12. МОДЕЛІ В ЗАДАЧАХ УПРАВЛІННЯ

Застосування є головним етапом життєвого циклу моделі. Саме на етапі практичного застосування остаточно з'ясовується її придатність для розв'язання тих задач, для яких вона створена. Можна виділити три головних напрямки застосування моделей:

- для розв'язання задач проектування;
- для оптимізації роботи системи в процесі експлуатації;
- для прогнозування процесів з метою розробки законів керування ними.

12.1 Модель як складова задачі оптимізації

Оптимізація систем може здійснюватися як на етапі проектування, так і на етапі експлуатації. На етапі проектування оптимізують структуру і параметри системи та її підсистем переважно на основі функціонально-вартісних критеріїв (враховують вартість технічних засобів і програмного забезпечення, терміни розробки, які теж пов'язані з витратами, та інші характеристики у контексті забезпечення певного рівня якості роботи системи). На етапі експлуатації оптимізують закон управління. Інколи на цьому етапі теж враховують економічні показники, зокрема, витрати енергії, але переважно оптимізують за технічними показниками: швидкодія, точність, вірогідність тощо.

Масове поширення оптимальних систем останнім часом зумовлене загальною комп'ютеризацію практично усіх систем управління і навіть простих регуляторів. Якщо раніше для створення оптимальних систем були необхідні складні технічні засоби, і кожна оптимальна система була оригінальним винаходом розробників, то наразі розрахунок і формування оптимального управління програмним шляхом доступне у будь-якій системі.

Нагадаємо, що задача оптимізації складається з таких елементів:

- *критерію оптимальності* K, максимум або мінімум якого необхідно забезпечити;
- *характеристик системи* X, за допомогою зміни яких і забезпечується досягнення оптимуму;
- *обмежень на параметри і характеристики системи* $L(X,\Theta_X)$, які повинний задовольняти оптимальний розв'язок;
- *моделі* $F[X,\Theta_X,K]$, яка встановлює залежність між характеристиками системи, параметрами оптимізації і критеріями;
- *умов оптимізації*, які визначають спосіб подання решти компонент постановки задачі.

Критерій оптимальності найчастіше є композицією багатьох характеристик системи. Якби ці характеристики були незалежними, задача пошуку оптимального розв'язку була б тривіальною. Але характеристики реальної системи

пов'язані між собою моделлю F. Залежно від форми подання модель може розглядатися або як доповнення критерію оптимальності, яке дозволяє скоротити кількість змінних, або як додаткове обмеження.

Звичайна постановка задачі оптимізації така: в деякому просторі S тим чи іншим засобом на основі *обмежень* виділяється деяка непуста множина M точок цього простору, яку називають припустимою множиною. Далі фіксується деяка дійсна функція $f(x)$, що задана в усіх точках x допустимої множини. Вона називається цільовою функцією або *критерієм оптимальності*. Задача оптимізації полягає в тому, щоб знайти точку x_0 в множині M, для якої функція $f(x)$ приймає екстремальне (максимальне або мінімальне) значення. В першому випадку для всіх точок x множини M задовольняється нерівність $f(x_0) \geq f(x)$, в другому випадку – нерівність $f(x_0) \leq f(x)$.

В практичних задачах можливі дві основні постановки оптимізаційних задач. В першому випадку задача розглядається в звичайному (евклідовому) просторі кінцевої розмірності. Точками x допустимої множини будуть кортежі $X = (x_1, x_2, ..., x_n)$ дійсних чисел, цільовою ж функцією $f(X) = f(x_1, ..., x_n)$ буде звичайна дійсна функція від n дійсних аргументів. Таку задачу ми будемо називати в подальшому задачею оптимізації функцій. В другому випадку постановки оптимізаційної задачі в якості припустимої множини виступає деяка множина M функцій дійсних змінних $y(x_1, ..., x_m)$, а цільовою функцією є деякий функціонал F, який ставить у відповідність кожній функції $y(x_1, ..., x_m)$ деяке дійсне число $F(y)$. Цю задачу ми будемо називати задачею оптимізації функціоналів або варіаційною задачею.

