі часто використовуються як перше наближення та в умовах, що відрізняються від ситуації впевненості.

2. *Імовірнісні моделі.* Якщо рішення приймаються в змінних умовах, їх зв'язок з результатами є стохастичним: певному рішенню може відповідати багато результатів, ймовірність появи яких відома. Адекватним відображенням таких умов є стохастичні моделі. Якщо результат означає значення критерію, то постановка задачі некоректна: неможливо максимізувати або мінімізувати випадкову величину. У цьому випадку в якості критерія повинна бути обрана одна з імовірнісних характеристик (математичне очікування, дисперсія або варіація).

Однак наявність випадкових чинників не завжди тягне за собою невизначеність результатів. Можливі випадки, коли елементарні компоненти процесу або системи поводяться випадковим чином, а поведінка системи в цілому не є випадковою. Така поведінка системи на макрорівні в наявності елементів випадковості на мікрорівні зветься *стохастичним детермінізмом*.

3. *Рішення приймаються в умовах невизначеності.* Це протилежна ситуація. Характер невизначеності може бути різним, але в цілому, це проявляється в тому, що певному рішенню відповідають кілька результатів, ймовірнісні характеристики яких невідомі. Математичні моделі, що описують такий невизначений тип зв'язку, різноманітні і не мають жодного імені. Зокрема, цей клас включає в себе ігрові моделі, матричні моделі типу «аукціон», нечіткі моделі.

Таким чином, вибір типу моделі, поряд з глибоким знанням предметної області, вимагає досвіду та інтуїції. Створення моделі базується на уявленнях аналітика, яке не може повністю відповідати реальному стану справ. При цьому велике значення мають оцінка впливу випадкових чинників, факторів невизначеності, рівня агрегації систем, допустима складність моделі.

Тому часто виникає дилема: побудувати точну, але дуже складну модель, на якій можна буде отримати лише приблизне до оптимального рішення, або поступитися точністю моделювання, але мати можливість застосовувати на моделі точні математичні методи оптимізації.

1.4. Класи операційних задач

В даний час вивчення операцій знаходить використання майже у всіх сферах людської діяльності. Звідси широкий спектр математичних моделей операцій та методів їх досліджень. Виділяють класи типових завдань, розгляд яких дозволяє краще подати коло проблем дослідження операцій. Розглянемо найбільш характерні для них.

1) Задача управління запасами. Під запасами розуміють ресурси, які на даний момент не використовуються. Це можуть бути матеріали, обладнання, напівфабрикати, готові вироби, приміщення, персонал, фінансові кошти тощо.

Проблема управління запасами може бути зменшена до 2-х питань:

1) скільки замовляти (купувати або виробляти);

2) коли або як часто замовляти.

Особливість цієї задачі полягає в тому, що статті витрат по-різному залежать від рівня запасів.

Зі збільшенням запасів зростають витрати на зберігання (складські витрати, заморожування оборотного капіталу, втрати від пошкоджень та старіння запасів, моральний знос тощо) і, в той же час, зменшуються витрати виробництва, пов’язані із відсутністю резервів (простої обладнання, аварії, штрафи тощо). Як наслідок, задача полягає в тому, щоб вибрати такі параметри управління запасами (обсяг партії, період поповнення тощо), які забезпечують мінімум загальних витрат, пов'язаних з запасами.

Розрізняють 3 основні ознаки класифікації запасів: за типом попиту – випадковий і не випадковий; за методом поповнення – миттєвий або із затримкою; за видом – однорідні та неоднорідні. Очевидно, що найпростіші задачі характеризуються не випадковим попитом, миттєвим поповненням та однорідними запасами.

2) Задачі розподілу зустрічаються на практиці найчастіше. Вони виникають у ситуаціях, коли можливі різні способи реалізації операцій. Залежно від рівня ресурсів, виділяються 3 групи цих задач.

Для *першої групи* характерно, що наявних ресурсів достатньо для виконання усіх робіт, але недостатньо для виконання кожної роботи оптимальним чином. У цих умовах необхідно так розподілити ресурси між усіма операціями (споживачами), щоб досягти найбільшої ефективності системи в цілому.

Класичним прикладом такої задачі є збалансована транспортна задача. З одного боку, є постачальники з відомими обсягами вантажу, з іншого – споживачі з відомими вантажними потребами, а кількість потреб дорівнює кількості можливостей (баланс). Крім того, для всіх пар «споживач–постачальник» відома вартість транспортування вантажу від постачальника до споживача.

Очевидно, що найкращим варіантом для задоволення окремого споживача буде доставка з мінімальними для нього витратами, але це може призвести до значного збільшення транспортних витрат у інших споживачів. Тому задача розподілу полягає в тому, щоб визначити таку схему перевезення, за якою загальні транспортні витрати будуть мінімальними.

Друга група включає задачі, в яких наявних ресурсів недостатньо, щоб виконати усі роботи або задовольнити усіх споживачів у повному обсязі. Задача є аналогічною вищезазначеній, але в деяких випадках необхідна додаткова інформація щодо впливу незадоволеного попиту на показник продуктивності, а рішення повинно містити дані про те, які роботи і в якому об'єзі не виконуються в умовах оптимального розподілу.

Прикладами таких задач можуть бути незбалансована транспортна задача, складання бюджету, задачі планування, складання сумішей тощо.

Задачі *третьої групи* відрізняються тим, що рівень (обсяг) використовуваних ресурсів не зафіксований і може змінюватися в деяких межах. У той же час витрати ресурсів залежать від їх обсягів. Завданням є визначення оптимального рівня ресурсів та оптимального їх розподілу за критерієм, який враховує як витрати ресурсів, так і ефективність їх використання.

3) Задачі масового обслуговування виникають, коли є потік заявок, вимог або клієнтів, які потребують обслуговування. Загалом, заявки проходять через випадкові інтервали часу, а тривалість обслуговування однієї заявки є також випадковою.

Системи, зайняті обслуговуванням таких потоків, називаються *системами масового обслуговування (СМО)*. Зазвичай в СМО розрізняються 4 компоненти: потік заявок (вхідний потік), черга заявок, обслуговуючі пристрої, вихідний поток.

Як випливає з характеру надходження та обслуговування заявок, СМО може виникнути як черга заявок на обслуговування, так і черга обслуговуючих пристроїв (тобто простоїв). Очевидно, що з будь-якою чергою пов'язані втрати.

Відрізняють СМО з відмовами, коли при завантажених пристроях заявка, що прийшла в систему, отримує відмову і, отже, черги заявок не може бути, і СМР з очікуванням, коли при завантажених пристроях обслуговування завка стоїть в черзі і чекає обслуговування.

Прикладами можуть служити задачі організації торгівлі, медичної допомоги, ремонту, серійного та масового виробництва, робота обчислювального центру та окремого комп'ютера у багатозадачних та багатокористувацьких режимах тощо.

4) Задачі выбору маршруту. Залежно від типу бажаного маршруту, є 2 варіанти таких задач.

Перший тип іноді називають *задачею про найкоротший шлях*. Між 2 вказаними елементами або вузлами є кінцевий набір шляхів, що складаються з переходів, що з'єднують проміжні точки. Один перехід може входити більше, ніж в один шлях. Кожен перехід характеризується індикатором або кількістю показників, які можуть бути часом, довжиною, вартістю, споживанням ресурсу тощо. Потрібно знайти шлях, найкоротший у сенсі прийнятого критерію.

Другий тип вибору маршруту називається *задачею комівояжера*. Таке ім'я розвинулося історично і пов'язане з пошуком оптимального маршруту руху представника торгової компанії – комівояжера. Останній, вийшовши зі свого міста, повинен обійти всі міста у сфері служби компанії та повернутися назад. У той же час кожне місто відвідується 1 раз. Є показники переходів між усіма парами міст, і потрібно знайти маршрут, який забезпечує мінімальні витрати часу або мінімальне споживання палива, або мінімальну вартість проїзду. Ця задача відрізняється замкнутістю бажаного маршруту та необхідністю проходження усіх пунктів.

5) Задачі заміни обладнання. Залежно від типу обладнання, відрізняють 2 типи задач заміни.

У одних задачах обладнання розглядається цілком. Його характеристики з часом погіршуються. Як наслідок, зменшуються продуктивність та якість операцій, зростають витрати на експлуатацію та поточний ремонт, знижується фонд корисного робочого часу обладнання.

Зі збільшенням частоти заміни обладнання всі ці показники будуть покращені, але в той же час різко зростуть капітальні витрати та витрати, спричинені демонтажом старого та встановленням нового обладнання. Тому виникає питання про визначення моменту заміни обладнання, найкращих у сенсі обраного критерію.

Інший тип задач заміни пов'язаний з обладнанням, що складається з великої кількості недорогих елементів, характеристики яких практично не змінюють їх властивості, але вони можуть раптово виходити з ладу. Моменти відмови, як правило, випадкові. Прикладами такого обладнання можуть служити електронні схеми, комп'ютери тощо. На відміну від завдань першого виду тут, поряд з моментами заміни, необхідно визначити рівень, на якому слід замінити заміну.

6) Задачі впорядкування виникають у зв’язку з тим, що остаточний набір незалежних робіт (операцій) виконується на одній групі обладнання, включаючи 2 чи більше верстата (обслуговуючі пристрої). Кожній парі «операція – машина» покладається у відповідність деякий індикатор. Задача полягає в тому, щоб визначити таку послідовність незалежної роботи одного й того ж обладнання, при якій досягається найкраще значення критерію оптимальності.

Очевидно, що клас завдань впорядкування досить широкий. В нього входять різноманітні задачі складання розкладів. Як і задачі вибору маршруту, розглянуті задачі відносяться до комбінаторних, складності вирішення яких пов'язано з їх дискретністю та експоненціальним зростанням кількості варіантів зі збільшенням розмірності задачі.

7) Задачі мережевого планування та управління. На відміну від задач упорядкування, у завданні мережевого планування розглядається комплекс взаємопов'язаних робіт. Джерелом є перелік робіт, що підлягають виконанню, з відомою тривалістю кожної роботи та з переліком робіт, що безпосередньо їм передують.

Відносини робіт моделюються орієнтованим графом, що називається мережевим розкладом або просто мережею. Можливі 2 варіанти мережі:

1) мережа типу «роботи – дуги», коли дуга відображає роботу з властивими їй параметрами (показниками) та зв’язками, а вершини – стан об'єкта (програми, проєкту), до якої відносяться роботи;

2) мережа типу «вершини – робота», коли дуги відображають лише зв’язки, а роботам ставляться у відповідність вершини. Для початкової побудови мережі другий варіант більш зручний, але перший тип частіше використовується для розрахунків.

В задачах мережевого планування та управління розрізняють аналіз та синтез мережі. Аналіз мережі полягає у розрахунку часу початку та закінчення роботи, резервів робочого часу, визначення критичного шляху та критичних робіт. Критичним є найдовший шлях від початкової події до фіналу, тобто це мінімальний час, за який можна виконати усі роботи.

Під синтезом мережі зазвичай розуміють її оптимізацію. Наприклад, в межах виділених ресурсів або витрат необхідно забезпечити мінімальний час завершення всього комплексу робіт або, навпаки, виконати усі роботи до певного періоду з мінімальними витратами.

8) Конкурентні задачі. Цими задачами моделюється прийняття рішень у ситуаціях невизначенності, обумовлених наявністю конфлікту, що виникає між операційними сторонами, які переслідують невідповідні цілі. Невизначеність у однієї із сторін виникає у зв'язку з невідомою лінією поведінки інших сторін, тоді як результат операції залежить від поведінки всіх учасників операції.

Розділяють 2 моделі конфліктів: гру та аукціон.

Гра характеризується відомою кількістю учасників, які називаються *гравцями*, правилами гри, багатьома можливими ситуаціями (комбінацій стратегій гравців) та відповідних їм виграшів або збитків, які називаються *платежами*.

Відповідно до методу гри відрізняються дискретні ігри, в яких гравці роблять чергові кроки, та безперервні, коли гравці діють безперервно. Останні також звуться іграми переслідування або диференціальними іграми, оскільки поведінка гравців описується диференціальними рівняннями. За кількістю ходів виділяють ігри з кінцевою та нескінченною кількістю кроків. Подібний розподіл здійснюється за кількістю стратегій гравців. Ігрові моделі використовуються в основному при дослідженні фінансових операцій.

Моделі *аукціону* використовуються, коли ступінь невизначеності вище, ніж припускає модель гри. Наприклад, в реальній аукціонній торгівлі, як правило, невідома навіть кількість учасників. Розробка та дослідження моделей цього типу знаходяться на початковому етапі.

9) Задачі пошуку об’єктів у предметній області, які задовольняють певним вимогам. Процес пошуку пов'язаний з такими вирогідними помилками.

Помилка вибірки виникає через неможливість охоплення усього набору об'єктів, серед яких можуть бути шукані. При дослідженні вибірки може бути *помилка спостереження*, яка полягає в тому, що не виявляється (пропускається) шуканий об'єкт, який входить у вибірку, або інший об'єкт приймається як бажаний (помилкове виявлення). Будь-які помилки призводять до втрат, упущенню прибутку тощо.

Для пошуку необхідні ресурси. З обмеженими ресурсами збільшення вибірки зменшує помилку відбору, але в той же час збільшується помилка спостереження, а зі зменшенням розміру вибірки – навпаки.

Задача полягає у визначенні розміру вибірки та стратегії пошуку, при яких досягається найбільша ефективність пошуку. Характерним прикладом такої задачі є контроль якості деталей або продуктів у серійному або масовому виробництві. Основним ресурсом при цьому виступають контролери ВТК, кількість яких завжди обмежена. Чим більше вибірка, тим більше непридатних деталей пройде через контролера, але тим менше часу у нього на тестування кожної деталі, яка потрапила у вибірку. Це означає більшу ймовірність пропуску непридатної деталі або відхилення придатної. Дефектні деталі, не виявлені контролером, можуть проявити себе лише при тестуванні готової продукції або у споживача, що тягне за собою збитки для виробника. Задача контролю якості полягає в тому, щоб організувати це так, щоб втрати були зведені до мінімуму.

Задача пошуку може бути ширше, якщо обсяг ресурсів, що виділяються на пошук, може відрізнятися в певних обмеженнях. Оскільки ресурси повинні бути сплачені, то їх збільшення не обов'язково призводить до зростання ефективності. Тому задача полягає у визначенні оптимального рівня ресурсів та оптимального їх використання.

Таким чином, на етапі розробки проєкту з контролю якості може виникнути необхідність визначення кількості контролерів, обсягу вибірки та стратегії контролю всередині, що забезпечить максимальний прибуток від виробленої продукції.

Деякі області застосування задач пошуку:

– організація ревізій;

– бухгалтерські операції;

– виявлення об'єктів (потерпілі аварій, пожеж тощо);

– розвідка корисних мінералів;

– організація контролю якості;

– зберігання та пошук інформації;

– розміщення реклами в районі або місті.

1.5. Постановка задачі оптимізації

Центральним етапом дослідження операцій є знаходження найкращого рішення від усіх можливих у данній операції.

У математичній моделі експлуатації це описується *цільовою функцією* (критерієм оптимальності) $f(X)$, а можливі рішення – областю допустимих рішень (допустимий множиною) $X \in D$.

У загальному випадку задача оптимізації виглядає так: знайти такий вектор $X^* \in D$, що забезпечує оптимум функції $f(X)$, тобто

$$f(X) \to \underset{X \in D}{\text{opt}}. \qquad (1.1)$$

Якщо рішення X^1 краще рішення X^2 при $f(X^1) > f(X^2)$, то кажуть про *задачу максимізації.* Якщо ж X^1 краще X^2 при $f(X^1) < f(X^2)$, то маємо *задачу минимізації.*

У задачах дослідження операцій множина *D* визначається кінцевим числом умов, у загальному випадку у вигляді рівнянь, нерівностей і прямих обмежень для змінних.

Іншим великим класом задач оптимізації є *задачі оптимального управління*. В них мета описується функціоналом, а шуканими є функції. Такі задачі виникають переважно при розробці та дослідженні динамічних або розподілених систем.

Найважливішими характеристиками задачі оптимізації є властивості функцій, включених до моделі, та вимір задачі. Розмірність задачі математичного програмування визначається двома величинами: кількістю *N* шуканих змінних (тобто, розмірністю вектора *X*) та кількістю *M* граничних умов. Суттєве значення має також структура математичної моделі задачі, наявність випадкових чинників, безперервність та дискретність змінних.

Різноманітність завдань оптимізації величезна, і сьогодні немає універсального методу, з яким можна було б вирішити будь-яку задачу. Загалом можна виділити наступні методи вирішення задач оптимізації:

– класичний аналіз (включаючи метод множників Лагранжа);

– лінійне програмування;

– динамічне програмування;

– дискретне програмування;

– нелінійне програмування;

– стохастичне програмування.

Всі вищезгадані групи методів, за винятком першого, мають загальну назву *методів математичного програмування*.

Методи лінійного програмування орієнтовані на моделі, в яких всі функції є лінійними, а змінні безперервні.

Дискретність (зокрема, цілочисленність) змінних – особливість методів дискретного програмування.

Динамічне програмування стосується завдань, в яких процес прийняття рішень може бути представлений як послідовність кроків.

Нелінійне програмування включає дуже широкий спектр методів вирішення проблем з нелінійними функціями різних структур.

Значний вплив випадкових чинників призводить до стохастичних моделей, щоб знайти рішення на яких використовуються імовірнісні методи.

Контрольні запитання до розділу 1

1. Назвіть етапи операційного проєкту.
2. Що включає постановка задачі операційного проєкту?
3. Що таке перевірка адекватності матмоделі в дослідженні операцій?
4. Як здійснюється пошук оптимального рішення на моделі?
5. Як здійснюється використання результатів дослідження?
6. Які вимоги висуваються до критерію оптимальності?
7. Охарактеризуйте види матмоделей, які використовуються в дослідженні операцій.
8. Перелічіть класи задач в дослідженні операцій.
9. У чому полягяє задача оптимального керування?
10. Назвіть методи розв'язання задач оптимізації.

2. ЛІНІЙНЕ ПРОГРАМУВАННЯ

Лінійне програмування – це область математики, присвячена теорії і методам вирішення крайніх задач, що характеризуються лінійними залежностями між змінними. Задачі лінійного програмування складають найпростіший і найбільш добре вивчений клас задач математичного програмування.

2.1 Математична модель

Побудова математичної моделі передбачає отримання відповіді на наступні питання.

1. Для визначення яких змінних (невідомих) необхідно побудувати модель? Іншими словами, необхідно ідентифікувати шукані величини.

2. В чому покладається мета вирішення задачі?

3. Якими обмеженнями повинні бути пов’язані змінні, щоб виконувались умови, характерні для поставленої задачі?

Відповіді на поставлені питання повинні міститися в описі проблеми. Якщо ці відповіді знайдені, то мета задачі і обмеження представлені як математичні функції ідентифікованих змінних.

Розглянемо кілька прикладів побудови математичних моделей лінійного програмування.