12.1.1 Класифікація задач оптимізації

Задачі оптимізації відрізняються великим розмаїттям, оскільки кожен з п'яти компонентів постановки задачі може мати декілька принципово відмінних варіантів. Скорочений перелік варіантів постановки задачі наведений на рис. 12.1 у вигляді класифікації задач. Легко підрахувати загальну кількість комбінацій компонентів постановки задач: 4·3·2·3·8=576. І для кожного з цих варіантів існує декілька методів розв'язання.

Перш за все треба розділяти задачі параметричної та структурної оптимізації.

Параметрична оптимізація є предметом, що розглядається в цьому розділі, де наведено постановку такої задачі та методи її розв'язання. Структурна оптимізація – це задача синтезу оптимальної структури системи, причому зміна структур та перетворення однієї структури в іншу здійснюється за спеціальним алгоритмом синтезу. Параметрична оптимізація об'єднує багато різних задач, що мають свої власні особливості та методи розв'язання.

Деталізована класифікація параметричної оптимізації наведена на рис. 12.2. До цього треба додати деякі коментарі:

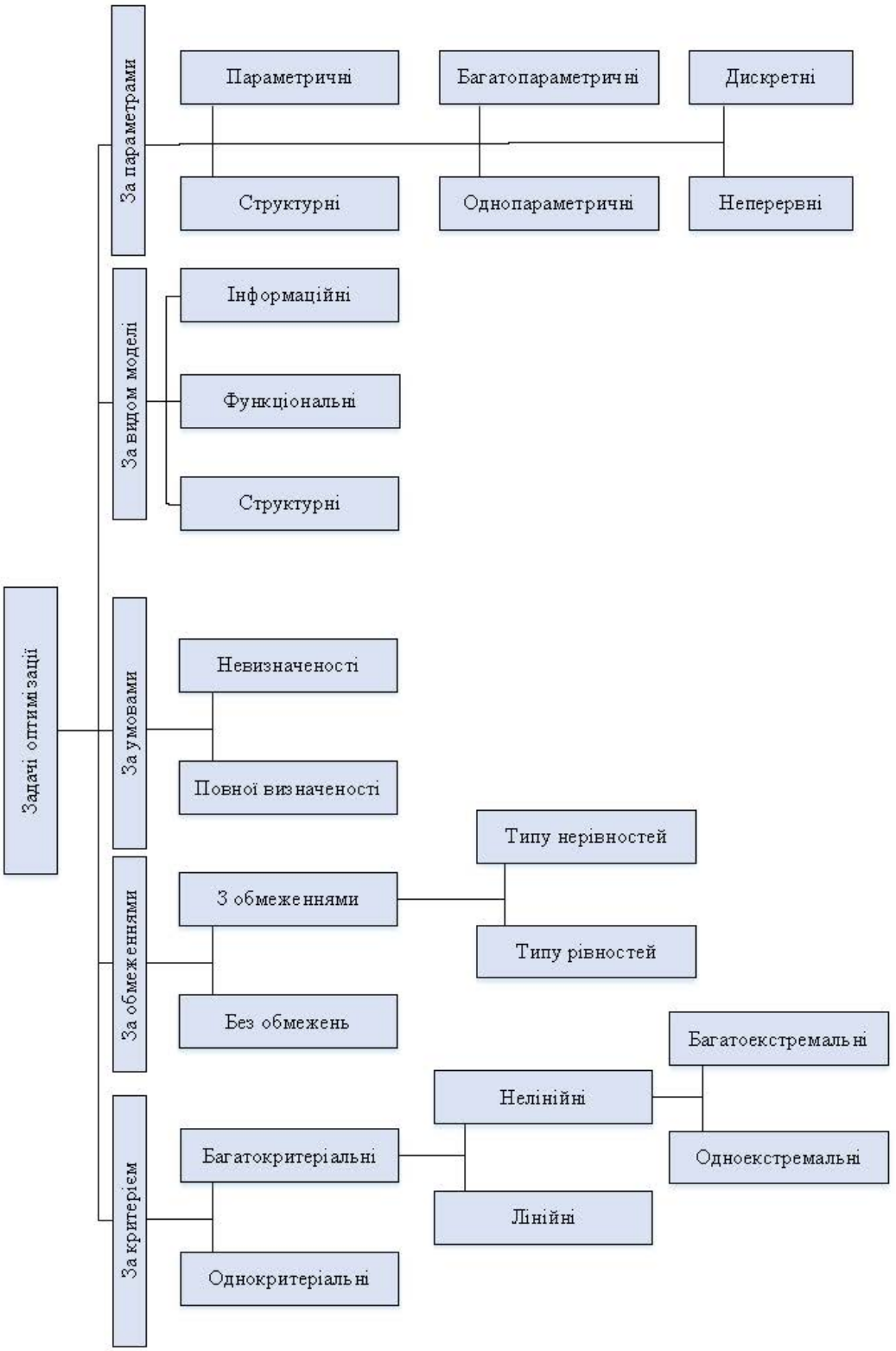


Рисунок 12.1 – Задачі оптимізації

Рисунок 12.2 – Класифікація задач параметричної оптимізації

1. Якщо існує декілька цільових функцій, то має місце задача векторної оптимізації.

2. Якщо кількість параметрів X, що керуються (тобто змінюються для досягнення оптимального значення), більше одного, то розв'язується задача багатопараметричної оптимізації.

3. Якщо існують обмеження та умови, що зв'язують параметри X, то виникає задача оптимізації з умовами, яка в кібернетиці дістала назву математичного програмування.

4. Математичне програмування об'єднує задачі нелінійного програмування (цільова функція в загальному випадку нелінійна), стохастичного програмування (параметри X – випадкова величина, а цільова функція – випадкова функція), динамічного програмування (оптимізація багатокрокових процесів пошуку рішення).

5. Якщо параметри, що керуються, приймають тільки дискретні значення, то виникає задача дискретної оптимізації, а якщо X – цілі числа, то – задача цілочислового програмування.

6. У випадку, коли цільова функція опукла, та область, де задані X, теж опукла, то має місце задача опуклого програмування. Якщо цільова функція та умови лінійні – задача лінійного (кусково-лінійного) програмування; цільова функція квадратична, а умови лінійні – квадратичного програмування; цільова функція та умови – лінійні комбінації функцій однієї змінної – сепарабельного програмування; цільова функція та умови подані у вигляді поліномів – геометричного програмування.

Визначальним для вибору методу оптимізації є вигляд критерію і обмежень, які, в свою чергу, визначаються моделлю системи.

Виділимо лише деякі характерні групи задач.

1) *Одновимірні (однопараметричні) задачі, в яких критерії залежать від однієї характеристики системи (параметра оптимізації).* Можливі варіанти цієї задачі наведені на рис. 12.3.

Крім наведених варіантів можуть існувати ще їх комбінації. Відповідно навіть для такої найпростішої групи задач використовується дуже багато методів пошуку екстремуму. Так наприклад, для випадку, зображеного на рис. 12.3, а (*лінійна залежність*), необхідна проста перевірка, який з кінців інтервалу відповідає максимуму, а який –мінімуму.