Задача планування виробництва. Деякій виробничій одиниці (заводу, цеху) необхідно визначити план виробництва k видів продукції. Кількість видів ресурсів дорівнює n. При цьому відома кількість кожного ресурсу b_i, $i = 1, m$. Технологічні можливості виробництва визначаються значеннями чисел a_{ij}, $i = 1, m$, $j = 1, n$, які показують, скільки i-го ресурсу необхідно для випуску однієї одиниці j-го продукту. Технологія виробництва вважається лінійною, тобто вважається, що всі ресурсні витрати зростають прямо пропорційно обсягу випуску.

Задано ціни $C_{ij} = 1, n$ для кожного виду продукту. Потрібно визначити виробничий план, що максимізує вартість випущеної продукції.

Опис завдання містить відповіді на зазначені вище питання.

План виробництва – це вектор $X = (x_1, x_2, ..., x_n)$, який показує, скільки буде вироблено кожного продукту. Це змінні задачі.

Мета рішення задачі (*цільова функція*) – максимальна вартість випущеної продукції:

$$Z(x) = \sum_{j=1}^{n} C_j \, x_j \to \max \tag{2.1}$$

З умови обмежених ресурсів, з одного боку, і загальних витрат ресурсів для виконання виробничого плану, з іншого боку, побудовано систему обмежень математичної моделі:

$$\sum_{j=1}^{n} a_{ij} x_i \le b_i, \quad i = \overline{1,m}; \quad x_j \ge 0, \quad j = \overrightarrow{1,n} \tag{2.2}$$

Математична модель (2.1)–(2.2), хоча й відображає певні особливості реального виробництва, є ідеалізованою. В ній, наприклад, не враховано динаміку виробництва, ритмічність поставок та деяких інших властивостей реального виробництва.

Транспортна задача. Транспортна модель використовується при розробці плану транспортування однорідних виробів з *m* пунктів відправлення в *n* пунктів призначення. При побудові моделі використовуються:

– величини, що характеризують обсяг виробництва a_i , $i = 1, m$ у кожному *i*-му вихідному пункті та попит $b_j = 1, n$ в кожному *j*-му пункті призначення;

– вартість *Cij* перевезення транспортної одиниці з кожного вихідного пункта до кожного пункта призначення.

Оскільки розглядається однорідний вид продукції, потреби пункта призначення можуть бути задоволені за рахунок будь-яких вихідних пунктів.

Мета вирішення транспортної задачі полягає у визначенні кількості продукції, яку слід транспортувати з кожного вихідного пункту до кожного пункту призначення таким чином, щоб загальні витрати на транспортування були мінімальними.

Основне припущення, що використовується при побудові моделі, полягає в тому, що величина транспортних витрат на кожному маршруті прямо пропорційна обсягу транспортуємої продукції. Крім того, вважається, що загальний обсяг виробництва дорівнює загальному попиту:

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j. \qquad (2.3)$$

Нехай x_{ij} – кількість продукції, яку слід транспортувати з вихідного пункту *i* до пункту призначення *j*. Тоді математична модель транспортної задачі формулюється такми чином:

– цільова функція

$$Z(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} C_{ij}\ x_{ij} \to \min \qquad (2.4)$$

– при обмеженнях

$$\begin{cases} \sum_{j=1}^{n} x_{ij} = a_i\ ,\ i = \overline{1,m} \\ \sum_{i=1}^{m} x_{ij} = b_j\ ,\ j = \overline{1,n} \end{cases} \qquad (2.5)$$

$$x_{ij} \geq 0,\ i = \overrightarrow{1,m},\ j = \overrightarrow{1,n}$$

Модель (2.4)–(2.5) з умовою (2.3) зветься *сбалансованою транспортною моделлю.* У реальних умовах обсяг виробництва не завжди дорівнює попиту. Однак транспортну модель завжди можна збалансувати шляхом введення фіктивних постачальників або клієнтів.

Транспортна модель має ряд важливих застосувань, які включають багато продуктову транспортну задачу та задачу з проміжними пунктами, задачу призначення та задачу складання змінних графіків. У той же час, транспортна модель та її узагальнення є особливими випадками мережевих моделей.

Задача про оптимальну суміш. Завдання визначення оптимального складу суміші виникає, коли з наявних видів сировини необхідно отримати новий продукт шляхом змішування (наприклад, бензин, металевий сплав тощо) з заданими технічними та іншими характеристиками. При цьому потрібно, щоб вартість такої суміші була мінімальною.

Припустимо, що суміш необхідно скласти з *n* різних видів сировини, кожна з яких містить види інтересуючих нас елементів (речовин).

Нехай a_{ij}, $i = 1,m$, $j = 1,n$ – кількість i-ї речовини в одиниці j-го виду сировини, вартість якої є $C_{ij} = 1,n$. Позначимо b_t найменшу допустиму кількість i-ї речовини, а через d_i – обсяг запасу сировини i-го виду.

Нехай X_J буде кількістю сировини j-го виду, яка повинна бути використана для складання суміші. Введені позначення дозволяють нам побудувати наступну математичну модель задачі:

– цільова функція

$$Z(x) = \sum_{j=1}^{n} C_j \; x_j \to \min \tag{2.6}$$

– при обмеженнях

$$\sum_{j=1}^{n} a_{ij} x_i \le b_i, \quad i = \overline{1,m}. \tag{2.7}$$

Умова негативності змінних $x_{ij} \ge 0$ є природньою і випливає з визначення цих змінних.

Залежно від конкретної постановки задачі та наявності різноманітних вимог, математична модель (2.6)–(2.7) може бути доповнена іншим видом обмежень.

2.2 Загальна, стандартна і канонічна форми моделі

Незважаючи на різницю в значущих ситуаціях, описаних у наведених прикладах, екстремальні математичні моделі, що відповідають їм, мають багато спільного. Отже, у кожній з цих задач необхідно максимально збільшити або мінімізувати лінійну функцію від декількох змінних. При цьому обмеження, накладені на комбінацію змінних, є лінійними нерівностями або лінійними рівняннями.

Кожна з лінійних задач програмування є особливим випадком *загальної задачі лінійного програмування*, математична модель якої складається з цільової функції

$$Z(x) = \sum_{i=1}^{m} C_i \; x_i \to \max(\min) \tag{2.8}$$

та системи обмежень:

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i, \quad i = \overline{1,k}; \quad k \le m, \tag{2.9}$$

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, i = \overline{k+1,m}, \tag{2.10}$$

$$x_j \ge 0, \ j = \overrightarrow{1,l}, \ l \le n, \tag{2.11}$$

де a_{ij}, b_i, C_j – задані постійні величини і $k \le m$.

Залежно від наявності різного типу обмежень відрізняють такі основні форми задач лінійного програмування.

Стандартна (або симетрична) форма, яка полягає у визначенні максимального значення функції (2.8) при виконанні умов (2.9), де $k = m$, $l = n$.

Стандартна форма моделі цікава тим, що велика кількість прикладних моделей природним чином зводиться до цього виду моделей.

Канонічна (або базова) форма моделі, що полягає у визначенні мінімального значення функції (2.8) при виконанні умов (2.9), де $k = 0$, $l = n$.

Канонічна форма моделі важлива, зважаючи на те, що основні обчислювальні схеми різних варіантів симплекс-методу розроблені спеціально для цієї форми.

Наведені форми задачі лінійного програмування еквівалентні в тому сенсі, що кожна з них за допомогою простих перетворень може бути зведена до двох інших. Отже, будь-яка задача лінійного програмування може бути приведена до канонічної форми. Тому здатність вирішувати задачу в канонічній формі дозволяє вирішити задачу в будь-якій іншій формі.

Щоб перейти від загальної (стандартної) форми лінійної проблеми програмування до канонічної форми, треба мати можливість, по-перше, зводити задачу мінімізації функції до задачі максимізації; по-друге, перейти від обмежень-нерівностей до обмежень-рівнянь; по-третє, заміняти ненегативними змінними ті змінні, які не підпорядковуються умовам негативності.

В тому випадку, коли необхідно знайти мінімум функції $Z_1 = c_1x_1 + c_2x_2 + \ldots + c_nx_n$, можна перейти до знаходження максімума функції $Z_2 = -z = -c_1x_1 - c_2x_2 - \ldots - c_nx_n$, оскільки $\min Z = -\max(-Z)$.

Обмеження-нерівність задачі лінійного програмування можна перетворити в обмеження-рівняня, додаючи до її лівої частини додаткову негативну змінну з відповідним знаком. Таким чином, обмеження-нерівність

виду $a_{ijx1} + a_{i2}x_2 + ...+ a_{in}x_n \le b_t$ перетворюється в обмеження-рівність $a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{ln}x_n + x_{n+1} = b_t$, $x_{n+1} \ge 0$, а обмеження-нерівність вида $a_{i1}x_1 + a_{i2}x_2 \ldots + a_{in}x_n \ge b_t$ – в обмеження-рівність $a_{i1}x_1 + a_{i2}x_2 + ...+ a_{in}x_n - x_{n+2} = b_i$,, $x_{n+2} \ge 0$.

Кількість додаткових негативних змінних у таких перетвореннях дорівнює кількості перетворених нерівностей.

І, нарешті, якщо змінна x_k не підпорядкована умові негативності, то її слід замінити двома ненегативними змінними x'_k и x''_k,, принявши $x_k - x'_k - x'_k$. Правомірність такої заміни очевидна, оскільки будь-яке число може бути представлено у вигляді різниці двох негативних чисел.

Лінійне програмування розробило певну термінологію, якої ми дотримуємося.

Визначення 2.1. Сукупность чисел – вектор $X = (x_{lt}\ x_г, \ldots, x_n)$, що задовольняють обмеження, наприклад (2.9)–(2.11), називається *допустимим рішенням*.

Визначення 2.2. Допустиме рішення $X^* = (x^*_1, x_2^*, \ldots, x^*)$, при якому цільова функція задачі, наприклад (2.8), приймає максимальне (мінімальне) значення, називається *оптимальним рішенням*.

Введемо для задачі (2.8)–(2.11) такі поняття і визначення:

$$A_1 x_1 + A_2 x_2 + ... + A_n x_n = B; \quad X \ge 0, \ Z = C \cdot X > max, \qquad (2.12)$$

де $C = (c_1, c_2, .., c_n)$; $X = (x_1, x_2 \ldots x_n)$; $C \cdot X$ – скалярний добуток векторів; $A_l, A_2, \ldots, A_n$ і B – мірні вектори-стовпці, складені з коефіцієнтів при невідомих та вільних членах системи рівнянь канонічної задачі:

$$A_1 = \begin{pmatrix} a_{11} \\ a_{21} \\ \ldots \\ a_{m1} \end{pmatrix}; \quad A_1 = \begin{pmatrix} a_{11} \\ a_{21} \\ \ldots \\ a_{m1} \end{pmatrix}; \quad \ldots; \quad A_1 = \begin{pmatrix} a_{1n} \\ a_{2n} \\ \ldots \\ a_{mn} \end{pmatrix}; \quad B = \begin{pmatrix} b_1 \\ b_2 \\ \ldots \\ b_m \end{pmatrix}.$$

Визначення 2.3. Рішення $X = (x_1, x_2, \ldots, x_n)$ канонічної задачі (2.12) зветься *опорним планом,* якщо система векторів a_j, відповідних позитивним компонентам x_j, лінійно незалежна.

Оскільки вектори Aj є m-мірними, то з визначення опорного плану випливає, що кількість його позитивних компонентів не може бути більшою, ніж m. Решта $(n - m)$ компонент дорівнює нулю.

Визначення 2.4. Опорний план називається *невиродженим*, якщо він містить рівно m позитивних компоненти, інакше він називається *виродженим*.

Визначення 2.5. Непуста множина планів канонічної задачі (2.12) є опуклим і називається *багатогранником рішень*, а його кутова точка – *вершиною.*

Рішення будь-якої задачі лінійного програмування також можна знайти симплексним методом, який реалізує ідею послідовного вдосконалення опорного плану.

2.3 Метод Жордана-Гаусса рішения систем лінійних рівнянь

Нагадаємо з курсу лінійної алгебри метод Жордана-Гаусса, що дозволяє вирішити систему лінійних алгебраїчних рівнянь виду

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, \quad i = \overline{1,m}. \tag{2.13}$$

Ця система перетворюється в еквівалентну їй систему за такими формулами:

$$a_{ij} = \begin{cases} \dfrac{a_{ij}}{a_{lk}}, & i = l \\ a_{lj} - \dfrac{a_{lk}\, a_{lk}}{a_{lk}}, & i \neq l \end{cases} \tag{2.14}$$

$$b_i = \begin{cases} \dfrac{b_l}{a_{lk}}, & i = l \\ b_i - \dfrac{b_l\, a_{lk}}{a_{lk}}, & i \neq l \end{cases} \tag{2.15}$$

де a_{ik} – дозволяючий елемент, який показує, що в результаті перетворень змінна x_k залишається лише в рівнянні j і, крім того, з коефіцієнтом 1; a_{ij} і b'_i – нові значення відповідних коефіцієнтів та вільних членів системи, отримані внаслідок перетворень.

Змінні, відносно до яких можна вирішити систему рівнянь, називаються *базисними змінними*. Решта змінних – *небазисні (або вільні*).

Результатом використання методу Жордана-Гаусса є наступне: або встановлюється, що система рівнянь є неспільною (всі коефіцієнти наступного розглянутого рівняння нульові, а вільний коефіцієнт відрізняється від нуля), або виявляються і відкидаються усі «зайві» рівняння, у яких всі коефіцієнти та вільний елемент є нульовими. «Зайві» рівняння – це рівняння, лінійно залежні від інших системних рівнянь.

Припустимо, що ранг матриці A коефіцієнтів вихідної системи рівнянь (2.14) дорівнює m. Тоді, як наслідок перетворення, початкова система рівнянь прийме, наприклад, такий вид:

$$x_i + \sum_{j=m+1}^{n} a_{ij}\, x_j = b_i, \quad i = \overline{1,m}, \tag{2.16}$$

де x_i – базисні змінні, що утворюють одиничный базис; x_j – небазисні змінні, значення яких довільні.

Трансформована система рівнянь (2.14)–(2.15) дає явний опис множини всіх рішень. Іншими словами, ця система рівнянь є загальним рішенням системи (2.13), з якого можна отримати конкретне рішення, якщо вказати будь-якими значеннями небазисних змінних.

Визначення 2.6. Конкретне рішення зветься *базисним рішенням,* якщо небазисні змінні $x_j = 0,\ j = m + 1,\ n.$

Отже, базисне рішення системи (2.14)–(2.15) дорівнює вектору $X = (b_1, b_2, ..., b_m, 0, 0, ..., 0)$. Ненегативне базисне рішення, відповідно до визначення 2.3 – це опорний план.

Перетворення виконуються над коефіцієнтами системи рівнянь (2.14)–(2.15), тому рішення зручно виконувати у вигляді послідовності таблиць, стовпчики яких містять коефіцієнти при невідомих та вільни члени.

Алгоритм методу Жордана-Гаусса включає наступні основні кроки.

1. Вибір дозволяючого елемента $a_{ik}^{(t)}$, відмінного від нуля; $t = 0, 1, 2, ...$ – номер ітерації.

2. Ділення елементів дозволяючої i-ї строки на дозволяючий елемент за формулами:

$$a_{lj}^{(t+1)} = \frac{a_{lj}^{(t)}}{a_{lk}^{(t)}}, \quad j = \overline{1,n}. \tag{2.17}$$

3. Перетворення усіх інших елементів таблиці за формулами:

$$a_{ij}^{(t+1)} = a_{ij}^{(t)} - \frac{a_{lj}^{(t)} a_{ik}^{(t)}}{a_{lk}^{(t)}}, \quad i \neq l; \tag{2.18}$$

$$b_{i}^{(t+1)} = b_{i}^{(t)} - \frac{b_{l}^{(t)} a_{ik}^{(t)}}{a_{lk}^{(t)}}, \quad i \neq l, \tag{2.19}$$

причому, при $j = k$ $a_{ij}^{(t+1)} = 0$.

Наступні ітерації починаються з п.1. Рішення системи отримано, якщо побудовано одиничний базис. У разі наявності небазисних змінних розглянутий алгоритм дозволяє перейти до нового одиничного базису. При цьому дозволяючий елемент обирається у стовпці коефіцієнтів небазисної змінної, яка в результаті перетворень стає базисною, а колишня базисна змінна x_i – небазисною.

2.4 Прямий симплекс-метод

Формули перетворення методу Жордана-Гаусса використовуються в прямому симплекс-методі рішення задачі лінійного програмування, зведеного до канонічної форми (2.11). На підставі цих перетворень, перехід від одного опорного плану до іншого, в якому збільшується значення цільової функції (якщо опорний план існує, і він невироджений).

Цей перехід можливий, якщо відомо деякий початковий план. Якщо канонічна форма задачі не має початкового опорного плану, то він будується за допомогою штучних змінних. Однак, незалежно від того, використовуються штучні змінні чи ні, для вирішення задачі використовується один і той же алгоритм симплекс-метода.

Нехай задача у канонічній формі має початковий опорний план:

$$Z(x) = \sum_{j=1}^{n} Cj\, x_j \to \max$$

$$x_i + \sum_{i=m+1}^{n} a_{ij}\, x_j = b_i, \quad i = \overline{1,m}, \tag{2.20}$$

$$x_j \geq 0,\ j = \overrightarrow{1,n},\ l \leq n; \quad b_i \geq 0,\ i = \overrightarrow{1,m}.$$

За визначенням 2.3, вектор рішень $X = (b_1, b_2, ..., b_m, 0, ..., 0)$ є опорним планом задачі (2.14)–(2.15). Цьому плану відповідає наступне значення цільової функції:

$$Z = C_1 x_1 + C_2 x_{21} + ... + C_m x_m = \sum_{i=1}^{m} C_i b_i . \qquad (2.21)$$

Для дослідження опорного плану на оптимальність і для визначення напрямку його вдосконалення на основі цільової функції будується «нульове» приведене рівняння. Це рівняння виходить після виключення базових змінних з виразу цільової функції.

Замінюючи $x_1, x_2, ..., x_m$ з системи обмежень (2.20) в цільову функцію, отримуємо «нульове» приведене рівняння у такій формі:

$$Z + a_{0m+1} x_{m+1} + a_{0m+2} x_{m+2} + ... + a_{0n} x_n = b_0 ; \qquad (2.22)$$

де $a_{0j} = \sum_{i=1}^{m} C_i a_{ij} - C_i , \quad j = \overline{m+1, n}; \quad b_0 = \sum_{i=1}^{m} C_i b_i .$

Коефіцієнти $\boldsymbol{a_{0j}}$ називаються оцінками відповідних небазисних змінних *(або оцінками оптимальності)*. Ці оцінки дозволяють характеризувати опорний план у вигляді наступних теорем-ознак можливих ситуацій.

Теорема 2.1. Якщо $a_{0j} \geq 0, j = 1, n$, то еталонний план є оптимальним (ознака оптимальності опорного плану).

Теорема 2.2. Якщо $a_{0j} < 0$ для деякого індексу j та усі відповідні до цього індексу величини $a_{ij} \leq 0, i = 1,m$, то значення цільової функції не обмежується згори на множині планів.

Теорема 2.3. Якщо $a_{0j} < 0$ для деяких індексів j і для кожного з цих індексів, принаймні, одне з чисел $a_{ij} > 0$, то можна перейти до нового опорного плану, при якому значення цільової функції збільшується.