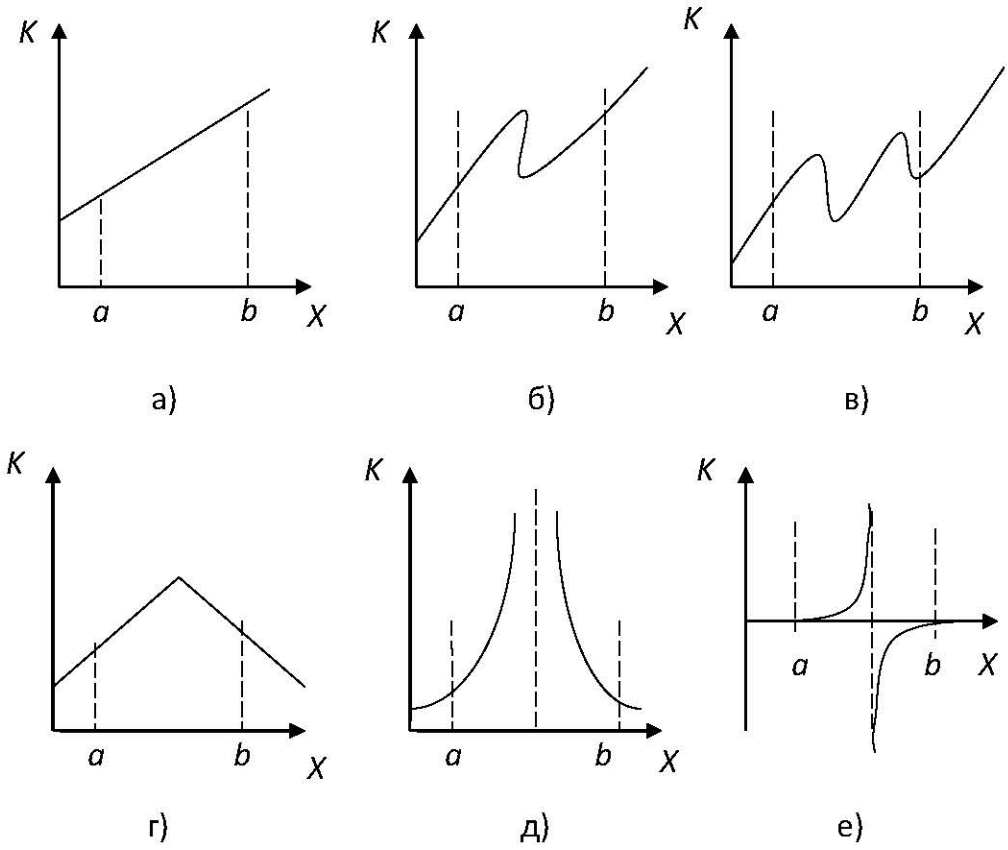
У випадку рис. 12.3, б (*нелінійна диференційовна залежність*) пошук екстремуму здійснюється шляхом диференціювання:

$$\frac{dK}{dx} = 0 \ – \textit{необхідна умова екстремуму,} \qquad (12.1)$$

$$\frac{d^2K}{dx^2} > 0 – \textit{умова мінімуму,}$$

$$\frac{d^2K}{dx^2} < 0 – \textit{умова максимуму,}$$

після чого, як і в усіх інших випадках, ще необхідно порівняти отримане значення $K(x_{опт})$ з кінцями інтервалу $K(a)$ і $K(b)$. У випадку рис. 12.3, в (*багатоекстремальна функція*) необхідна додаткова перевірка, який з наведених екстремумів є глобальним оптимальним рішенням.



Випадки рис. 12.3, г-е відповідають *недиференційованим залежностям* $K(x)$. Для таких задач використовуються різноманітні пошукові (покрокові) методи, в яких однією з головних проблем є визначення початкової точки пошуку та критерію зупинки. Найскладнішим з цієї точки зору є випадок рис. 12.3, е. Тут при пошукові мінімуму і виборі початкової точки на інтервалі $[a,c)$ буде здійснюватись рух в напрямку точки a, що не є правильним рішенням. А при виборі точки на інтервалі $(c,b]$ постає проблема нестійкості алгоритму визначення точки зупинки в околі точки c.

2) *Багатовимірні задачі малої розмірності* (до 5 параметрів оптимізації). Можливі варіанти таких задач аналогічні першій групі з тією різницею, що критерій $K(X)$, зображується поверхнею відповідної розмірності, але перехід від одновимірного до багатовимірного випадку значно ускладнює і, відповідно, урізноманітнює методи оптимізації.

Так, при лінійній залежності $K(X)$ використовується вже ціла група методів лінійного програмування. При унімодальній диференційованій залежності $K(X)$ рівняння (12.1) перетворюється на систему рівнянь у частинних похідних,

яку часто важко розв'язати, тому використовуються різноманітні градієнтні методі. У більш складних випадках використовуються методи направленого (детермінованого) пошуку з тими ж проблемами, що й для першої групи, але ускладненими багатовимірностями.

3) *Багатовимірні задачі середньої розмірності* (від 6 до 10 параметрів оптимізації). Для цієї групи задач переваги набувають методи направленого пошуку. Методи прямих обчислень оптимального значення вже важко застосувати через велику розмірність систем рівнянь, до яких вони приводять. Але трудомісткість направленого пошуку зростає пропорційно $\prod_{i=1}^{n} m_i$ (де n – кількість факторів, m_i – кількість можливих значень параметрів оптимізації) і при $n > 10$ навіть для сучасних комп'ютерів стає надто високою.

4) *Багатовимірні задачі великої розмірності* (більше 10 факторів). Для таких задач незаперечної переваги набувають методи випадкового пошуку. Останнім часом ці методи набули великої популярності і бурхливого розвитку. З'явилося безліч модифікацій методу Монте-Карло, генетичні мурашині інші алгоритми.

5) *Задачі, в яких аргументами критерію оптимальності є функція.* Методи розв'язування таких задач суттєво залежать від характеру обмежень, що є складовою задачі оптимізації. Найпоширенішим методом є класичний варіаційний метод на основі невизначених множників Лагранжа, метод на основі "принципу максимуму" Понтрягіна, динамічне програмування на основі принципу Беллмана та інші.

6) *Задачі дискретної оптимізації, що ґрунтуються на графових моделях.* Методи розв'язання цих задач мають переважно комбінаторний характер з різноманітними модифікаціями, направленими на зменшення кількості варіантів.

Роль моделі при розв'язанні задачі оптимізації особливо наочна при використанні методу невизначених множників Лагранжа у поєднанні з теоремою Куна-Таккера. Відповідно до цього методу задачу пошуку екстремуму функції $K(x)$ при обмеженнях $g_i(x) = 0, \quad i = 1,...,n,$, які подають модель системи, замінюють на задачу пошуку екстремуму функції Гамільтона більшої розмірності, але без обмежень. Функція Гамільтона поєднує критерій оптимізації з моделлю

$$H(x,\lambda) = K(x) - \sum_{i=1}^{n} \lambda_i g_i(x),$$

де λ_i – невизначені множники Лагранжа, які є додатковими параметрами оптимізації.

12.1.2 Багатокритеріальна оптимізація

На практиці часто виникає випадок, коли замість однієї цільової функції $f(x)$ задано декілька цільових функцій $f_1(x),...,f_R(x)$. Така задача багатокри-

теріальної оптимізації має декілька постановок. В одній з них потрібно оптимізувати один з критеріїв, припустимо, $f_1(x)$, причому решту (R-1) критеріїв утримують в заданих межах: $a_i \le f_i(x) \le b_i$ $(i = 2,3,...,k)$. В цьому разі фактично йдеться про звичайну однокритеріальну оптимізацію. Що ж до нерівностей, які обмежують інші критерії, то їх можна розглядати як додаткові обмеження на припустиму область M.

В іншому випадку постановка полягає в упорядкуванні заданої множини критеріїв та послідовній оптимізації за кожним з них. Інакше, якщо проводять оптимізацію за першим критерієм $f_1(x)$, то одержують деяку множину $M_1 \subset M$, на якій функція $f_1(x)$ приймає оптимальне (екстремальне) значення. Прийнявши його за нову допустиму множину, проводять оптимізацію за другим критерієм та одержують в результаті нову допустиму множину $M_2 \subset M_1$. Якщо продовжити цей процес, то можна одержати після оптимізації за останнім критерієм $f_R(x)$ множину M_R, яка і буде кінцевим результатом багатокритеріальної оптимізації. Звідси, якщо на деякому кроці i $(i < k)$ множина M_1 зведеться до однієї точки, процес оптимізації можна буде закінчити, оскільки $M_i = M_{i+1} = ... = M_k$. Зрозуміло, що як і в випадку звичайної однокритеріальної оптимізації, задача може взагалі не мати розв'язку.