Обчислювальний процес зручно вести в симплекс-таблицях. Початкова симплекс-таблиця для задачі (2.20) з урахуванням «нульового» рівняння (2.22) представлена в таблиці. 2.1. «Нульове» рівняння наведено у нижній частині цієї таблиці.

Таблиця 2.1

t	Базисні змінні	x_1	x_2	…	x_m	x_{m+1}	…	x_n	b_i
1	x_1	1	0	…	0	$a_{1\,m+1}$	…	$a_{1\,n}$	b_1
2	x_2	0	1	…	0	$a_{2\,m+1}$	…	$a_{2\,n}$	b_2
…	…	…	…	…	…	…	…	…	…
m	x_m	0	0	…	1	$a_{m\,m+1}$	…	$a_{m\,n}$	b_m
0	x	0	0	…	0	$a_{0\,m+1}$	…	$a_{0\,n}$	b_0

Під симплекс-таблицею вважається таблична форма подання математичної моделі в канонічній формі з опорним планом та оцінками оптимальності. Отже, алгоритм симплекс-методу містить такі основні кроки.

1. Вибір роздільного стовпчика j, для якого $a_{0j} < 0$. Нехай $j = k$. Отже, в базис вводиться вектор коефіцієнтів зміної x_k.

2. Виберіть роздільного рядка $i = l, \quad i = 1, m$ за найменшим співвідношенням елементів опорного плану до позитивних елементів роздільного стовпця, тобто $b_l / a_{lk} = \min b_i / a_{ik}$ для всіх $a_{ik} > 0$. Отже, з базиса виключена змінна x_i. Вибрана кількість a_{lk} називається *роздільним елементом*.

3. Перерахунок всіх елементів сімплекс-таблиці відповідно до формул Жордана-Гаусса:

– елементи нового опоорного плану та нове значення цільової функції розраховуються за формулами:

$$b_i = \begin{cases} \dfrac{b_l}{a_{lk}}, & i = l \\ b_i - \dfrac{b_l\, a_{lk}}{a_{lk}}, & i \neq l \end{cases} \qquad (2.23)$$

– інші елементи – за формулами:

$$a_{ij} = \begin{cases} \dfrac{a_{ij}}{a_{lk}}, & i = l \\ a_{lj} - \dfrac{a_{lk}\, a_{lk}}{a_{lk}}, & i \neq l \end{cases} \qquad (2.24)$$

Нова ітерація починається з пункту 1.

У процесі вирішення в п.п. 1 і 2 алгоритму враховуються зазначені 3 ознаки можливих ситуацій. Основний алгоритм симплексного методу може бути доповнений ще 2-ма ознаками.

1. Якщо при виборі роздільного рядка (п. 2) мінімальне співвідношення $\min b_i / a_{ik}\}$ не єдине, тоді новий опорний план є вироджений, тобто одна або більше базисних змінних дорівнюють нулю. У цьому випадку при переході з одного опорного плану до іншого значення цільової функції може залишитися незмінним для декількох ітерацій.

2. Якщо в симплекс-таблиці, що містить оптимальний опорний план, принаймні одна оцінка дорівнює нулю ($a_{0j} = 0$), то оптимальне рішення неоднозначне.

Приклад 2.1. Вирішити симплекс-методом наступну задачу лінійного програмування:

$$\begin{cases} Z = 6x_1 + 2x_2 \to \max; \\ \begin{cases} 4x_1 + 5x_2 \le 61 \\ 4x_2 - 3x_1 \le 24 \\ 5x_1 - 3x_2 \le 30 \end{cases} \\ x_1 \ge 0;\ \ x_2 \ge 0. \end{cases} \tag{2.25}$$

Дайте геометричне уявлення про проблему проблеми.

Рішення. Запишемо канонічну форму початкової задачі:

$$\begin{cases} Z = 6x_1 + 2x_2 \to \max; \\ \begin{cases} 4x_1 + 5x_2 = 61 \\ 4x_2 - 3x_1 = 24 \\ 5x_1 - 3x_2 = 30 \end{cases} \\ x_1 \ge 0;\ \ j = \overline{1, 5}\,, \end{cases} \tag{2.26}$$

де початкове базисне рішення X = (0; 0; 61; 24; 30) є початковим опорним планом.

«Нульове» рівняння $Z - 6x_1 - 2x_2 = 0$ не містить базісних змінних, тому воно є приведеним.

Визначимо розв'язання задачі іншим методом, використовуючи її геометричну інтерпретацію.

Кожна нерівність системи обмежень та умова ненегативності являє собою полуплощину. Перетинання полуплощин утворює опуклу багатокутну множину (множина може бути обмеженою, необмеженою або порожньою).

Кожна вершина багатокутника визначає опорний план. Для початкової математичної моделі нашої задачі область допустимих рішень є обмежений багатокутник (рис 2.1), виділений внутрішньою штриховкою.

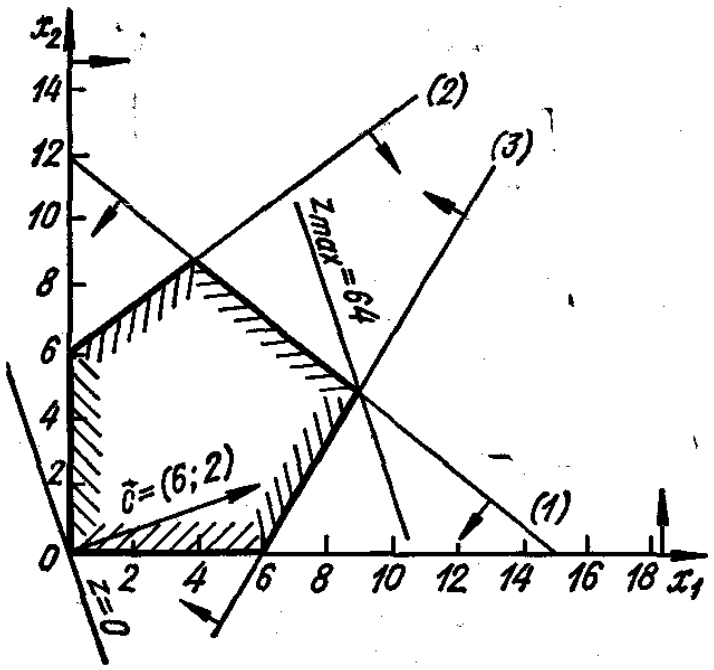


Рисунок 2.1 – Геометрична інтерпретація прямого симплекс-методу

Цільова функція графічно зображена множиною паралельних прямих, званих *лініями рівня*, кожна з яких відповідає певному значенню Z.

Щоб вирішити задачу, лінія рівня зсувається в межах області допустимих рішень (*багатокутника рішень*) у напрямку вектора-градієнта *grad Z = (dz / dx₁; dz / dx₂); x⁰ = (6; 2)* до крайньої точки області. Координати цієї точки й визначають оптимальний план задачі.

В цій задачі максимальне значення функція *Z* приймає в точці перетину прямих (1) и (3):

$$\begin{cases} 4x_1 + 5x_2 = 61 \\ 5x_1 - 3x_2 = 30 \end{cases} \qquad (2.27)$$

Вирішивши цю систему рівнянь, отримаємо координати оптимального рішення: $x_1^* = 9$, $x_2^* = 5$. При цьому максимальне значення цільової функції $Z_{max} = 64$.

Контрольні запитання до розділу 2

1. Що таке лінійне програмування?
2. На які три питання має відповісти побудова матмоделі?
3. Наведіть коротке формулювання завдання виробничого планування.
4. Що таке план виробництва?
5. Сформулюйте цільову функцію завдання виробничого планування.
6. У чому полягає мета розв'язання транспортної задачі?
7. Які величини використовуються при побудові моделі транспортної задачі?
8. Наведіть коротке формулювання задачі про оптимальну суміш.
9. З чого складається матмодель загальної задачі лінійного програмування?
10. Що є результатом розв'язання систем лінійних рівнянь методом Жордана-Гаусса?

3. ДИСКРЕТНЕ ПРОГРАМУВАННЯ

Задачі дискретного програмування відрізняються тим, що на змінні наложено вимогу дискретності, у конкретному випадку – цілочисельності.

Характерні джерела цілочисельності (дискретності):

1. Нероздільність об'єктів, поданих змінними (наприклад, x – кількість працівників або відправлених вагонів).

2. Варіантність типу «так-ні» (наприклад, включати або ні цей пакет у портфель цінних паперів).

3. Заданність можливих значень характеристик нормативними документами (наприклад, перетину дротів, діаметрів труб, розмірів профілів тощо).

4. Комбінаторність (наприклад, розміщення об'єктів, процедура обхода об'єктів, упорядкування).

5. Логічні умови (наприклад, фіксовані витрати виникають лише при виробництві продукції).

Розділяють задачі цілком цілочисельні та частково цілочисельні (змішані). В останніх вимога цілочисельні накладається не на всі змінні.

Будь-яка серія дискретних значень може бути представлена лінійною комбінацією цілочисельних змінних. Тому дискретна задача легко зводиться до цілочисельної за рахунок значного збільшення кількості змінних.

3.1 Проблема цілочисельності

Цілочисельне програмування є найважливішим розділом дискретного програмування. Може здатися, що цілочисельна задача вирішується простіше за безперервну. При малій розмірності та вузьких діапазонах змінних, задачу можна вирішити простим перебіром. В інших випадках потрібні відповідні методи.

Незважаючи на лінійність моделі, допустима множина цілочисельних задач не є опуклою. Так, у повністю цілочисельній задачі вона є множиною окремих точок, що мають цілочисельні координати. Методи лінійного програмування ґрунтуються на випуклості допустимої множини, і тому не можуть безпосередньо застосовуватися до цілочисельних задач.

Звичайно, можна нехтувати вимогою цілочисельності та використовувати один з методів лінійного програмування, але тоді, за рідкісним винятком, результат не буде цілочисельним. Округлення дрібних значень змінних є проблематичним. Дійсно, оскільки оптимальне рішення безперервної задачі полягає у вершині допустимої множини, округлення може призвести до неприпустимості. При 2-х дрібних змінних є 4, а при n змінних – 2^n варіантів округлення! Які з них дають допустимі рішення, можна визначити лише після перевірки всіх обмежень. Слід мати на увазі, що, по-перше, цілочисельна задача може виявитись нерозв'язною, незважаючи на розв'язність безперервної задачі; по-друге, допустимість округленного рішення ще не означає його оптимальність.

У деяких випадках рішення цілочисельної задачі знаходять шляхом вирішення її як безперервної. Так, якщо в оптимальному рішенні безперервної задачі, нецілочисельні значення змінних є великими (їх порядок > 100), округлення до цілих виправдане: можливі порушення умов та відхилення від оптимальності є незначними.

При особливих властивостях цілочисельної задачі рішення її як безперервної завжди дає цілочисельний результат. Цей результат має місце, якщо всі вершини допустимої множини є цілочисельними. Багатогранна множина, яка має цю властивість, називається цілочисельною.

Визначимо умови, за яких множина стає цілочисельною. Візьмемо багатогранну множину *M (B)*:

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \qquad i = \overline{1, m}\,; \tag{3.1}$$

$$x_j \geq 0, \quad j = 1, n, \tag{3.2}$$

де усі a_{ij} – фіксовані цілі числа, $B = (b_1, b_2, \ldots, b_m)^{\mathrm{T}}$ и $m < n$ (ранг матриці $[a_{ij}]$ дорівнює m.) Для нього справедлива наступна теорема.

Теорема. Для того, щоб усі вершини багатогранної множини M(B) при будь-якому цілочисельному векторі B були цілочисельними, необхідно і достатньо, щоб кожний мінор порядка m матриці умов $[a_{ij}]$ дорівнював або 0, або +1 або –1.

Якщо, замість рівності (3.1), множина задана нерівностями, зазначені в теоремі значення відносяться до всіх сінорів матриці $[a_{ij}]$.

Клас задач, що задовольняють теорему, дуже вузький (це транспортні задачі, задачі про призначення та ін.). Такі задачі відносяться до *легко*

вирішуваних (за словами Дж. Едмондса), оскільки для них є *поліномиальні* алгоритми (час і кількість ітерацій зростає поліноміально зі збільшенням розміру задачі). Решта цілочисельних задач входять до класу *складно вирішуваних* задач (клас *NP*, за Карпом і Куком).

Для вирішення таких задач застосовуються різні підходи. З точних методів, можна назвати такі:

1. Метод відсікань.
2. Метод гілок та границь.
3. Побудови послідовності планів.
4. Модифікації динамічного програмування.
5. Послідовного аналізу варіантів.

Усі ці методи, окрім першого, є *комбінаторними*. На додаток до перелічених точних, існує також велика кількість приблизних методів.

3.2 Метод відсікання

Ідею цього методу висказав Джорджем Данцигом. Вона полягає у перетворенні неопуклої множини цілочисельної задачі до опуклої цілочисельної множини шляхом відсікань від опуклої множини безперервної задачі частин, які не містять цілочисельних точок. Тоді використання методів лінійного програмування гарантує отримання оптимального цілочисельного рішення (якщо задача може бути розв’язана).

З цією метою будується опукла оболонка допустимої множини цілочисельної задачі. Опуклою оболонкою неопуклої множини Q називається найменша опукла множина, що містить Q. В цілочисельній задачі вона може бути побудована за допомогою з’єднання крайніх цілочисельних точок допустимої множини *гіперплощинами*. Приклад побудови опуклої оболонки для задачі з 2-ма змінними показано на рисунку 3.1.

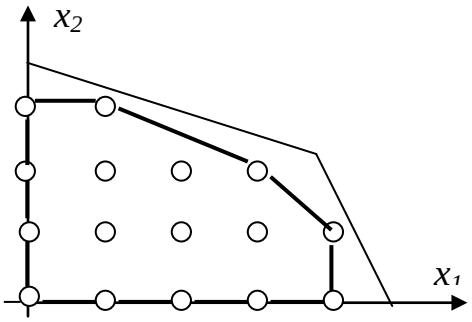


Рис. 3.1. Побудова опуклої оболонки для задачі з 2-ма змінними

Тут, з’єднання крайніх точок безпосередньо дозволило отримати цілочисельну багатогранну множину, яка містить усі допустимі рішення цілочисельної задачі. Без вимоги цілочисельності, допустима множина цієї задачі є опуклим чотирикутником. Як видно, різниця цих множин не містить цілочисельних рішень.

Геометрично все виглядає досить простим. Але тривалий час було неможливо формалізувати процедуру побудови цілочисельної множини. Першим це зміг зробити американський математик Ральф Гоморі. Він запропонував ітеративну процедуру, згідно з якою на кожній ітерації відсікається частина множини безперервної задачі (БЗ), яка не містить цілочисельних рішень, але включає в себе оптимальне рішення БЗ, і на скороченій безперервній множині знаходиться нове оптимальне рішення одним з методів лінійного програмування. Ітерації закінчуються, коли оптимальне рішення наступної БЗ виявляється цілочисельним або виявляється нерозв’язанність і БЗ, і цілочисельної задачі. У цьому випадку випукла оболонка може бути побудована лише частково.

Розглянемо приклад (рис. 3.2). Оптимальне рішення БЗ як за критерієм L_1, так і за L_2 знаходиться у вершині A.

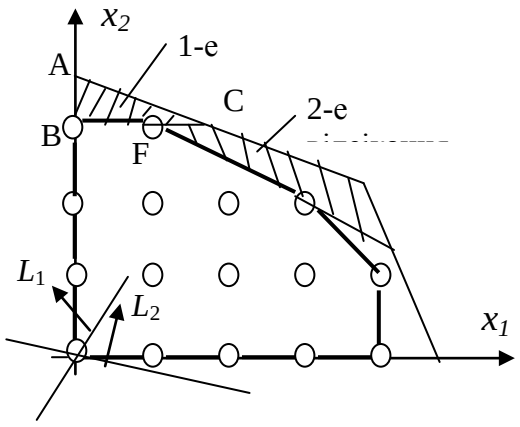


Рис. 3.2. Ітераційна процедура відсікання Гоморі

Після першого відсікання нецілочисельної частини множини, що містить точку A, з'являється цілочисельна вершина B. При вирішенні задачі згідно з критерієм L_1, в ній буде оптимум БЗ, а отже, й початкової цілочисельної задачі. Якщо ж взяти критерій L_2, то оптимум БЗ опиниться у вершині C, яка не є цілочисельною. Тому треба ще одне відсікання, після чого буде отримано

оптимальне цілочисельне рішення в точці *F*. В обох випадках, опукла оболонка будується лише частково.

Проблема полягала в отриманні регулярної умови, приєднання якого до обмежень БЗ призводить до необхідного відсікання. Ця умова повинна відповідати наступним вимогам:

1) не виконуватись в поточному оптимальному рішенні БЗ;

2) виконуватись у всіх допустимих цілочисельних рішеннях.

Перша вимога забезпечує відсікання частини безперервної множини, друга – незмінність цілочисельної множини.

Гоморі запропонував кілька варіантів отримання умов відсікання. Ми покажемо виведення умови відсікання, яке використовується в 1-му алгоритмі Гоморі. Нехай отримано оптимальне рішення БЗ. Рівняння, що відповідає рядку оптимальної симплекс-таблиці з *i*-ю базисною змінною, записується наступним чином:

$$x_i^* + \sum_{j \in небаз} \alpha_{ij} \cdot x_j^* = \alpha_{i0}, \qquad (3.3)$$

де α_{i0} – значення базисної змінної x_i^* (з стовпця A_0), α_{ij} – коефіцієнти при небазисних змінних (з стовпців A_j).

Нас цікавлять в змінні, які мають нецілі значення у отриманому оптимальному рішенні. У цих випадках коефіцієнт α_{i0} в (3.3), природньо, не цілий, а коефіцієнти α_{ij} можуть бути будь-якими дійсними числами.

Нецілe значення представимо у вигляді цілих і дрібних частин. Ціла частина числа є найбільшим цілим числом, яке не перевищує самого числа. Будемо позначати цілу частину символами $\lfloor \ldots \rfloor$.

Наприклад, $\lfloor 2.1 \rfloor = 2, \lfloor 1.9 \rfloor = 1, \lfloor -2.1 \rfloor = -3$.