Третя постановка застосовує процес зведення багатьох критеріїв до одного за рахунок введення апріорних вагових коефіцієнтів λ_i для кожного з критеріїв $f_i(x)$. Як і коефіцієнти можуть бути вибрані будь-які дійсні числа. Їх значення вибирають, виходячи з інтуїтивного подання ступеня важливості різних критеріїв: більш важливі критерії одержують ваги з більшими абсолютними значеннями. Після встановлення ваг λ_i багатокритеріальна задача зводиться до однокритеріальної з цільовою функцією $f(x) = \lambda_1 f_1(x) + ... + \lambda_R f_R(x)$.

Замість простої лінійної комбінації вхідних критеріїв можуть використовуватися і більш складні засоби формування з них нового критерію.

В практиці проектування великих систем і управління такими системами, як правило, використовується багато критеріїв. В ряді випадків їх вдається в той чи іншій спосіб звести до одного критерію і тим самим повернутися до вже дослідженого випадку однокритеріальної оптимізації.

Найпростіший спосіб такого зведення – так зване зважування критеріїв. Якщо $f_1(x),..., f_n(x)$ – це функції, що виражають значення використовуваних критеріїв, то кожній з них, відповідно до відносної важливості критеріїв, вибирається додатний ваговий коефіцієнт λ_i. Операція зважування критеріїв (цільо-

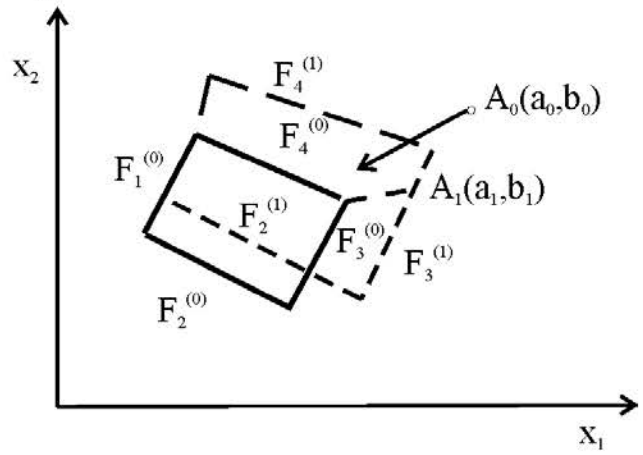
вих функцій) $f_1(x),...,f_n(x)$ полягає в заміні їх єдиним критерієм (цільовою функцією) $f(x) = \lambda_1 f_1(x) + ... + \lambda_n f_n(x)$.

Але для багатьох задач, що пов'язані з великими системами, подібне зведення виявляється практично неможливим, так що в процесі оптимізації доводиться мати справу з векторною (багатокритеріальною) цільовою функцією. При цьому припустима область M може змінюватись в процесі оптимізації. Більше того, в її цілеспрямованій зміні як раз і полягає основна змістовна сутність процесу оптимізації для подібного класу задач.

Наведемо одну з характерних формалізованих постановок задачі системної оптимізації.

Розв'язок відшукується безпосередньо в просторі K критеріїв оптимізації, які ми позначимо x_1 і x_2. Процес розв'язання починається з того, що в заданому просторі K вибирається деяка точка A_0 з координатами a_0, b_0 – бажаний розв'язок задачі. За цим будуються початкові обмеження $F_1^{(0)}(x_1, x_2) \geq 0,...,F_n^{(0)}(x_1, x_2) \geq 0$, що задають початкову припустиму область P_0. Прямою перевіркою встановлюється, чи належить точка області P_0. В першому випадку в принципі може бути застосована звичайна (класична) процедура оптимізації або за одним з критеріїв x_1, x_2, або за тією чи іншою їх комбінацією.

Але при системному підході застосовується, зазвичай, запропонований Л. С. Понтрягіним спосіб, а саме: відповідно до моделі M вищого рівня, що управляє вибором критеріїв, точка A_0 виводиться з меж припустимої області P_0, як це показано на рис. 12.4.



Після цього виділяються ті обмеження, які не виконуються в точці A_0 (в випадку, що розглядається, ними будуть $F_3^{(0)}$ і $F_4^{(0)}$). Звертаючись до моделей

M_3 і M_4, які формують ці обмеження, в діалоговому режимі опробовуються ті чи інші рішення, що змінюють відповідні обмеження в потрібному напрямку (якщо така зміна можлива). Оптимальним при цьому вважається той напрямок, що зменшує абсолютну величину від'ємних відхилень $F_i^0(a_0, b_0)$ (у випадку, який розглядається, $F_3^0(a_0, b_0)$ і $F_4^0(a_0, b_0)$.

12.1.3 Гладка оптимізація

У випадку, коли функція цілі $f(x)$ і функції $p_i(x)$, які задають обмеження, є диференційованими (гладкими), для розв'язання задач оптимізації використовується знаходження градієнта. Для будь-якої функції $g(x)$, що диференціюється, її градієнтом $\nabla g(x)$ в точці x називається вектор

$$\nabla g(x) = \begin{pmatrix} \dfrac{\partial g(x)}{\partial x_1} \\ \ldots \\ \dfrac{\partial g(x)}{\partial x_n} \end{pmatrix}.$$

Вектор градієнта $\nabla g(x)$ задає (в даній точці x) напрям найшвидшого росту функції $g(x)$, а зворотний йому напрям $-\nabla g(x)$, що називається антиградієнтом, напрям найшвидшого спадання цієї функції. Точки, в яких градієнт функції перетворюється в нуль, називаються її стаціонарними точками. Якщо екстремум функції $f(x)$ досягається всередині припустимої області (не на її межі), то в точці оптимуму $x = a$ її градієнт перетворюється в нуль, тобто має місце система рівнянь

$$\left.\frac{\partial f(x)}{\partial x_1}\right|_{x=a} = 0, \ldots, \left.\frac{\partial f(x)}{\partial x_n}\right|_{x=a} = 0.$$

Ці рівняння (умови стаціонарності функцій) мають місце не тільки для абсолютних, але й для локальних екстремумів. Разом з тим умови стаціонарності не є достатніми.

Умови Куна-Такера

У випадку, коли екстремальна точка a лежить на границі припустимої множини, вона не обов'язково є стаціонарною. Але за деякої додаткової умови може, якщо замінити цільову функцію $f(x)$ так званою функцією Лагранжа

$L(x) = f(x) + \sum_{i+1}^{m} \lambda_i p_i(x)$, (де $p_i(x)$ – ліві частини всіх граничних умов, досягти того, щоб граничні екстремальні точки функції $f(x)$ були стаціонарними точками функції Лагранжа $L(x)$ при відповідних значеннях параметрів λ_i.

Умова, про яку йдеться, є так званою умовою регулярності – виконується на практиці. Вона полягає в лінійній незалежності в даній точці $x = a$ градієнтів всіх мережних функцій $p_i(x)$, для яких $p_i(a) = 0$.

При виконанні цієї умови для будь-якої точки максимуму x^* в задачі "знайти max $f(x)$ при $p_i(x) = 0,\ (i = 1,...,l)$, $p_i(x) \geq 0,\ (i = l+1,...,m)$" і для будь-якої точки мінімуму x^* в задачі "знайти min $f(x)$ при $p_i(x) = 0,\ (i = 1,...,l)$, $p_i(x) \leq 0,\ (i = l+1,...,m)$" повинні виконуватися такі три умови, які називаються *умовами Куна-Таккера:*

1) x^* лежить в припустимій множині;

2) $\lambda_i p_i(x^*) = 0$ при $i = 1,...,m$;

3) $\nabla f(x^*) + \sum_{i+1}^{m} \lambda_i \nabla p_i(x^*) = 0$.