Тоді для дрібної частини коефіцієнтів в (3.3) маємо $\hat{\alpha}_{ij} = \alpha_{ij} - \lfloor \alpha_{ij} \rfloor$, отже, для нецілого α_{ij} завжди $0 < \hat{\alpha}_{ij} < 1$. Перепишемо (3.3) з виділеними цілими і дрібними частинами коефіцієнтів:

$$x_i^* + \sum_{j \in небаз} \left(\lfloor \alpha_{ij} \rfloor + \hat{\alpha}_{ij} \right) x_i^* = \lfloor \alpha_{i0} \rfloor + \hat{\alpha}_{i0}. \qquad (3.4)$$

Залишимо в лівій частині (3.4) тільки цілі частини коефіцієнтів. Тоді, враховуючи ненегативність $\hat{\alpha}_{ij}$ и x_j^*, отримаємо нерівність:

$$x_i^* + \sum_{j \in небаз} \lfloor \alpha_{ij} \rfloor x_j^* \le \lfloor \alpha_{i0} \rfloor + \hat{\alpha}_{i0}. \qquad (3.5)$$

Тепер використаємо умову цілочисельності. При цілих змінних ліва частина нерівності (3.5) може приймати лише цілі значення. Тому, якщо відкинути дрбову частину, несувора нерівність лівої та правої частин збережеться:

$$x_i^* + \sum_{j \in небаз} \lfloor \alpha_{ij} \rfloor x_j^* \le \lfloor \alpha_{i0} \rfloor. \qquad (3.6)$$

Примітка: В цьому легко впевнитись на простих прикладах:

$$9 \le 10{,}3;\ 9 \le 9{,}75.$$

Віднімаючи (3.6) від рівняння (3.3), отримаємо:

$$\sum_{j \in небаз} \hat{\alpha}_{ij}\, x_j^* \ge \hat{\alpha}_{i0}. \qquad (3.7)$$

Це шукана умова відсікання. Дійсно, у оптимальному рішенні БЗ (як у будь-якому базисному) небазисні змінні дорівнюють нулю, а $\hat{\alpha}_{i0} > 0$, тому нерівність (3.7) в ньому не виконується. Тому додавання (3.7) до початкових умов БЗ призведе до звуження допустимої множити за рахунок відсікання його частини з оптимальною вершиною.

З багатьма нецільовими змінними виникає питання вибору змінної, на якій слід побудувати відсікання. Немає жорсткого обґрунтування для вибору. На підставі експериментальних даних рекомендується взяти змінну з найбільшою дрібною частиною.

Друге питання стосується способу врахування чергової умови відсікання: його можна додати до умов початкової задачі та розв’язати задачу знову або, після додавання, продовжити симплекс-перетворення з отриманого оптимального рішення, яке стало неприйнятним. У алгоритмі Гоморі другий варіант застосовується як більш економічний.

Перед додаванням умова відсікання зводиться до рівняння:

$$\sum_{j \in небаз} \hat{\alpha}_{ij} \cdot x_j^* - x_{\bar{n}+1} = \hat{\alpha}_{i0}. \qquad (3.8)$$

Тому що небазисні змінні дорівнюють нулю, нова додаткова змінна $x_{\bar{n}+1} = -\alpha_{i0} < 0$. Тому рекомендується для подальшого рішення застосовувати подвійний симплекс-метод.

Згідно з першим алгоритмом Гоморі, необхідно виконати наступні дії:

1. Перетворити умови задачі так, щоб всі коефіцієнти стали цілими числами.

2. Вирішіти початкову задачу без урахування цілочисельності (БЗ) одним з методів лінійного програмування. Якщо безперервна задача нерозв'язна, то зафіксувати нерозв'язнаість початкової задачі і перейти до кроку 5.

3. Перевірити рішення на цілочисленість: якщо рішення цілочислене, то зафіксувати оптимальне рішення та перейти до кроку 5.

4. Якщо не всі змінні цілі числа, то з оптимальної таблиці вибрати змінну з найбільшою дрібною частиною.

5. Виписати з симплекс-таблиці рядок з вибраною базисною змінною.

6. Виділити дрібні частини коефіцієнтів у отриманому рівнянні та записати умову відсікання відповідно до (3.7).

7. Привести умову відсікання до рівняння (3.8), помножити його на –1 та додати отриманий рядок до оптимальної симплекс-таблиці. При цьому розмірність базиса збільшується на одиницю. Додаткова змінна з нового рядка приймається як відсутня базисна змінна.

8. Вирішити розширену задачу подвійним симплекс-методом. Якщо задача розв'язна, перейти до кроку 3. Інакше зафіксувати нерозв'язність цілочисленої задачі.

Гоморі виявив конвергенцію цього алгоритму. Але алгоритм сходиться за кінцеву кількість кроків, оцінки необхідної кількості ітерацій не існує. Швидкість конвергенції алгоритму дуже низька. Наприклад, при 10–15 змінних для отримання рішення може бути потрібно більше тисячі ітерацій.

Відмічається залежність швидкості конвергенції від форми та порядку запису умов задачі, значний вплив помилок округлення. Неприємна сторона алгоритму є постійне збільшення кількості рядків до кількості змінних у канонічній формі. Якщо задача розв'язна, алгоритм знаходить цілочисельне рішення лише на останній ітерації. Тому, у разі переривання розрахунку, не буде отримано жодне, навіть прийнятне рішення.

Ефективність алгоритму збільшується зі зменшенням значень a_{ij} і b_i та заповненості матриці умов A. Можна рекомендувати використання методу для задач невеликого виміру (до десятків змінних), коли значення a_{ij} і b_i невеликі та в оптимальному рішенні безперервної задачі більшість змінних мають цілі значення. Алгоритм мало підходить для вирішення комбінаторних задач у цілочисельній постановці.

3.3 Метод гілок та границь

Початок розробки підходу, який називався методом гілок та границь, був закладений роботою Ленда, на підставі якої почали розробляти алгоритми вирішення цілочислених задач різного характеру.

Метод полягає в тому, щоб побудувати дерево задач, коренем якого є початкова проблема, можливо, без умови цілочисельності (тобто безперервна задача). Нижні задачі створюються верхніми так, що їх допустимі множини є не пересічними підмножинами верхньої задачі. Зростання дерева відбувається завдяки перспективним гілкам. Перспективність визначається оцінкою критерію термінальної задачі гілки V та рекорду Z.

Оцінка V – це значення критерію, свідомо не гірше, ніж оптимальне, а Z – значення критерію початкової задачі, досягнуте під час рішення (початкове значення може бути свідомо гірше, ніж оптимальне). Отже, задача буде генеруючою лише за умови, що її оцінка краща, ніж рекорд. У цьому випадку рівень задачі не має значення.

Розглянемо метод застосовно до лінійної цілочисленної задачі. Незважаючи на те, що не існує жодних обмежень щодо кількості задач, безпосередньо створених перспективною, в алгоритмах, як правило, використовується розбиття на 2 задачі, тобто будується бінарне дерево (рис. 3.3).

При цьому для целочисельних множин виконуються співвідношення:

$$\bigcup_i D_i = D, \quad D_i \bigcap_i D_j = \varnothing, \quad i \neq j. \tag{3.9}$$

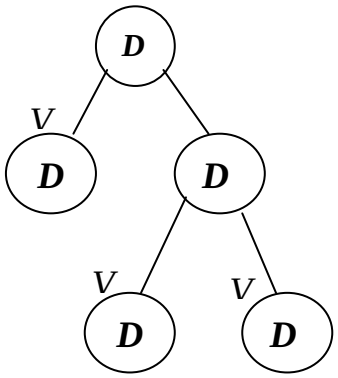


Рис. 3.3. Бінарне дерево

Очевидно, якщо, наприклад, оцінка V_{22} буде гіршою, ніж рекорд або $D_{22} = \varnothing$, права гілка обривається (також кажуть, що вона прозондована). Якщо оцінка V_{22} краще, ніж Z, виконується розгалуження: множина D_{22} ділиться на 2 підмножини. Рішення буде завершено, коли всі гілки будуть прозондовані.

Вид оцінки залежить від направленності критерію: при максимізації використовується верхня оцінка, при мінімізації – нижня. Подальше викладення методу буде відноситися до задачі на максимум.

Для алгоритмічної реалізації схеми гілок та границь необхідно вирішити 2 фундаментальні питання:

1. Як розбити перспективну множину на підмножині.

2. Як визначити верхню оцінку критерію на встановленій множині.

Відповіді на них залежать від типу задачі (частково або повністю цілочисельна, має спеціальні властивості чи ні, з булевими або небулевими змінними). Нижче розглядається загальний випадок.

Нехай відомий діапазон можливих значень j-ї змінної $0 \le x_j \le d_j$, яка у безперервному оптимальному рішенні виявилася нецілочисельною рівною x_j *. Тоді цілочисельне значення цієї змінної може бути досягнуто або в інтервалі $0 \le x_j \le \lfloor x_j^* \rfloor$, або в інтервалі $\lfloor x_j^* \rfloor + 1 \le x_j \le d_j$, де $\lfloor x_j^* \rfloor$ – ціла частина x_j^* (рис. 3.4).

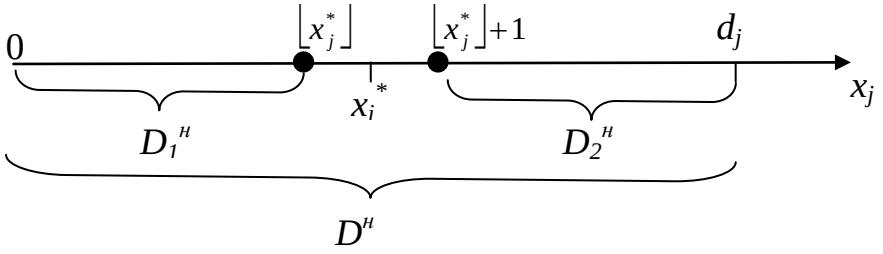


Рис. 3.4. Розбиття множин

Це відповідає розбиттю безперервної множини $D^н$ на 2 множини $D_1^н$ і $D_2^н$, що не перетинаються і об'єднання яких не дорівнює $D^н$. В той же час таке розбиття цілочисельного множества задовольняє співвідношенням (3.9). При цьому цілочисельні множини, як початкова, так і генеровані, додаються до відповідних безперервних множин. Отже, пошук цілочисельного рішення на безперервній множині дає той же результат, що і на цілочисельній. Легко побачити, що приведене виділення підінтервалів за одною змінною призводить до розбиття початкової множини на 2 підмножини при будь-якій кількості змінних.

Тепер ми звернемося до другого питання. Оскільки цілочисельна множина – це підмножина відповідної безперервної, оптимальне значення критерію на безперервній множині завжди буде не менше, ніж на цілочисельній. Тому, як верхню оцінку V можна взяти оптимальне значення критерію L^* безперервної задачі.

Вибір початкового значення рекорду залежить від ситуації:

• якщо відоме будь-яке цілочисельне значення, то рекорд приймається рівним критерію в цьому рішенні;

• при позитивності усіх коефіцієнтів критерію можна взяти нульове значення рекорду;

• в інших випадках як початкове значення рекорду приймають M – максимальне число, що його можна представити в комп'ютері.

У ході розбиття формуються генеровані задачі, які розміщуються у *переліку задач*. Початковий перелік містить лише 1 задачу – початкову задачу без умови цілочисельності. І вподальшому, перелік буде містити лише безперервні задачі.

Таким чином, базовий алгоритм, який реалізує метод гілок та границь, включає наступні кроки.

1. Встановлюється початкове значення рекорду, і в переліку задач розміщується початкова задача без вимоги цілочисельності змінних.

2. Перелік задач аналізується: якщо він порожній, то переходять до кроку 6. В іншому випадку обирається одна з задач з видаленням її з переліку.

3. Вибрана задача вирішується одним із методів лінійного програмування. Якщо задача нерозв'язна або оптимальне значення критерію $L^* \le Z$, гілка обривається (задача прозондована). Перехід до кроку 2.

4. Отримане рішення аналізується на цілочисленість. Якщо рішення цілочислене, воно фіксується, рекорду призначається оптимальне значення критерію вирішеної безперервної задачі (Z: = L*), гілка обривається і здійснюється перехід до кроку 2.

5. Обирається одна із змінних, що мають нецілочисельні значення. За нею здійснюється розгалуження: створюються 2 задачі, одна утворюється шляхом підключення до вирішуваної (батьківської) задачі умови $x_j \le \lfloor x_j^* \rfloor$, інша – додаванням до батьківської задачі обмеження $x_j \ge \lfloor x_j^* \rfloor + 1$. Ці задачі записуються у переліку задач. Переходять до кроку 2.

6. Виведення результатів (якщо значення рекорду перевищує початкове, то отримано оптимальне рішення початкової задачі, інакше вона нерозв'язна).

Наведений вище алгоритм є базовим, оскільки він не включає однозначні правила вибору задач із переліку та розгалуженої змінної. Для частково цілочисельних задач при виборі змінної для розгалуження виключаються безперервні змінні.

Кількість вирішених задач істотно залежить від вибору задачі з переліку та змінної для розгалуження.

З алгоритму випливає, що гілка обривається з одної з 3-х причин:

1. Нерозв'язність задачі.
2. Задача має цілочисельне рішення.
3. Верхня оцінка не більше рекорду.

Метод гілок та границь має такі переваги у порівнянні з методом відсікання:

– накопичення помилок менш значуще, оскільки процес рішення йде по різних гілках;

– при примусовій зупинці процесу рішення є висока ймовірність отримання цілочисельного результату, але без встановення його оптимальності;

– при вирішенні безперервних задач розміри симплекс-таблиць не збільшуються.

Недоліки методу гілок та границь:

– не можна оцінити кількість задач, які потрібно буде вирішити. Чим ближче знизу початкове значення рекорду та згори оцінка критерію задачі до шуканого оптимального значення критерію, тим менше вершин матиме дерево

рішень, а отже, й витрата ресурсів. Однак, завжди повинно бути на увазі, що переоцінка початкового рекорду може призвести до нерозв'язності задачі;

– відсутність ознаки оптимальності. Оптимальне рішення можна отримати задовго до зупинки алгоритму, але, як правило, заздалегіть це виявити неможливо. Оптимальність встановлюється лише після вичерпання усього переліку задач.

Очевидно, що ефективність методу збільшується із зменшенням діапазонів значень змінних та кількості нецілих змінних у рішенні початкової безперервної задачі.

3.4 Адитивний алгоритм

Адвійний алгоритм розроблений у зв'язку з задачами з булевими змінними. Він виконує лише операції додавання та віднімання (звідси й назва методу). Тому накопичення помилок не відбувається. Алгоритм являє собою реалізацію одного з методів часткового перебору. Його також можна розглядати як особливий випадок методу гілок та границь.

Модель задачі повинна бути представлена у стандартній формі. Далі ми розглянемо алгоритм у відношенні до наступної задачі:

$$L = \sum_{j=1}^{n} c_j x_j \to \min; \tag{3.10}$$

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i, \quad i = \overline{1, m}; \tag{3.11}$$

$$\forall x_j = \begin{vmatrix} 0 \\ 1 \end{vmatrix}. \tag{3.12}$$

При цьому $\forall c_j \ge 0$, що означає виконання ознаки оптимальності симплекс-методу в початковому рішенні (у задачах на мінімум).

Дійсно, оскільки коефіцієнти додаткових змінних $c_i = 0$, то $z_j = 0$ і $\Delta_j = z_j - c_j \le 0$. Тому, якщо ще й $\forall b_i \ge 0$, то одразу маємо оптимальне рішення задачі: всі n початкові змінні та критерій дорівнюють нулю. Однак, як правило, не всі b_i позитивні і нульове початкове рішення виявляється неприйнятним.

Якщо у критеріях є негативні коефіцієнти, то модель перетворюється: змінні x_k с $c_k < 0$ замінюються по всій моделі на $x_k = 1 - x_k'$, а утворена в критерії постійна відкидається (після отримання рішення вона додається до оптимального значення критерію). Рівності та нерівності ≥ перетворюються на нерівності ≤. Таким чином, будь-яка початкова модель може бути приведена до виду (3.10)–(3.12) з $\forall c_j \geq 0$.

Представимо умови (3.11) в канонічній формі:

$$\sum_{j=1}^{n} a_{ij} x_j + S_i = b_i, \quad i = \overline{1, m}, \tag{3.13}$$

де S_i – додаткові змінні. Тоді початкове рішення, очевидно: $\forall x_j = 0$ й $S_i = b_i$.

Як вже було сказано, якщо $\forall S_i \geq 0$, воно оптимальне. В іншому випадку здійснюється частковий перебір рішення. Щоб пояснити алгоритм, використовуємо таблицю. 3.1, що аналогічна симплексній.

Таблиця 3.1 – Адитивний алгоритм

A_0 (рішення)	x_1	x_2	. . .	x_n
b_1	a_{11}	a_{12}	. . .	a_{1n}
b_2	a_{21}	a_{22}	. . .	a_{2n}
. . .	. . .	. . .	. . .	. . .
b_m	a_{m1}	a_{m2}	. . .	a_{mn}
L	c_1	c_2	. . .	c_n

У алгоритмі змінні поділяються на фіксовані і вільні. Змінна, якій призначено певну величину, називається фіксованою. Значення вільної змінної можна змінити. Основним об'єктом алгоритму є *часткове рішення* – це рішення, в якому частина змінних фіксована. Воно описується впорядкованою множиною індексів фіксованих змінних. У цьому випадку індекс змінних, що дорівнюють нулю, записується зі знаком мінус. Наприклад, якщо на *t*-й ітерації фіксовані $x_2 = 0$, $x_4 = 1$, то часткове рішення представляється як множина $I_t = \{-2, 4\}$.

Початкова часткова множина завжди порожня ($I_0 = \varnothing$), а значення рекорду $z = \infty$. Алгоритм складається з 4-х перевірок, які виконуються для того, щоб визначити наявність перспективних вільних змінних. Якщо така змінна знаходиться, то змінивши її значення, можна покращити результат. За

замовчуванням прийнято, що вільні змінні знаходяться на найнижчому рівні (дорівнюють нулю). Часткове рішення вважається *0*, якщо воно не може привести до допустимого рішення та зменшення значення критерію.

Нехай маємо часткове рішення I_t з критерієм L^t і вектором додаткових змінних $S^t = (S_1^t, S_2^t, ..., S_m^t)$. До нього застосовуються наступні перевірки.

1. Для кожної вільної змінної x_r коефіцієнти a_{ir} перевіряються в рядках $S_i^t < 0$ (табл. 3.7). Якщо у всіх таких рядках $a_{ir} \geq 0$, змінна x_r виключена, оскільки зміна її значення від 0 до 1 не призведе до позитивності щонайменше однієї з розглянутих S_i^t.

2. Аналізується можливість поліпшення критерію. Якщо для вільної змінної x_r виконується нерівність $C_r + L^t \geq z$, зміна її значення не може призвести до зменшення рекорду. Тому вона виключється.

Вільні змінні, що залишаються після цих перевірок, утворюють множину P_t. Якщо вона порожня, то наявне часткове рішення не є перспективним, тобто воно вважається прозондованим.

3. Виявляється можливість отримання допустимого рішення на основі цього часткового. У рядках з $S_i^t < 0$ перевіряється умова

$$\sum_{j \in P_t} \min(0, a_{ij}) > S_i^t. \tag{3.14}$$

Якщо вона виконується, принаймні, для одного рядка, всі змінні з P_t виключаються, оскільки значення навіть всіх цих змінних від 0 до 1 не забезпечить допустимості рішення (ненегативності вектору S). У цьому випадку рішення I_t вважається прозондованим (гілка обривається).

Якщо умови (3.14) не виконуються, здійснюється перевірка 4.

4. Коли $P_t \neq \varnothing$, то гілка продовжується. Для отримання нового часткового рішення з I_t розраховуються оцінки кожної змінної з P_t:

$$v_j^t = \sum_i \min(0, S_i^t - a_{ij}). \tag{3.15}$$

Оцінка дає сумарну величину недопустимості, що залишається після зміни значення змінної $x_j \in P_t$ з 0 на 1. Як видно з (3.15), негативна оцінка вказує на наявність недопустимості.

З отриманих оцінок визначається максимальна:

$$\max_j v_j^t = v_k^t. \tag{3.16}$$

Очевидно, якщо $v_k^t = 0$, то зміна ХК С з 0 на 1 дає допустиме рішення з меншою величиною критерію. Отже, рекорду *z* призначається значення $L^t + C_k$, а нове часткове рішення $I_{t+1} = \{I_t, k\}$ вважається прозондованим.

Якщо ж $v_k^t < 0$, то допустиме рішення не досягнуте і часткове його рішення $I_{t+1} = \{I_t, k\}$ піддається усім перевіркам. Якщо внаслідок перевірки воно виявиться прозондованим, нове часткове рішення отримують з I_{t+1}, змінюючи знак індексу введеної змінної: $I_{t+2} = \{I_t, -k\}$, тобто фіксацією x_k із значенням 0.

У загальному випадку, прозондоване часткове рішення може містити позитивні та негативні індекси. Для отримання нового часткового рішення змінюють знак самого правого позитивного індексу, а індекси, що йдуть за ним, відкидають. Отже, з рішення {2, –1, –3, 5, –7, –6} слідує часткове рішення {2, –1, –3, –5}. Якщо уявити весь процес рішення у вигляді дерева (аналогічно методу гілок та границь), то відкидання *l* останніх індексів означає повернення на *l* рівней угору. Умовою завершення роботи Адитивного алгоритму є відсутність позитивних індексів у частковому рішенні.

3.5 Інші методи

Клас точних методів також включає метод побудови послідовності планів. В цьому методі початкова лінійна цілочислена задача з *m* умовами замінюється задачею з однією умовою (розширена задача). Для визначення цього стану вирішуються *m* задач, кожна з однією умовою. Знаходиться мінімум з *m* оптимальних значень критеріїв $\min L_i^* = L_t^*$. Задача *t* приймається як розширена (допустима множина початкової задачі є її підмножина). Якщо оптимальний план цієї задачі не є допустимим для початкової, він видаляється з допустимого набору розширеної задачі, і вона вирішується знову. Новий оптимальний план знову перевіряється на допустимість та видаляється при недопустимості.

Так будується послідовність незростаючих (за критеріями) планів. Процес рішення завершується, як тільки наступний план розширеної задачі виявляється допустимим для початкової, і тому він буде її оптимальним рішенням. Для загальної задачі цілочисленого програмування рішення розширеної рекомендується шукати методом динамічного програмування.

Точні методи цілочисленого програмування відрізняються високою працемісткістю, причому із збільшенням кількості змінних вона зростає

експоненціально. Тому вони використовуються для задач середнього виміру. Практично вирішують загальні задачі з кількома сотнями змінних, а також спеціальні задачі (особливі властивості, булеві змінні) – з тисячами змінних. При цьому у більш «потужних» алгоритмах метод гілок та границь поєднується з методом відсікання. Останній використовується при вирішенні підзадач як спосіб уточнення верхньої оцінки: відсікання здійснюється лише до зміни значення критерію.

Для вирішення завдань середнього та великого виміру використовуються приблизні методи. Клас приблизних методів дуже великий. В багатьох з них використовуються ідеї точних методів, модифіковані для пошуку приблизного рішення. Значна група складається з методів на основі евристики – правил, які не мають суворого обґрунтування, але відповідають за здоровий глузд (представлення щодо властивостей оптимального рішення з урахуванням специфіки задачі). Такі методи працюють швидко, але неможливо оцінити якість отриманого рішення. Як правило, знаходиться локальне рішення.

У методі вектора рецесії локальність рішення перевіряється безпосередньо. Рецесія мінімізованого критерію – це векторна функція від x відносно окружності радіусу $r > 0$ (замкнута куля) з центром у точці x. Розмірність вектора рецесії дорівнює кількості цілочисельних точок в цій кулі без x. Його компоненти характеризують поведінку критерію при відхиленні від x. Вони схожі на відносні оцінки в симплекс-методі, але розраховуються для цілочисельних точок в кулі.

Якщо всі компоненти вектора негативні, то поточна точка x – це локальний мінімум. У цьому випадку радіус кулі збільшується, а вектор рецесії перераховується. Послідовність радіусів $r_1 < r_2 < \ldots < r_m$ та початкова точка визначаються апріорі. Якщо якийсь компонент показує поліпшення рішення, то здійснюють перехід до кращої точки, яка приймається за поточну, і вектор рецесії розраховується відносно цієї точки для кулі, без зміни радіусу. Якщо встановлено, що досліджувана точка є локальним мінімумом на кулі з максимальним радіусом, вона приймається за рішення задачі. Характеристики цього методу дуже залежать від способу розрахунку вектора рецесії та попередньо відібраних значень параметрів алгоритму.

На додаток до перелічених підходів, є приблизні методи, які використовують випадковий пошук та особливі генетичні алгоритми. Останні поєднують випадкові дії з детермінованими.

Контрольні запитання до розділу 3

1. Назвіть характерні джерела цілісності у задачах дискретного програмування.
2. Які точні методи застосовуються для складних задач дискретного програмування?
3. У чому полягає ідея методу відсікань розв'язання задач дискретного програмування?
4. У чому полягає ітераційна процедура, запропонована Р. Гоморі?
5. Яким двом вимогам має відповідати регулярна умова?
6. Наведіть послідовність дій згідно з першим алгоритмом Гоморі.
7. У чому полягає метод гілок та границь?
8. Які два основні питання необхідно вирішити для алгоритмічної реалізації схеми гілок та границь.
9. Охарактеризуйте три ситуації, від яких залежить вибір початкового значення рекорду у методі гілок та границь.
10. У чому полягає ідея адитивного алгоритму розв'язання задач дискретного програмування.

4. НЕЛІНІЙНЕ ПРОГРАМУВАННЯ

4.1. Характеристика задач нелінійного програмування

Методи нелінійного програмування використовуються для вирішення задач з нелінійними змінними функціями. У загальному випадку задача математичного програмування записується так:

$$\left.\begin{aligned} & f(\mathbf{x}) \to \max(\min); \\ & \varphi_i(\mathbf{x}) \le (\ge) 0, \qquad i = \overline{1, m_1}; \\ & \psi_k(\mathbf{x}) = 0, \qquad k = \overline{1, m_2}. \end{aligned}\right\} \tag{4.1}$$

Якщо принаймні одна функція в моделі (4.1) нелінійна, це задача нелінійного програмування (НП). Розмірність задачі характеризується розмірністю вектору змінних n та кількістю умов $m_1 + m_2$. Однак складність задачі визначається не стільки її розмірністю, скільки властивостями функцій цілей та обмежень.

Різноманітність задач НП дуже велика. Універсальних методів вирішення таких задач не існує. Дуже обмежена кількість точних методів і набагато більше приблизних.

Найбільш розвинені методи вирішення задач *опуклого програмування*. Цей клас включає задачі НП з опуклою допустимою множиною та опуклою цільовою функцією при мінімізації або увігнутою – при максимізації. Допустима множина опукла, якщо всі функції ψ_k є лінійними, а функції φ_i – опуклими при нерівності $\le$ або увігнуті при $\ge$. Наприклад, умова $x_1^2 + x_2^2 \le r^2$ породжує опуклу множину, перетин якої з прямою $x_1 + x_2 = 0$ дає теж опуклу множину. Очевидно, задачі НП пов'язані з цим класом.

Основна особливість задач опуклого програмування полягає в тому, що вони *унімодальні*, тобто будь-який з їх локальних оптимумів є глобальним. Для низки задач опуклого програмування з диференційованими функціями розроблені точні методи. Найбільші труднощі виникають при вирішенні багатоекстремальних задач, які, за визначенням, не належать до класу опуклих.

Важливим класом НП є задачі *квадратичного програмування*. В них цільова функція – це сума лінійних та квадратичних форм, а усі умови лінійні.

У випадку випуклості (ввігнутості) квадратичної форми, вони є особливим випадком задач випуклого програмування.

У нелінійному програмуванні виділяють також задачі *сепарабельного програмування*. Це задачі, в яких всі функції є сепарабельні. Функція сепарабельна, якщо вона представлена як сума функцій окремих змінних. Лінійна функція – це окремий випадок сепарабельності. Сепарабельна задача може бути одночасно і задачею опуклого програмування.

Задачі *геометричного програмування* складають окремий клас НП. Всі функції в таких задачах є *поліномами*. Загалом, поліномом називається функція виду:

$$f(\mathbf{x}) = \sum_{k=1}^{l} C_л \cdot x_1^{\alpha_{k1}} \cdot x_2^{\alpha_{k2}} \cdot \ldots \cdot x_n^{\alpha_{kn}}, \quad \forall C_k > 0,$$

в якій α_{kj} – будь-які дійсні числа.

Задачі геометричного програмування ставляться тільки на мінімум:

$$\left.\begin{array}{l} f(\mathbf{x}) \to \min, \\ \varphi_i(\mathbf{x}) \le b_i, \\ \mathbf{x} > 0. \end{array}\right\}$$

Такі задачі частіше виникають в конструкторських розробках. Для них розроблені спеціальні методи.

Кусочно-лінійне програмування включає спеціальні методи вирішення задач з кусочно-лінійними функциями. Зокрема, це функції:

$$f(\mathbf{x}) = \max_i f_i(\mathbf{x}) \quad \text{і} \quad f(\mathbf{x}) = \min_i f_i(\mathbf{x}),$$

якщо усі $f_i(X)$ – лінійні функції. Перша з них – опукла, друга – увігнута. Задачі з такими функціями можуть бути включені до класу опуклих задач програмування. Їх рішення базується на перетворенні моделі до лінійного вигляду з подальшим використанням методів лінійного програмування (ЛП).

До лінійних також зводяться задачі *дробово-лінійного програмування*. Вони відрізняються від лінійних лише дробовою цільовою функцією, чисельник та знаменник якої – лінійні функції.

4.2. Умови оптимальності

Важлива властивість задач НП – це диференційованість функцій критерію та обмежень. Для таких задач отримані умови оптимальності, на основі яких побудована низка методів НП.

Нехай задача має вигляд:

$$\left.\begin{array}{c} f(\mathbf{x}) \to \max; \\ \varphi_i(\mathbf{x}) \ge 0, \quad i = \overline{1, m_1}; \\ \psi_k(\mathbf{x}) = 0, \quad k = \overline{1, m_2}; \\ \mathbf{x} \ge 0. \end{array}\right\} \tag{4.2}$$

Узагальнений метод множників Лагранжа застосовується й до умов-нерівностей. Запишемо функцію Лагранжа (регулярну) для задачі (4.2):

$$F(\mathbf{X}, \mathbf{\Lambda}) = f(\mathbf{X}) + \sum_{i=1}^{m_1} \lambda_i \varphi_i(\mathbf{X}) + \sum_{k=1}^{m_2} \lambda_k \psi_k(\mathbf{X}). \tag{4.3}$$

В теорії НП показано, що ця функція має сідлову точку ($\mathbf{X}^*$, $\mathbf{\Lambda}^*$) з максимумом по $\mathbf{X}$ та мінімумом по $\mathbf{\Lambda}$:

$$F(\mathbf{X}, \mathbf{\Lambda}^*) \le F(\mathbf{X}^*, \mathbf{\Lambda}^*) \le F(\mathbf{X}^*, \mathbf{\Lambda}). \tag{4.4}$$

Тому задача (4.2) зводиться до пошуку сідлової точки функції (4.3).

4.3. Квадратичне програмування

Задачі квадратичного програмування (КП) виникають, якщо цільова функція є сумою лінійних та квадратичних форм, а усі умови лінійні.

Наприклад, у задачі з 2-ма змінними цільова квадратична функція записується таким чином:

$$f(x_1, x_2) = \boxed{d_1 x_1 + d_2 x_2} + \boxed{\tfrac{1}{2}(C_{11} x_1^2 + C_{12} x_1 x_2 + C_{21} x_1 x_2 + C_{22} x_2^2)}.$$

лінійна форма — *квадратична форма*

У векторній формі ця функція прийнимає вигляд:

$$f(x_1, x_2) = [d_1, d_2]\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \frac{1}{2} \cdot [x_1, x_2]\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

Узагальнюючи на випадок багатьох змінних, отримуємо:

$$f(\mathbf{X}) = \mathbf{D^T} \cdot \mathbf{X} + \frac{1}{2} \cdot \mathbf{X^T} \cdot \mathbf{C} \cdot \mathbf{X};$$

$$\mathbf{D} = \begin{bmatrix} d_1 \\ d_2 \\ \dots \\ d_n \end{bmatrix}; \qquad \mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}; \qquad \mathbf{C} = \begin{bmatrix} C_{11} & C_{12} & \dots & C_{1n} \\ C_{21} & C_{22} & \dots & C_{2n} \\ \dots & \dots & \dots & \dots \\ C_{n1} & C_{n2} & \dots & C_{nn} \end{bmatrix}.$$

Матриця **C** – квадратна, діагонально-симетрична ($C_{ij}=C_{ji}$).

В цілому, задачу квадратичного програмування ставлять в вигляді:

$$f(\mathbf{X}) = \mathbf{D^T} \cdot \mathbf{X} + \frac{1}{2} \cdot \mathbf{X^T} \cdot \mathbf{C} \cdot \mathbf{X} \to \max; \tag{4.10}$$

$$\mathbf{B} - \mathbf{AX} \geq 0; \tag{4.11}$$

$$\mathbf{X} \geq 0. \tag{4.12}$$

Щоб зробити її задачею випуклого програмування, цільова функція (4.10) повинна бути увігнутою.

Властивості функції визначаються матрицею С. Для увігнутості функції необхідно, щоб матриця С була негативно визначена (сувора увігнутість) або негативно напіввизначеною. Матриця С є негативно визначеною, якщо для всіх, ненульових Х є дійсною умова X^T С Х < 0, і негативно напівизначеною, якщо X^T С Х ≤ 0. У разі мінімізації, цільова функція повинна бути опуклою, що має місце при позитивно визначеній або позитивно напівизначеній матриці С. Практично визначити властивість квадратичної функції можна за допомогою достатніх умов екстремуму: якщо функція в стаціонарній точці має максимум, вона увігнута, а якщо мінімум, то опукла.

Очевидно, що розглянутий спосіб знаходить за кінцеву кількість кроків глобальне рішення задачі КП з увігнутою цільовою функцією. При суворій увігнутості задача має одне рішення, при несуворій увігнутості можливі багато рішень. Якщо функція не увігнута, метод знаходить якийсь локальний максимум.

Залача КП може бути використана як апроксимація при нелінійності критерію, відмінній від квадратичної, так і у невеликій області гладкі функції з

високою точністю можуть бути представлені квадратичною функцією. Низка послідовних апроксимацій з рішенням задачі КП дозволяють знайти рішення початкової нелінійної задачі з необхідною точністю.

4.4. Сепарабельне програмування

В сепарабельному програмуванні розглядаються задачі, в яких цільова функція та всі функції обмежень є сепарабельними.

Нагадаємо, що функція багатьох змінних сепарабельна, якщо вона має вигляд суми функцій окремих змінних:

$$f(x_1, x_1, ..., x_n) = \sum_{j=1}^{m} f_j(x_j). \tag{4.18}$$

Лінійні функції завжди сепарабельні, тому лінійне програмування можна розглядати як особливий випадок сепарабельного.

Вирішення задач сепарабельного програмування грунтується на перетворенні їх в задачу лінійного програмування шляхом апроксимації нелінійних функцій кусково-лінійними. Таким чином, початкова нелінійна задача замінюється апроксимуючою лінійною. Тому розглянутий спосіб є приблизним, а точність рішення безпосередньо залежить від точності апроксимації і теоретично може бути будь-якою високою.

Існує 2 основні способи запису апроксимуючої задачі, що відрізняються формою представлення початкових змінних.

4.5. Задачі дробово-лінійного програмування

Якщо цільова функція є співвідношенням лінійних функцій, і всі умови лінійні, то задача відноситься до класу задач дробово-лінійного програмування.

У загальному випадку цільова функція має вигляд:

$$f = \frac{c_0 + \sum_{j=1}^{n} c_j x_j}{d_0 + \sum_{j=1}^{n} d_j x_j}. \tag{4.28}$$

Ця функція легко перетворюється на лінійну, якщо її знаменник суворо позитивний з усіма дійсними значеннями змінних. Для цього введемо нову змінну *r* наступним чином:

$$\frac{1}{d_0 + \sum_{j=1}^{n} d_j x_j} = r. \tag{4.29}$$

Очевидно, що з заданою умовою цільова функція може бути тільки більше нуля. Тоді функція (4.28) приймає вигляд

$$f = c_0 r + \sum_{j=1}^{n} c_j x_j r.$$

Замінивши добуток змінних

$$x_j r = y_j, \tag{4.30}$$

врешті маємо

$$f = c_0 r + \sum_{j=1}^{n} c_j y_j. \tag{4.31}$$

Отримали лінійну функцію від *n* невід'ємних змінних y_j та однієї позитивної змінної *r*. Ця функція повинна розглядатися разом з умовою, що випливає з (4.29):

$$d_0 r + \sum_{j=1}^{n} d_j x_j r = 1 \quad ,$$

або, після заміни (4.30) –

$$d_0 r + \sum_{j=1}^{n} d_j y_j = 1. \tag{4.32}$$

Щоб завершити побудову еквівалентної лінійної моделі, обмеження задачі повинні бути записані в нових змінних. Для цього, помножимо обидві частини кожного обмеження $\sum_j a_{ij} x_j \le b_i$ на *r*:

$$\sum_j a_{ij} x_j r \le b_i r$$

і, здійснюючи заміну, отримаємо

$$\sum_j a_{ij} y_j - b_i r \le 0, \quad i = \overline{1, m.} \tag{4.33}$$

В результаті перетворення у нас є проблема лінійного програмування з критерієм (4.31), обмеженнями (4.32), (4.33) та змінними $r > 0$, $y_j \ge 0$, $\forall j$. Отримавши його рішення одним з методів LP, ми розраховуємо початкові змінні за допомогою очевидної формули

$$x_j^* = \frac{y_j^*}{r^*}.$$

Для забезпечення виконання умови позитивності знаменника, доцільно ввести його явно в модель, тобто додати нерівність

$$d_0 + \sum_{j=1}^{n} d_j x_j \ge \varepsilon,$$

де ε – дуже мала позитивна константа.

Можливість переходу до лінійної задачі геометрично обумовлена тим, що лінії рівня дробово-лінійної описуються лінійним рівнянням. Дійсно, нехай це буде $f = \bar{f} = const$. Тоді з (4.28) слідує:

$$d_0 \bar{f} + \bar{f} \sum_j d_j x_j = c_0 + \sum c_j x_j \qquad \text{або}$$

$$\sum_j (c_j - \bar{f}\, d_j)\, x_j = d_0 \bar{f} - c_0. \tag{4.34}$$

Це звичайне лінійне рівняння відносно x_j. Тому лінії рівня функції (4.28) у багатовимірному просторі-гіперплощині, а у двовимірному – прямі. Однак, зі зміною $\bar{f}$, вони не рухаються паралельно, а обертаються навколо множини обертання.

Множина обертання – це множина точок $n-2$, утворена перетином нульових ліній рівня чисельника і знаменника:

$$\begin{cases} c_0 + \sum c_j \cdot x_j = 0, \\ d_0 + \sum d_j \cdot x_j = 0. \end{cases}$$

Для $n-2$ вона складається з однієї точки, а при $n-3$ – є прямою (віссю обертання), утвореною перетином двох площин.

4.6. Методи спуску

Так звуться чисельні ітераційні методи оптимізації, орієнтовані на пошук мінімума.

Загальна схема рішення полягає в генерації послідовності наближень, яка асимптотично збігається до мінімуму. Як правило, це локальний мінімум, але у випадку завдач опуклого програмування він також є глобальним. За кінцеве число ітерацій методи дозволяють отримати приблизне рішення з вказаною точністю. Конвергенція методу, як і швидкість конвергенції, залежать від властивостей задачі і початкового наближення. Швидкість конвергенції багато в чому визначає ефективність методу.

Основні показники ефективності (швидкості конвергенції) є кількість ітерацій та кількість розрахунків функції при інших рівних умовах. Кількість ітерацій для однієї й тої задачі сильно залежить від початкового наближення (початкової точки) та необхідної точності. У той же час кількість необхідних ітерацій зростає набагато швидше, ніж точність.

При виборі методу необхідно враховувати властивості цільової функції: унімодальність або мульти-екстремальність, диференційність, опуклість-увігнутість або їх відсутність тощо. Крім того, функції можуть мати «неприємний» для методу особливості, такі як сідлові точки та «яри». «Яр» (при максимізаціх «гребінь») проявляється в тому, що впродовж нього функція змінюється набагато слабкіше, ніж у поперечному напрямку. На карті ліній рівня його видно по сильній витягнутості ліній вздовж «дна» яру з одночасною «сплющенністю» у перетині. Найпростішим прикладом такої функції є еліпсоїд

$$f(x) = \frac{x_1^2}{a^2} + \frac{x_2^2}{b^2}.$$

При $a >> b$ функція має «яр» впродовж осі x_1, яка є його «дном». Чим сильніше нелінійність «дна», тим більше уповільнює швидкість конвергенції методу. Можлива навіть зупинка процесу оптимізації при недосягненні мінімуму. Тому урахування таких властивостей особливо важливо при виборі методу оптимізації.

Існують методи безумовної оптимізації, які використовуються для виявлення мінімуму без обмежень щодо змінних, та методи умовної оптимізації, коли пошук здійснюється за наявності обмежень.

Відповідно до інформації, яка використовується для визначення напрямку пошуку, розрізняються такі методи «спуску»:

• нульового порядку або прямі, якщо розраховується лише значення цільової функції;

• першого порядку або градієнтні, що використовують перші похідні (градієнт);

• другого порядку, що вимагає розрахунків також других похідних;

• випадкового пошуку, що застосовують механізм випадкового вибору напрямку;

• генетичні, що поєднують елементи детермінізму та випадковості вибору;

• комбіновані методи.

При оптимізації *n*-мірних функцій часто використовуються методи одномірної мінімізації. Тому спочатку розглянемо ці методи, потім багатовимірні методи безумовної оптимізації та, нарешті, методи пошуку умовного мінімуму. Всі методи, наведені нижче, призначені для мінімізації унімодальних функцій.

4.7. Методи одномірної мінімізації

Спершу розглядаються прямі методи. Перші з них орієнтовані на пошук безумовного мінімуму, інші – на пошук мінімуму на заданому безперервному інтервалі.

4.7.1. Метод поділу кроку навпіл

Початкова точка та початковий крок встановлюються на основі попередніх знань про функцію. Рух з постійним кроком в одному напрямку продовжується, доки значення функції не зменшується. Зі збільшенням значення функції напрямок змінюється на протилежний, а крок зменшується удвічі.

Пошук закінчується, коли на наступному етапі значення функції погіршується, а довжина кроку менше зазначеної точності (рис 4.1). Очевидно, що після проходження мінімуму, коливання відбуваються навколо нього зі зменшенням амплітуди.

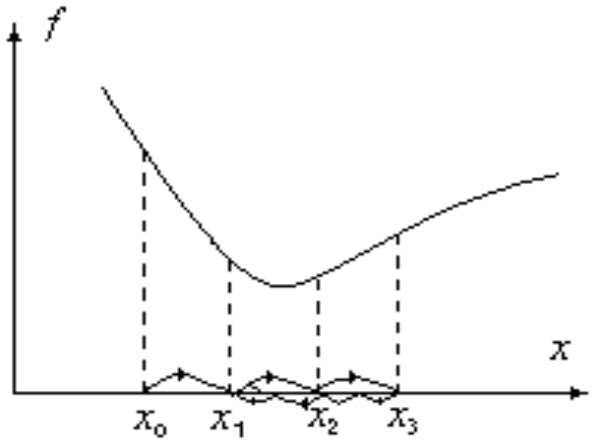


Рис. 4.1. Метод поділу кроку навпіл

Алгоритм методу ділення кроку навпіл:

1) Задати начальну точку x_0, начальний крок h_0 і точність ε_f; $k = 0$.
2) Обчислити $f(x_k)$.
3) $x_{k+1} = x_k + h_k$.
4) Обчислити $f(x_{k+1})$.
5) Якщо $f(x_{k+1}) < f(x_k)$, то $h_{k+1} = h_k$ та перейти на 9.
6) Якщо $h_k < \varepsilon$, перейти на 4.
7) Якщо $k = 0$, то $h_{k+1} = -h_k$ та перейти на 9.
8) $h_{k+1} = -h_k / 2$;
9) $k = k + 1$ та перейти на 3.
10) Кінець.

Перевірка в п.7 необхідна, щоб не зменшити крок до досягнення околиці мінімуму. Після закінчення алгоритму будь-яке значення між x_k и x_{k+1} можна взяти як оптимальне x^*. При великих коливаннях функції в околиці мінімуму, умову п.6 можна замінити на $(h_k < \varepsilon)$ & $(f(x_{k+1}) - f(x_k) < \varepsilon_f)$, де ε_f – точність функції.

4.7.2. Квадратична апроксимація

Щоб знайти приблизний мінімум, початкова функція апроксимується функцією другого порядку

$$\bar{f}(x) = a(x - x_0)^2 + b(x - x_0) + c, \qquad (4.35)$$

де x_0 – точка відліку реального діапазона значень x, зокрема $x_0 = 0$. Для визначення коефіцієнтів, включених у функцію (4.35), вибираються 3 точки і в них обчислюють значення функції f. Отримують просту систему 3-х рівнянь з 3-ма невідомими a, b і c:

$$f(x_1) = a(x_1 - x_0)^2 + b(x_1 - x_0) + c,$$
$$f(x_2) = a(x_2 - x_0)^2 + b(x_2 - x_0) + c,$$
$$f(x_3) = a(x_3 - x_0)^2 + b(x_3 - x_0) + c.$$

Стаціонарна точка x_a функції (4.35) обчислюється з умови рівності її похідної нулю:

$$x_a = x_0 - \frac{b}{2a}. \qquad (4.36)$$

Вона використовується в умові зупинки як нова точка для апроксимації.

Алгоритм методу квадратичної апроксимації.

1. Задати першу точку x_1, крок Δx та точність за координатою ε і функцією ε_f.
2. Визначити 2-у точку: $x_2 = x_1 + \Delta x$.
3. Обчислити $f(x_1)$ та $f(x_2)$.
4. Якщо $f(x_2) < f(x_1)$, прийняти $x_3 = x_2 + \Delta x$, інакше $x_3 = x - \Delta x$.
5. Обчислити $f(x_3)$.
6. Обчислити $f_m = \min\{ f(x_1), f(x_2), f(x_3)\}$, визначити точку x_m, що відповідає f_m.
7. За 3-ма точками і значенням функції знайти коефіцієнти в (4.35).
8. Обчислити x_a за формулою (4.36).
9. Якщо ($| f_m - f(x_a)| < \varepsilon_f$) & ($|x_m - x_a | < \varepsilon$), закінчити пошук.
10. Якщо $f_m < f(x_a)$, взяти x_m та 2 найбільш близькі до неї точки; інакше взяти x_a та 2 найбільш близькі до неї точки. Обрані точки пронумерувати в порядку зростання значень, обчислити значення f в новій точці й перейти на 6.

4.7.3. Метод поділу інтервалу навпіл

Цей метод, як і 2 наступних, належить до методів скорочення інтервалу невизначеності. Вважається, що точка мінімуму знаходиться на заданому інтервалі [*a, b*].

Розглянутий метод також називається 3-х точечним. Точки обираються таким чином, що інтервал поділяється на 4 рівних частини (рис. 4.2):

$$x_m = (a + b) / 2, \quad x_1 = (a + x_m) / 2, \quad x_2 = (x_m + b) / 2.$$

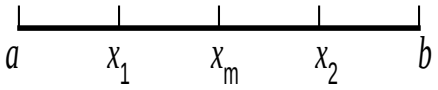


Рис. 4.2. Метод ділення інтервалу навпіл

Функція розраховується в цих точках і після порівняння його значень інтервал зменшується в 2 рази. З новим інтервалом повторюється дія.

Алгоритм методу ділення інтервалу навпіл:

1. Встановити точність координати ε.
2. Обчислити x_m та $f(x_m)$.
3. Обчислити x_1 та x_2.
4. Обчислити $f(x)$ в этих точках.
5. Якщо $f(x_1) < f(x_m)$, то прийняти $b = x_m$, $x_m = x_1$.

Інакше, якщо $f(x_2) < f(x_m)$, то взяти $a = x_m$, $x_m = x_2$, інакше $a = x_1$, $b = x_2$.

6. Якщо $b - a < \varepsilon$, закінчити пошук, інакше перейти на 3.

Легко розрахувати, що після n обчислень функції інтервал невизначеності зменшиться в $2^{\frac{n-1}{2}}$ раз. Доведено, що цей метод є більш ефективним, ніж інші прямі методи, які використовують рівномірно розташовані точки.

4.7.4. Метод золотого перетину

Золотий перетин – це певне відношення частини до цілого. Відрізок АВ ділиться точкою С у відношенні золотого перетину (рис. 4.3), якщо

$$\frac{AC}{AB} = \frac{CB}{AC}. \qquad (4.37)$$

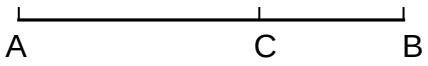


Рис. 4.3. Золотий перетин відрізка АВ точкою С

Приймемо АВ = 1, АС = x, СВ = 1 – x, тоді з (4.37) отримуємо рівняння:

$x^2 + x - 1 = 0$, з якого слідує $x = \frac{\sqrt{5}-1}{2} \approx 0.618$, $1 - x = \frac{3-\sqrt{5}}{2} \approx 0.382$.

Ці відношення використовуються для вибору 2-х точок всередині інтервалу невизначеності. Вони розташовуються, як показано на рисунку 4.4. Кожна точка розділяє інтервал [a, b] у відношенні золотого перетину.

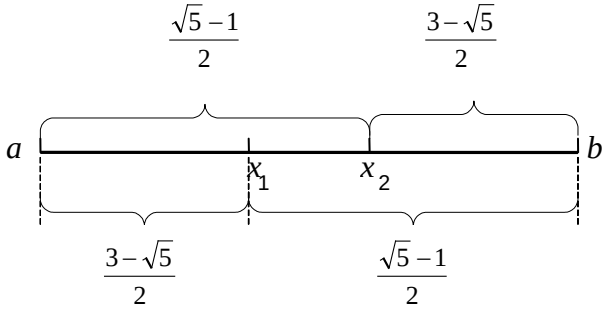


Рис. 4.4. До методу золотого перетину

На цих точках розраховується функція. Якщо $f(x_1) > f(x_2)$, то частина інтервалу $[a, x_1]$ відкидається, якщо $f(x_1) < f(x_2)$, то відрізається частина $[x_2, b]$, а при рівності значень функції – будь-яка з них. Частина інтервалу, що залишається, дорівнює $(\sqrt{5}-1)/2$ від початкової величини. Очевидно, що після такого зменшення інтервалу одна з внутрішніх точок залишається зі зміною індексу, а друга приймається на основі золотого перетину або, що те ж саме, симетрично точці, що залишається (див. рис 4.4). Скорочення інтервалу продовжується до досягнення зазначеної точності.

Алгоритм методу золотого перетину:

1. Задати точність за координатою ε.
2. Обчислити $x_1 = a + ((3-\sqrt{5})/2)(b-a)$, $x_2 = a + ((\sqrt{5}-1)/2)(b-a)$.
3. Обчислити $f(x_1)$, $f(x_2)$.
4. Якщо $f(x_1) > f(x_2)$, прийняти $a = x_1$, $x_1 = x_2$, $x_2 = a + ((\sqrt{5}-1)/2)(b-a)$

або $x_2 = a + b - x_1$, інакше $b = x_2$, $x_2 = x_1$, $x_1 = a + \left(\left(3 - \sqrt{5}\right)/2\right)(b - a)$,

або $x_1 = a + b - x_2$.

5. Якщо $(b - a) < \varepsilon$, то завершити пощук.

6. Обчислити функцію в новій точці й перейти на 4.

Ітерації алгоритму ілюструються на рисунку 4.5.

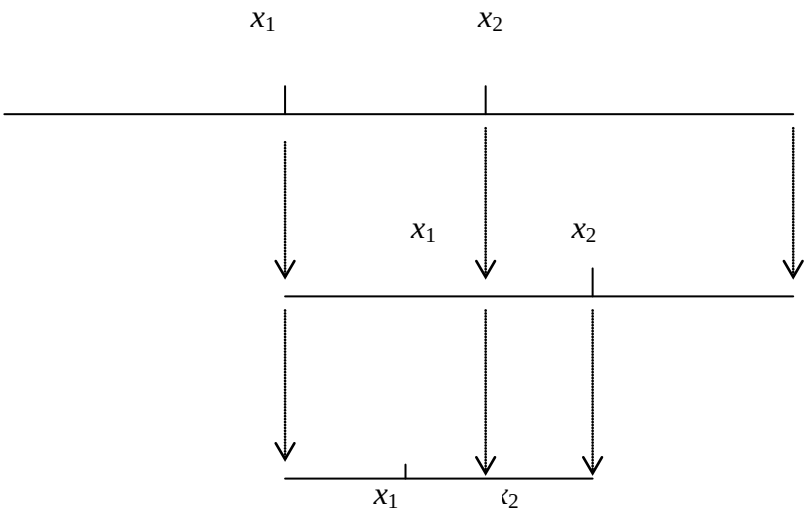


Рис. 4.5. Ітерації алгоритму методу золотого перетину

4.7.5. Метод Фібоначчі

Метод методу практично повністю збігається з методом золотого перетину. Різниця полягає в тому, що замість золотого перетину використовується співвідношення номерів Фібоначчі: на k-й ітерації частка малого та великого інтервальних сегментів дорівнює F_{n-k-1} / F_{n-k+1} и F_{n-k} / F_{n-k+1} відповідно.

Числа Фібоначчі F_ν обчислюються за співвідношеними: $F_0 = F_1 = 1$, $F_\nu = F_{\nu-1} + F_{\nu-2}$, $\nu \geq 2$.

Точки x_1 и x_2 обчислюються за формулами:

$$x_1^k = a_k + \frac{F_{n-k-1}}{F_{n-k+1}} \cdot (b_k - a_k), \tag{4.38}$$

$$x_2^k = a_k + \frac{F_{n-k}}{F_{n-k+1}} \cdot (b_k - a_k). \tag{4.39}$$

Як видно, вони ідентичні наведеним у попередньому розділі. Однак, якщо при використанні золотого перетину внутрішні точки не можуть зливатись, то тут це не так. Насправді, при $k = n - 1$ з (4.38) і (4.39) маємо:

$$x_1^{n-1} = a_{n-1} + \frac{F_0}{F_2} \cdot (b_{n-1} - a_{n-1}), \quad x_2^{n-1} = a_{n-1} + \frac{F_1}{F_2} \cdot (b_{n-1} - a_{n-1}).$$

Але тому, що $F_0/F_2 = F_1/F_2 = 1/2$, то $x_1^{n-1} = x_2^{n-1}$, точки зливаються в середині інтервалу. Тому до початку ітерацій необхідно визначити значення n, що гарантує досягнення мінімуму з заданою точністю ε. Після 1-ї ітерації довжина інтервалу буде F_{n-1} / F_n від початкової величини, після 2-ї – $(F_{n-2} / F_{n-1})(F_{n-1} / F_n)$,…, після ($n$-1)-ї –

$$\left(\frac{F_{n-1}}{F_n}\right) \cdot \left(\frac{F_{n-2}}{F_{n-1}}\right) \cdot \left(\frac{F_{n-3}}{F_{n-2}}\right) \cdot \ldots \cdot \left(\frac{F_1}{F_2}\right) = \frac{F_1}{F_n} = \frac{1}{F_n}.$$

Отже, довжина останнього інтервалу дорівнюватиме $(b_1 - a_1)/F_n$, де $[a_1, b_1]$ – початковий інтервал. Для забезпечення заданної точності треба, щоб

$$\frac{b_1 - a_1}{F_n} \le \varepsilon \text{ или } F_n \ge \frac{b_1 - a_1}{\varepsilon}. \qquad (4.40)$$

Таким чином, співвідношення (4.40) дозволяє визначити номер числа Фібоначчі за початковими даними. На початковому інтервалі точки розраховуються за формулами (4.38) та (4.39) при $k = 1$. В наступних ітераціях числа Фібоначчі не потрібні, оскільки одна точка переноситься з попередньої ітерації, а друга приймається симетрично до неї, тобто краще використовувати другі формули з алгоритму п. 4 золотого перетину.

Після злиття внутрішніх точок залишається невизначеність з позицією мінімуму. Щоб її усунути, друга точка береться ліворуч або праворуч від центру на відстані $\varepsilon_1 \cong (0{,}01 \ldots 0{,}05)\ \varepsilon$. Для випадку зсуву другої точки ліворуч (рис. 4.6), при $f(x_1) < f(x_1 - \varepsilon_1)$ мінімум лежить в інтервалі (2), інакше – в (1) .

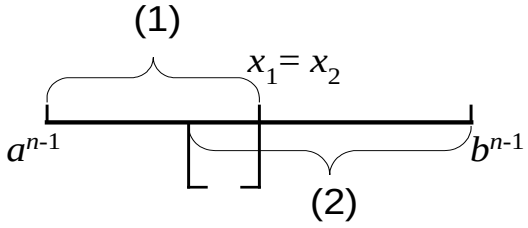


Рис. 4.6. Випадок зсуву другої точки ліворуч

Метод Фібоначчі є найбільш ефективним з усіх прямих методів. Метод золотого перетину дуже близький до нього: при $n > 9$ вони практично збігаються за ефективністю і чим більше n, чим ближче ці методи. Нарешті, відношення, яке використовується в методі Фібоначчі на 1-й ітерації, стає рівним золотому перетину:

$$\lim_{n\to\infty}\left(\frac{F_{n-2}}{F_n}\right)=\frac{3-\sqrt{5}}{2}.$$

4.8. Багатовимірний пошук безумовного мінімуму

До методів багатовимірного пошуку безумовного мінімуму відносяться прямі методи: координатного спуску (Гауса-Зейделя), конфігурації (Хука-Дживса) і симплексний, а також градієнтні методи, метод другого порядку та алгоритми випадкового пошуку.

4.8.1. Метод Гаусса-Зейделя

Процедура пошуку зводиться до вирішення послідовності задач одновимірної мінімізації для кожної змінної. Нехай буде вибрано початкову точку:

$$\mathbf{x}^{(0)}=\left(x_1^{(0)},x_2^{(0)},...,x_n^{(0)}\right).$$

Зафіксуємо усі змінні, крім першої, на початкових значеннях і вирішуємо задачу

$$f(x_1,x_2^{(0)},...,x_n^{(0)})\Rightarrow\min_{x_1}$$

одним з одновимірних методів. Фіксуємо x_1 на значенні x_1', отриманому в рішенні, і робимо змінну x_2 вільною. Отримуємо іншу одновимірну задачу:

$$f(x'_1,x_2,x_3^{(0)},...,x_n^{(0)})\Rightarrow\min_{x_2}\to x'_2.$$

Аналогічно побудовуються і вирішуються наступні одновимірні задачі, остання з яких має вигляд:

$$f(x'_1,x'_2,...,x'_{n-1},x_x)\Rightarrow\min_{x_n}\to x'_n.$$

Ці n задач складають один цикл. Його результатом є точка X^1. Вона приймається як відправна точка для наступного подібного циклу. Пошук закінчується, коли відстань між двома послідовними точками стає меншою за вказану похибку:

$$\left\| \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)} \right\| < \varepsilon .$$

Роботу методу ілюструє рисунок 4.7, на якому показано траєкторію пошуку мінімуму функції $f = (2 - x_1)^4 + 2\,(x_1 - 2\,x_2)^2$.

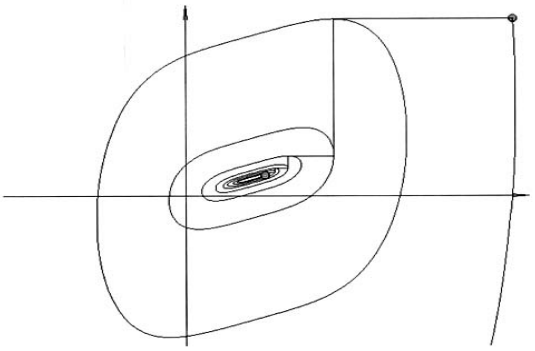
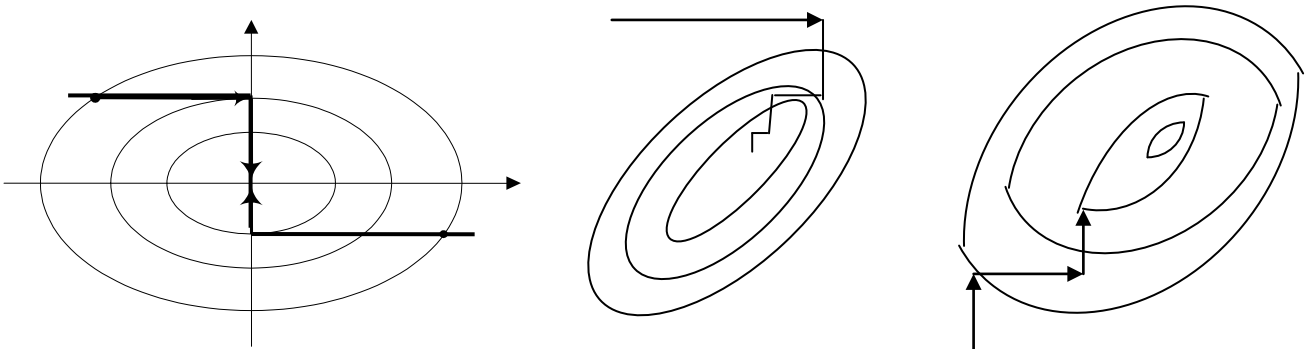


Рис. 4.7. Робота методу Гаусса-Зейделя

Метод відрізняється алгоритмічною простотою. Однак йому притаманний ряд недоліків. Його ефективність суттєво залежить від напрямку координатних осей відносно ліній рівня. Це чітко видно на прикладі квадратичної функції: коли координати співпадають з осями еліпсів, мінімум досягається за один цикл з будь-якої вихідної точки (рис 4.8, а), а з їх поворотом кількість циклів значно збільшується (рис. 4.8, б).



а б в

Рис. 4.8. Варіанти методу Гаусса-Зейделя

З цього прикладу також випливає, що метод неефективний в умовах яру. Якщо функція не диференцюється в окремих точках, пошук може зупинитися, не досягаючи околиці мінімуму. Рисунок 4.8, в показує такий випадок: точка зупинки далека від шуканого мінімуму.

З аналізу траєкторій пошуку у наведених вище прикладах можна зробити висновок, що ефективність пошуку збільшиться, якщо до описаних однотиповим циклам додати рух у напрямку, що проходить через точки $X^{(k)}$ та $X^{(k+1)}$.

Цей рух називається прискорюючим кроком. Він використовується в методі, розглянутому в наступному розділі.

4.8.2. Метод Хука-Дживса (метод конфігурацій)

У цьому методі кожна ітерація складається з 2-х етапів: 1) вивчення пошуку; 2) рух за зразком (прискорюючий крок).

Досліджуючий пошук схожий на той один цикл узгодження координат. Кінцева точка циклу називається базовою. Дві послідовні базові точки визначають напрямок пошуку на 2-му етапі. Точка, отримана у результаті прискорюючого кроку, називається тимчасовою. Початкова точка одночасно є базовою і тимчасовою.

По-перше, ми надамо варіант методу, запропонований Хуком і Дживсом і описаний у літературі під назвою «метод конфігурації». Він використовує лише дискретні кроки. Процедура пошуку показана на рисунку 4.9.

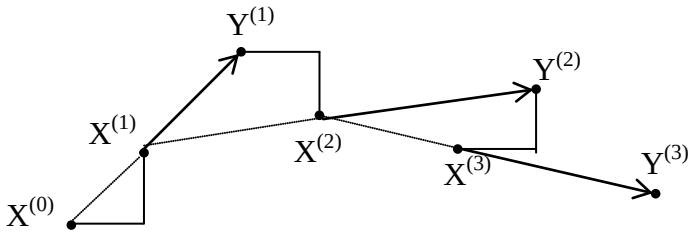


Рис. 4.9. Метод конфігурацій

З вихідної точки $X^{(0)}$ виконується крок Δ за x_1 в одному напрямку і, якщо це успішно, значення фіксується, інакше напрямок кроку змінюється на протилежний. Також робиться за рештою координат. Результатом першого досліджуючого пошуку є точка $X^{(1)}$. У напрямку $X^{(1)} - X^{(0)}$ виконується рух за

зразком, який дає тимчасову точку $Y^{(1)} = X^{(1)} + \alpha(X^{(1)} - X^{(0)})$, де α – коефіцієнт прискорення (рекомендується вибирати його значення в діапазоні від 1 до 2).

З отриманої тимчасової точки знову здійснюється досліджуючий пошук, що призводить до наступної базової точки $X^{(2)}$. Прискорюючий крок у напрямку $X^{(2)} - X^{(1)}$ дає тимчасову точку $Y^{(2)}$. Фази пошуку повторюють, не змінюючи величину кроку, якщо $f(Y^{(k)}) < f(X^{(k)})$. В іншому випадку, якщо $\Delta < \varepsilon$, пошук закінчується та $X^{(k)}$ приймається як рішення, а інакше слід покласти $Y^{(k)} = X^{(k)}$, зменшити крок удвічі і перейти до досліджуючого пошуку з точки $Y^{(k)}$. Іншими словами, після кожного прискорюючого кроку перевіряється його успішність і, залежно від результату, вибираються подальші дії.

Модифікація методу Хука-Дживса полягає в тому, щоб замінити дискретні кроки одномірною мінімізацією. У цьому варіанті досліджуючий пошук повністю збігається з одним циклом покоординатного спуску, тобто для кожної координати виконуються не дискретні кроки, а шукається мінімум. При русі за зразком також шукається мінімум функції тільки за однією невідомою – коефіцієнтом прискорення α:

$$f\left(\mathbf{X}^{(k)} + \alpha\left(\mathbf{X}^{(k)} - \mathbf{X}^{(k-1)}\right)\right) \Rightarrow \min_{\alpha}.$$

За оптимальним значенням α^* визначається часова точка

$$\mathbf{Y}^{(k)} = \mathbf{X}^{(k)} + \alpha^*\left(\mathbf{X}^{(k)} - \mathbf{X}^{(k-1)}\right).$$

Пошук закінчується, коли вілстань між двома сусідніми базовими точками стає меншою за вказане значення:

$$\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right\| < \varepsilon.$$

До цієї умови можна додати вимогу щодо точності функції:

$$\left|f\left(\mathbf{x}^{(k+1)}\right) - f\left(\mathbf{x}^{(k)}\right)\right| < \varepsilon_f.$$

Метод забезпечує швидке наближення до області бажаного рішення. Ще однією важливою перевагою методу є його продуктивність в умовах ярності. Траєкторія пошуку добре адаптується до згинань дна яру.

4.8.3. Симплексний метод

У цьому методі рух до мінімуму здійснюється за допомогою симплексу – опуклого багатогранника з кількістю вершин на одну більше, ніж розмірність простору. Приклад симплексу на площині є будь-який трикутник, у тривимірному просторі – піраміда з трикутником на базі.

Симплекс має чудову властивість: якщо ви берете лише одну точку за його межами як нову вершину, то з'єднавши її з вершинами прилежної грані, отримаємо новий симплекс. На рисунку 4.10 зліва від початкового симплекса з вершинами 1, 2, 3 побудовано 2 інших симплекса, а справа новий симплекс побудований на вершинах 2, 3, 4 початкового симплекса та нової вершини 5.

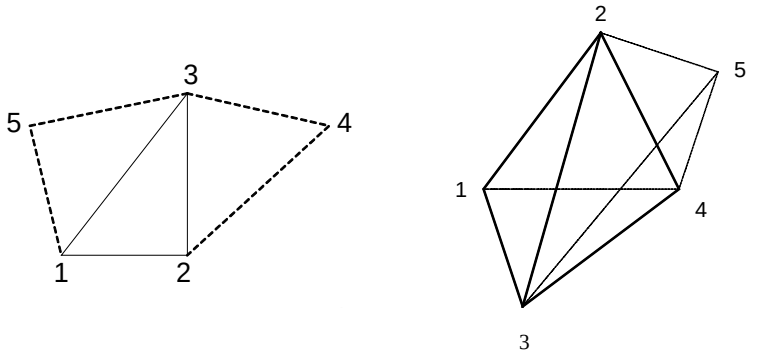


Рис. 4.10. Отримання нового симплекса

Очевидно, якщо ви приймаєте одну з вершин симплекс, то всі інші вершини будуть знаходитись на протилежній грані симплексу. Ці властивості дійсні для будь-якої розмірності простору.

Знаючи значення функції у вершинах симплекса, легко визначити напрямок, в якому функція може покращити своє значення. У цьому напрямку будується новий симплекс: визначається найгірша вершина (за значенням функції), і вона відображається через центр протилежної грані, як показано на рисунку 4.11.

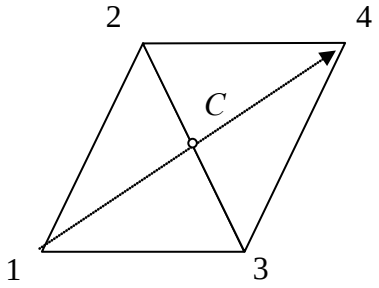


Рис. 4.11. Симплексний метод

У побудованому симплексі значення функції невідоме лише у вершині 4. Таким чином, після побудови нового симплексу функція розраховується лише один раз при будь-якому виміру простору. Тільки в початковому симплексі необхідно розрахувати функцію у всіх вершинах.

Послідовне відображення найгірших вершин переміщує симплекс у напрямку зменшення значення функції, що, в кінцевому підсумку, призводить до пошуку мінімуму з заданою точністю. Характер руху симплексу представлено на рисунку 4.12.

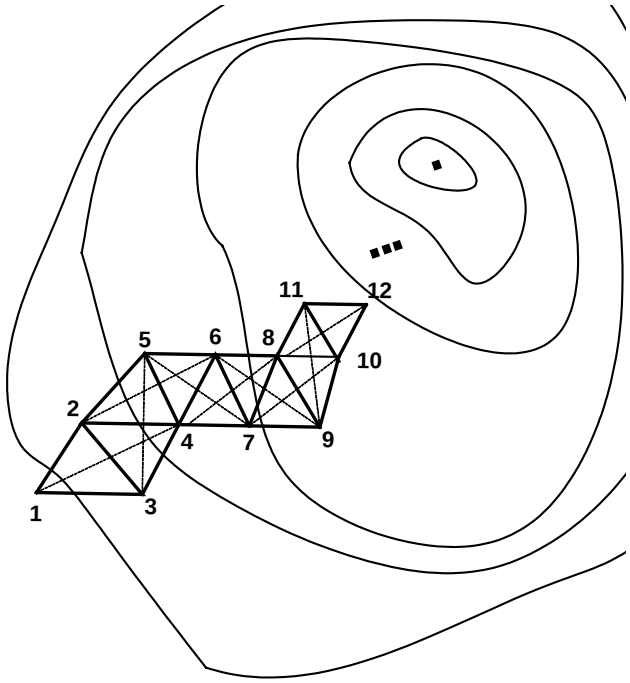


Рис. 4.12. Характер руху симплексу

Відбиття вершин показано пунктирними лініями. По-перше, відбивається вершина 1, потім в отриманому симплексі відображається вершина 3, симплекс 2, 4, 5 замінюється на симплекс 4, 5, 6 і так далі. Очевидно, що в процесі руху, особливо біля екстремуму, нова вершина може бути не кращою від відбитої. Щоб запобігти петлі або відбивається інша вершина, або зменшується розмір нового симплексу.

Розглянемо обчислювальну сторону методу. Регулярний симплекс, як правило, приймається як початковий. Регулярність означає рівність довжин усіх ребер. Він будується наступним чином. Початкова точка $X^{(0)}$ береться для основної вершини, по відношенню до якої розташовані інші вершини. Їх координати розраховуються за формулами:

$$x_j^{(i)} = \begin{cases} x_j^{(0)} + \delta_1, & \text{если } j \neq i, \\ x_j^{(0)} + \delta_2, & \text{если } j = i; \quad i, j = \overline{1,n}, \end{cases}$$

де i – номер вершини, j – координата вершини (індекс змінної), n – число змінних (розмірність простору), а приращення δ_1 и δ_2 залежать від n та вибраного масштабного множника μ:

$$\delta_1 = \left(\frac{\sqrt{n+1} + n - 1}{n\sqrt{2}} \right) \mu,$$

$$\delta_1 = \left(\frac{\sqrt{n+1} - 1}{n\sqrt{2}} \right) \mu.$$

При $\mu = 1$ усі ребра мають одиничну довжину.

Розрахунок координат нової вершини, що виникає внаслідок відбиття однієї з вершин останнього симплексу, полягає в наступному. Хай відбивається вершина k (наприклад, вершина 1 на рис 4.12). Тоді спочатку визначається центр протилежної грані C:

$$\mathbf{X^c} = \frac{1}{n} \sum_{\substack{i=1 \\ i \neq k}}^{\mathbf{n+1}} \mathbf{x^{(i)}},$$

а потім – координати нової вершини s:

$$\mathbf{x}^{(s)} = \mathbf{x}^{(k)} + \lambda \cdot (\mathbf{x}^C - \mathbf{x}^{(k)})$$

або

$$\mathbf{x}^{(s)} = \mathbf{x}^C + \theta \cdot (\mathbf{x}^C - \mathbf{x}^{(k)}), \qquad (4.42)$$

де $\lambda > 0$ і θ – параметри відбиття.

Метод характеризується як ефективний. Він мало чутливий до перешкод або помилок при визначенні значень цільової функції, оскільки визначаються співвідношення значень, а не абсолютні величини. Через зміну розміру симплексу, він може працювати й в умовах яру.

4.9. Градієнтний метод

Якщо функція є диференцюємою, обчислення похідних дає інформацію про поведінку функції в досліджуваній точці і, отже, дозволяє знайти найкращий напрям пошуку.

Швидкість зміни функції $f(\mathrm{X})$ у довільному напрямку l визначається похідною

$$\frac{df}{dl} = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i} \frac{dx_i}{dl}. \tag{4.43}$$

Тут частні похідні $\frac{dx_i}{dl}$ є направляючими косинусами. Геометрична ілюстрація для випадку функції двох змінних представлена на рисунку 4.13.

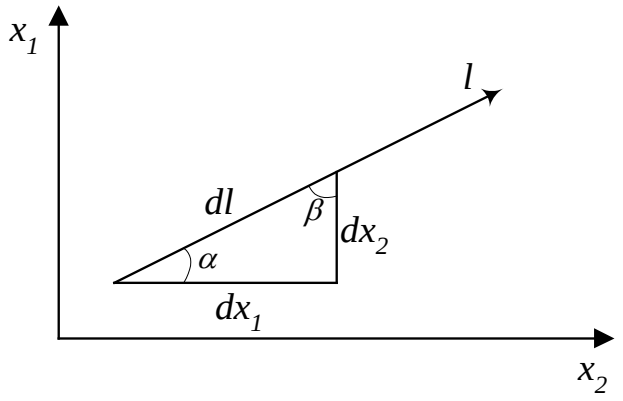


Рис. 4.13. Градієнт функції двох змінних

Очевидно, що $\left.\begin{aligned} \frac{dx_1}{dl} &= \cos(\hat{x_1}, l) = \cos\alpha, \\ \frac{dx_2}{dl} &= \cos(\hat{x_2}, l) = \cos\beta. \end{aligned}\right\}$.

Поверхня рівня цільової функції, що описується рівнянням $f(\mathrm{X}) = const$, має розмірність $n - 1$.

Задамо координати наступним чином. Проведемо через точку $n - 1$ взаємно ортогональні дотичні до поверхні рівня, прийнявши їх за осі координат. Як останню вісь, ми приймаємо нормаль до поверхні рівня (рис. 4.14).

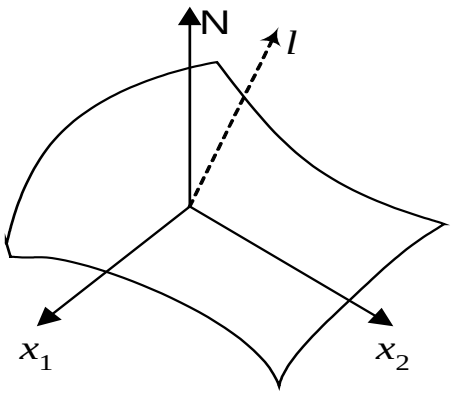


Рис. 4.14. Побудова градієнта на 3-х мірній поверхні

У такій системі координат, похідні на всіх x_j дорівнюють нулю. Тому з (4.43) маємо

$$\frac{df}{dl} = \frac{\partial f}{\partial N} \cdot \cos(\overset{\wedge}{N,l}).$$

Звідси випливає, що максимальна швидкість збільшення функції буде у напрямку l, що збігається з нормаллю. Вектор, що має напрямок нормалі до поверхні функції на заданій точці та довжину $\partial f / \partial N$, називається *градієнтом*.

Позначається градієнт як grad f (X) або ∇f (X). Він повністю визначається своїми проекціями – похідними функціями за змінними:

$$\nabla f(\mathbf{x}^k) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ..., \frac{\partial f}{\partial x_n} \right).$$

У задачах на мінімум використовується зворотний напрямок – *антигіградієнт*.

Значення похідних можна знайти за приблизними формулами:

$$\frac{\partial f}{\partial x_i} \approx \frac{\Delta f}{\Delta x_i} = \frac{f(x_1,..., x_{i-1}, x_i + \Delta x_i, x_{i+1},..., x_n) - f(x_1,..., x_i,..., x_n)}{\Delta x_i},$$

$$\frac{\partial f}{\partial x_i} \approx \frac{\Delta f}{\Delta x_i} = \frac{f(x_1,..., x_{i-1}, x_i + \Delta x_i / 2, x_{i+1},..., x_n) - f(x_1,..., x_i - \Delta x_i / 2,..., x_n)}{\Delta x_i}.$$

Більш точний результат дає друга формула, але вона вимагає більше розрахунків. Чим точніше необхідно розрахувати похідну, тим менше має бути Δx. Однак неприйнятно, щоб різниця між значеннями функції відповідала похибці розрахунку.

Якщо змінні мають різні одиниці виміру, ви можете перейти до відносних змінних y_i, використовуючи мінімальні та максимально можливі значення змінних x_i:

$$y_i = \frac{x_i - x_i^{\min}}{x_i^{\max} - x_i^{\min}}.$$

Очевидно, что значення y_i лежать в діапазоні [0,1].

Знання градієнта (антіградієнта) дозволяє переміщатись від поточної точки у напрямку найбільшого поліпшення функції. Для лінійних функцій воно постійно і, тому, його повинно визначити лише один раз. У нелінійних

функціях градієнтне значення залежить від вектору X, тобто, він повинен бути перерахований при будь-якій зміні X.

У градієнтному методі пошук мінімуму виконується відповідно до рекурентних формул:

$$\mathbf{x}^{k+1} = \mathbf{x}^k - h^k \cdot \nabla f(\mathbf{x}^k),$$

$$\mathbf{x}^{k+1} = \mathbf{x}^k - h^k \cdot \frac{\nabla f(\mathbf{x}^k)}{\left\|\nabla f(\mathbf{x}^k)\right\|}, \qquad (4.44)$$

де h^k – шаг. У першій формулі значення зміни змінних ΔX залежить як від кроку, так і величини градієнта. Більш зручно, коли відстань між послідовними точками залежить тільки від значення кроку. Тому друга формула є кращою. В ній використовується нормалізований градієнт, довжина якого завжди дорівнює одиниці (через поділ на довжину). Тому він лише визначає напрямок, але не впливає на значення ΔX.

Формула (4.44) у запису для окремих змінних приймає вигляд:

$$x_i^{k+1} = x_i^k - h^k \cdot \frac{\partial f / \partial x_i}{\sqrt{\sum_{j=1}^{n} \left(\partial f / \partial x_j\right)^2}}. \qquad (4.45)$$

Алгоритм градієнтного методу.

1) Встановити початкову точку, початкове значення кроку та точність градієнта ε, обчислити $f(\mathrm{X}^0)$ и покласти $k = 0$.

2) В поточній точці X^k обчислити градієнт (похідні $\partial f / \partial x_i$) та його довжину.

3) Перевірити: якщо $\left\|\nabla f(\mathbf{x}^k)\right\| < \varepsilon$, то закінчити пошук.

4) Визначити нову точку X^{k+1} відповідно до (4.45) та обчислити в ній значення функції.

5) Перевірити: якщо $f(\mathrm{X}^{k+1}) \le f(\mathrm{X}^k)$, збільшити k на одиницю та перейти до кроку 2. Якщо інакше – зменшити h^k удвічі і перейдіть до кроку 4.

При гладкій похідній траєкторія пошуку в ідеалі (при безперервному розрахунку градієнту) також буде гладкою та ортогональною лініям рівня. При кінцевій величині кроку траєкторія стає кусочно-лінійною.

Якісний характер пошуку показаний на рисунку 4.21, а на рисунку 4.22 наведена реалізація алгоритму з 2-х початкових точок по відношенню до функції $f(X) = (x_1 - x_2)^2 + (x_2 - 2)^4$. Обидві траєкторії ведуть майже одну точку.

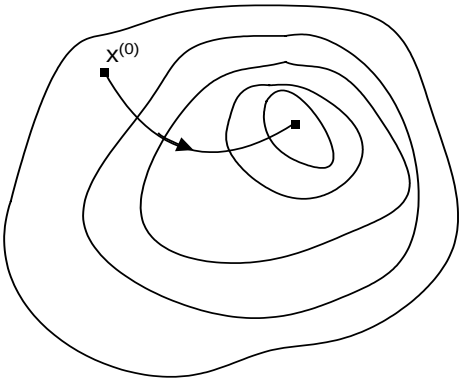


Рис. 4.15. Якісний характер градієнтного пошуку

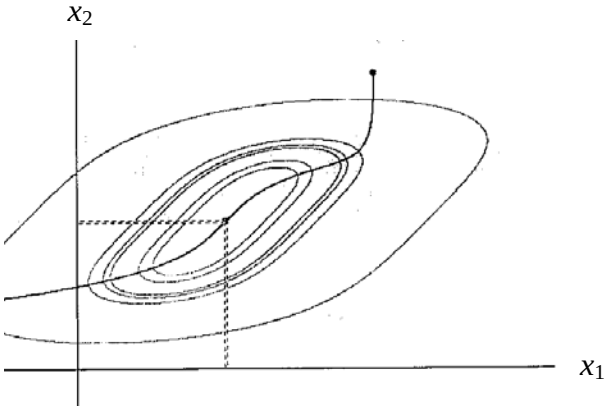


Рис. 4.16. Реалізація алгоритму пошуку з 2-х початкових точок

Найбільш трудомісткою частиною алгоритму є розрахунок градієнта (його обчислюють після кожного успішного кроку). В методі найскорішого спуску, який представляє собою модифікацію градієнтного методу, градієнт визначається рідше, особливо в початковому періоді пошуку.

Модифікація полягає в тому, що після розрахунку градієнта, замість одного дискретного кроку, шукається мінімум у напрямку антиградієнту, тобто здійснюється одномірний пошук за h:

$$f\left(\mathbf{x}^k - h \cdot \frac{\nabla f}{\|\nabla f\|}\right) \Rightarrow \min_h . \qquad (4.46)$$

В результаті рішення задачі (4.46) визначається оптимальний крок h^*, за яким знаходиться наступна точка:

$$\mathbf{x}^{k+1} = \mathbf{x}^k - h^* \cdot \frac{\nabla f}{\|\nabla f\|}. \qquad (4.47)$$

Очевидно, що при такому визначенні нової точки, значення функції в ній буде кращим, ніж в X^k.

Алгоритм найскорішого спуску.

1) Встановити початкову точку та точність градієнта ε, покласти $k = 0$.

2) В поточній точці X^k обчислити градієнт (похідні $\partial f / \partial x_i$) і його довжину.

3) Перевірити: якщо $\left\|\nabla f(\mathbf{x}^k)\right\| < \varepsilon,$ то закінчити пошук.

4) Провести одномірний пошук (4.46).

5) Визначити нову точку X^{k+1} відповідно до (4.47), збільшуючи k на одиницю та перейти до кроку 2.

Градієнтні методи погано працюють в умовах яру: на його дні різко падає швидкість руху і можлива зупинка до досягнення околиці мінімуму.

4.10. Метод Ньютона

Метод Ньютона – це класичний метод другого порядку. Якщо напрямок градієнта знаходиться з лінійного представлення функції у околиці поточної точки, то в методі Ньютона використовується квадратичне наближення цільової функції.

Квадратичне наближення диференційованої функції f в точці X^k записується як

$$f(\mathbf{x}) \approx f(\mathbf{x}^k) + \nabla f(\mathbf{x}^k)(\mathbf{x} - \mathbf{x}^k) + 1/2 \cdot (\mathbf{x} - \mathbf{x}^k) \cdot \mathbf{H}(\mathbf{x}^k)(\mathbf{x} - \mathbf{x}^k), \qquad (4.48)$$

де H – матриця других похідних f (матриця Гессе). У стаціонарній точці градієнт функції дорівнює нулю, що застосовно до (4.48) дає

$$\nabla f(\mathbf{x}^k) + \mathbf{H}(\mathbf{x}^k)(\mathbf{x}^{k+1} - \mathbf{x}^k) = 0. \qquad (4.49)$$

Вважаючи, що матриця H неособлива (є обернена до неї матриця), з (4.49) ми отримуємо реркурентний спосіб генерації послідовності точок

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \mathbf{H}^{-1}(\mathbf{x}^k) \cdot \nabla f(\mathbf{x}^k). \qquad (4.50)$$

На підставі виводу формули (4.50) зрозуміло, що для квадратичної цільової функції X^{k+1} є точка мінімуму. Отже, мінімум такої функціх з будь-якої вихідної точки досягається за один крок (4.50). Для нелінійної унімодальної функції, крім квадратичних, точки послідовності (4.50) асимптотично наближаються до мінімуму. Умова закінчення пошуку така ж, як в градієнтних методах $\left\|\nabla f(\mathbf{x}^k)\right\| < \varepsilon$. (У випадку лінійного наближення матриця H стає одиничною і пошук вироджується в градієнтний).

Передумовою конвергенції методу є існування оберненої матриці у всіх генерованих точках. Доказ конвергенції методу отримано за умови, що відправна точка досить близька до точки мінімуму. У цьому випадку метод має високу швидкість конвергенції. Якщо пошук починається з віддаленої точки і функція істотно відрізняється від квадратичної, метод може не сходитись або давати слабку конвергенцію. Причиною такої поведінки є те, що значення функції в точці X^{k+1} не обов'язково менше, ніж у точці X^k.

Для усунення зазначених недоліків існують модифікації методу.

Методи сполучених напрямків, як і метод Ньютона, засновані на властивостях квадратичних функцій. У зв'язку з цим кажуть про сполучені напрямки відносно квадратичної функції.

Нехай буде дано матрицю $H_{n\times n}$. Вектори $\mathbf{d_1}, \mathbf{d_2}, ..., \mathbf{d_k}$ $(k \le n)$ називаються сполученими або H-сполученими, якщо вони лінійно незалежні і

$$\mathbf{d}_i^{\mathrm{T}} \cdot \mathbf{H} \cdot \mathbf{d}_j = 0 \quad \text{при } i \neq j \,. \tag{4.52}$$

Ці вектори визначають сполучені напрямки. Для квадратичної функції двох змінних сполучені напрямки отримують наступним чином. Візьмемо довільний напрямок $\mathbf{d}_1$ і на ньому знайдемо мінімум, рухаючись з точки X^1. Повторюємо пошук мінімуму на напрямку $\mathbf{d}_1$ з точки $X^2 \neq X^1$ (рис. 4.17). Напрямок $\mathbf{d}_2$, визначений прямою, що проходить через знайдені мінімуми, є сполученим з напрямком $\mathbf{d}_1$. У той же час напрямок $\mathbf{d}_2$ проходить через шуканий мінімум функції *f*. Отже, з будь-якої відправної точки мінімум квадратичної функції двох змінних досягається за два одномірних пошуки уздовж сполучених напрямків.

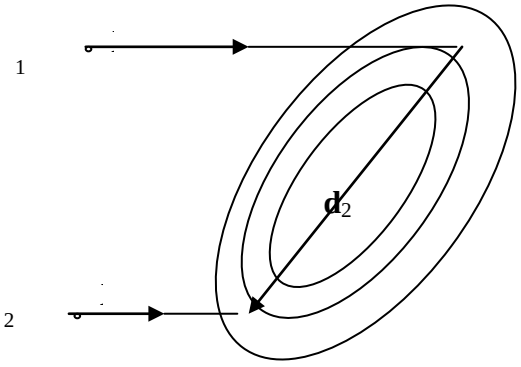


Рис. 4.17. Пошук мінімума функції методом Ньютона

4.11. Методи випадкового пошуку

Методи, що розглядаються тут, базуються на використанні випадкового механізму встановлення початкової точки та вибору напрямку руху. Оскільки під час процесу пошуку розраховуються тільки значення цільової функції, ці методи можна віднести до класу прямих.

Випадковий механізм вибору напрямку здійснюється за допомогою датчика випадкових чисел, що рівномірно розподілені на інтервалі $[-b, b]$. Напрямок встановлюється випадковим вектором

$$\Xi = (\xi_1, \xi_2, \xi_3, ..., \xi_n),$$

компоненти якого обчислюються за формулою:

$$\xi_i = \frac{\beta_i}{\sqrt{\sum_{j=1}^{n} \beta_j^2}},$$

де n випадкових чисел β_i генеруються датчиком. Очевидно, що такий випадковий вектор має одиничну довжину і визначає лише напрямок. У цьому випадку всі напрямки однаково можливі.

Розглянемо приклад простого алгоритму випадкового пошуку.

Алгоритм з поверненням при невдалому кроці:

1. Задати початковий крок h^0, число спроб $s \le n$ та точність ε.
2. Сгенерувати (задати) початкову точку X^0 та обчислити в ній функцію f; прийняти $i = 1$ (i – лічильник спроб).
3. Сгенерувати випадковий вектор напрямку Ξ_i^k.
4. На напрямку Ξ_i^k визначити точку $X^{k+1} = X^k + h^k \cdot \Xi^k$ і обчислити в ній функцію f.
5. Перевірити:

а) якщо $f(X^{k+1}) < f(X^k)$, прийняти $k = k + 1$, $i = 1$ й повернутися на 3;

б) якщо $f(X^{k+1}) \ge f(X^k)$ и $i < s$, прийняти $i = i + 1$ й повернутися на 3;

в) якщо $f(X^{k+1}) \ge f(X^k)$, $i = s$ и $h^k > \varepsilon$, прийняти $h^k = h^k/2$, $i = 1$ й повернутися на 3;

г) якщо $f(X^{k+1}) \ge f(X^k)$, $i = s$ и $h^k \le \varepsilon$, пошук завершити, прийнявши X^k за точку мінімуму.

Таким чином, пошук зупиняється, якщо у поточній точці створені поспіль S напрямків виявилися невдалими на кроці, що менший заданої точності. На рисунку 4.18 показаний характер руху при пошуку екстремуму за цим алгоритмом (жирною лінією виділені успішні кроки).

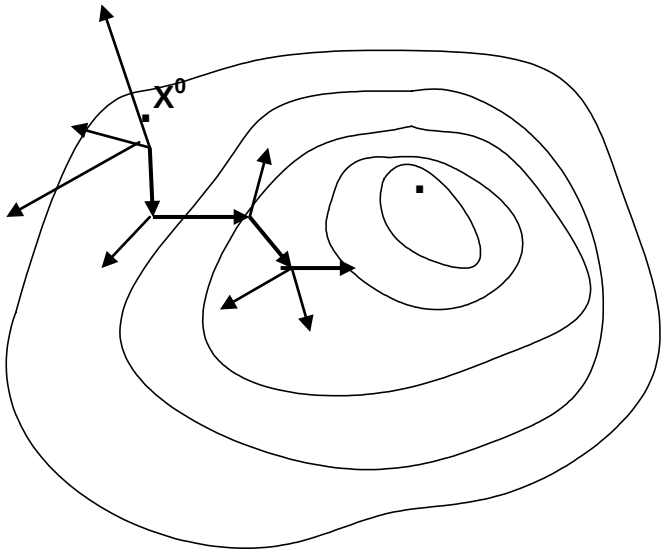


Рис. 4.18. Характер руху при пошуку екстремуму
з поверненням при невдалому кроці

На завершення розгляду методів нелінійного програмування слід відзначити, що, незважаючи на інтенсивні дослідження проблеми глобальної оптимізації, на сьогодні не існує ефективних методів, побудованих на ідеї глобальної поведінки функції. Практичне застосування знаходять в основному підходи, що використовують локальний спуск з багатьох початкових точок з подальшим вибором найкращого рішення. Такий множинний спуск іноді називають *мультистартом*. Йому притамані такі недоліки як можливість повторного спуску в одну й ту ж точку та відсутність гарантії влучення в область глобального мінімуму.

Ефективність мультистарта підвищують за рахунок забезпечення виходу з невеликих мінімумів, виключення повторних спусків в знайдені мінімуми тощо. З цією метою процесу спуску надають інерцію, яка дозволяє прослизнути «неглибокий» мінімум (метод важкої кулі), або змінюють цільову функцію, щоб завдати їй «тунельного ефекту», що забезпечує перехід від знайденого результату до більш глибокого мінімуму, або використовують редукцію задачі та інші ідеї.

Контрольні запитання до розділу 4

1. Які задачі належать до класу задач опуклого програмування?
2. Що означає унімодальність задач опуклого програмування?
3. Які завдання відносяться до класу задач квадратичного програмування?
4. Які завдання відносяться до класу задач геометричного програмування?
5. Які завдання відносяться до кусково-лінійного програмування?
6. Які завдання належать до дробово-лінійного програмування?
7. На чому ґрунтується вирішення задач сепарабельного програмування?
8. Що таке «множина обертання»?
9. Що є основними показниками ефективності ітераційних методів?
10. Які методи «спуску» виділяються за напрямом пошуку?

РЕКОМЕНДОВАНА НАВЧАЛЬНО-МЕТОДИЧНА ЛІТЕРАТУРА

1. Вентцель Е. С. Исследование операций: задачи, принципы, методология. М. : Наука, 1980. 208 с.
2. Дегтярев Ю. И. Исследование операций : учеб. для вузов. М. : Высшая школа, 1986. 320 с.
3. Зайченко Ю. П. Исследование операций : учеб. пособие для студентов вузов. К. : Вища школа, 2006. 392 с.
4. Дослідження операцій у середовищі електронних таблиць Excel : навч. посібник / І. Ю. Лесникова, Н. В. Халіпова, М. В. Терещенко та ін. К. : Центр учбової літератури, 2007. 186 с.
5. Вуколов Э. А. Основы статистического анализа. Практикум по статистическим методам и исследованию операций с использованием пакетов STATISTlCA и EXCEL : учебное пособие. М. : Форум, 2008. 464 с.
6. Дискретне програмування. Методичні вказівки до проведення практичних та самостійних занять з курсу «Дослідження операцій» для студентів факультету кібернетики / упорядн.: В. І. Тюптя, В. І. Шевченко, В. К. Стрюк. К. : Електронна бібліотека факультету кібернетики Київського національного університету ім. Тараса Шевченка, 2019. 35 с.

Навчальне видання

Михайловський Микола Володимирович
Єгоров Олександр Петрович

ДОСЛІДЖЕННЯ ОПЕРАЦІЙ

Навчальний посібник

Тем. план 2022, поз. № 208

Підписано до друку 27.10.2022. Формат 60х84 1/16. Папір друк. Друк плоский.
Облік.-вид. арк. 4,70. Умов. друк. арк. 4,65. Замовлення № 65.

Український державний університет науки і технологій
49010, м. Дніпро, вул. Лазаряна, 2

Редакційно-видавничий відділ УДУНТ