де λ_i – деякі дійсні числа (множники Лагранжа), довільні для всіх умов типу рівності $(i = 1,...,l)$ і невід'ємні для всіх умов типу нерівності $(i = l+1,...,m)$.

Чисельні методи гладкої оптимізації

Метод оптимізації, який використовує умови Куна-Таккера, на практиці застосовується відносно рідко через велику складність аналізу системи співвідношень, що виникає. Найчастіше застосовуються чисельні методи оптимізації, для яких достатньо вміти знаходити чисельні значення градієнта в будь-якій заданій точці.

Загальна ідея методу градієнтного спуску (підйому)

Нехай неперервно диференційовна цільова функція $f(x)$ задана в усіх точках простору X. Вирушаючи від точки x на дуже малий крок ε в будь-якому напрямку d, де d – одиничний вектор, в силу умови диференційованості функції $f(x)$. Одержимо нерівності:

$$f(x + \varepsilon d) \geq f(x) \geq f(x - \varepsilon d).$$

Це означає, що рух (на дуже малий крок) в напрямку градієнта функції забезпечує найбільший підйом, а в напрямку антиградієнта – найбільше спадання

цієї функції. Ці напрямки називають відповідно напрямком найшвидшого підйому і найшвидшого спуску функції $f(x)$ в даній точці x.

Виходячи з заданої точки x^0, будуємо послідовність точок $(x^0, x^1, x^2, ...)$ так, що переміщення від кожної точки x^{i-1} до наступної точки x^i проводиться в напрямку найшвидшого підйому (при пошуку максимуму) або найшвидшого спуску (при пошуку мінімуму). Важкість практичного застосування такої процедури полягає перш за все у виборі величини кроку ε_i при переході від точки x^{i-1} до точки x^i. Справа в тім, що нерівності мають місце лише в малому околі точки x, тобто для малої величини кроку ε_i. При великому значенні кроку можна, прямуючи в напрямку найшвидшого підйому, одержати не збільшення, а зменшення значення цільової функції $f(x)$. Те ж саме має місце і для процедури спуску. Отже, величину кроку ε_i в процедурах найшвидшого підйому і спуску необхідно вибирати достатньо малою. Проте при прямуванні малими кроками для наближення до екстремальної точки може знадобитись дуже велика кількість кроків. Крім того, від процедур підйому і спуску звичайно вимагається не просто наближення до екстремальної точки $x = a$, а збіжність до неї. Це означає, що зі збільшенням кількості кроків i відстань $\left|x^i - a\right|$ від точки x^i до точки екстремуму $x = a$ повинна наближатися до нуля. Це можливо лише в тому випадку, якщо з наближенням до точки екстремуму величина кроку ε_i буде наближатись до нуля. На рис. 12.5 наведений алгоритм пошуку екстремуму функції за методом градієнтного спуску.

Пропорційно-градієнтний метод

Найбільш простим способом забезпечення необхідного зменшення кроку при наближенні до точки екстремуму для диференціальної функції $f(x)$ є вибір довжини кроку ε_i пропорційним довжині вектора градієнта в точці x^{i-1}:

$$\varepsilon_i = \alpha\left|\nabla f(x^{i-1})\right|, \quad \alpha > 0, \quad \alpha = const.$$

Такий вибір кроку означає, що перехід від точки x^{i-1} до точки x^i буде виконуватись за формулою $x_i = x^{i-1} + \alpha\nabla f(x^{i-1})$ для процедури підйому (знаходження максимуму функції $f(x)$) і за формулою $x_i = x^{i-1} + \alpha\nabla f(x^{i-1})$ для процедури спуску (знаходження мінімуму функції $f(x)$).

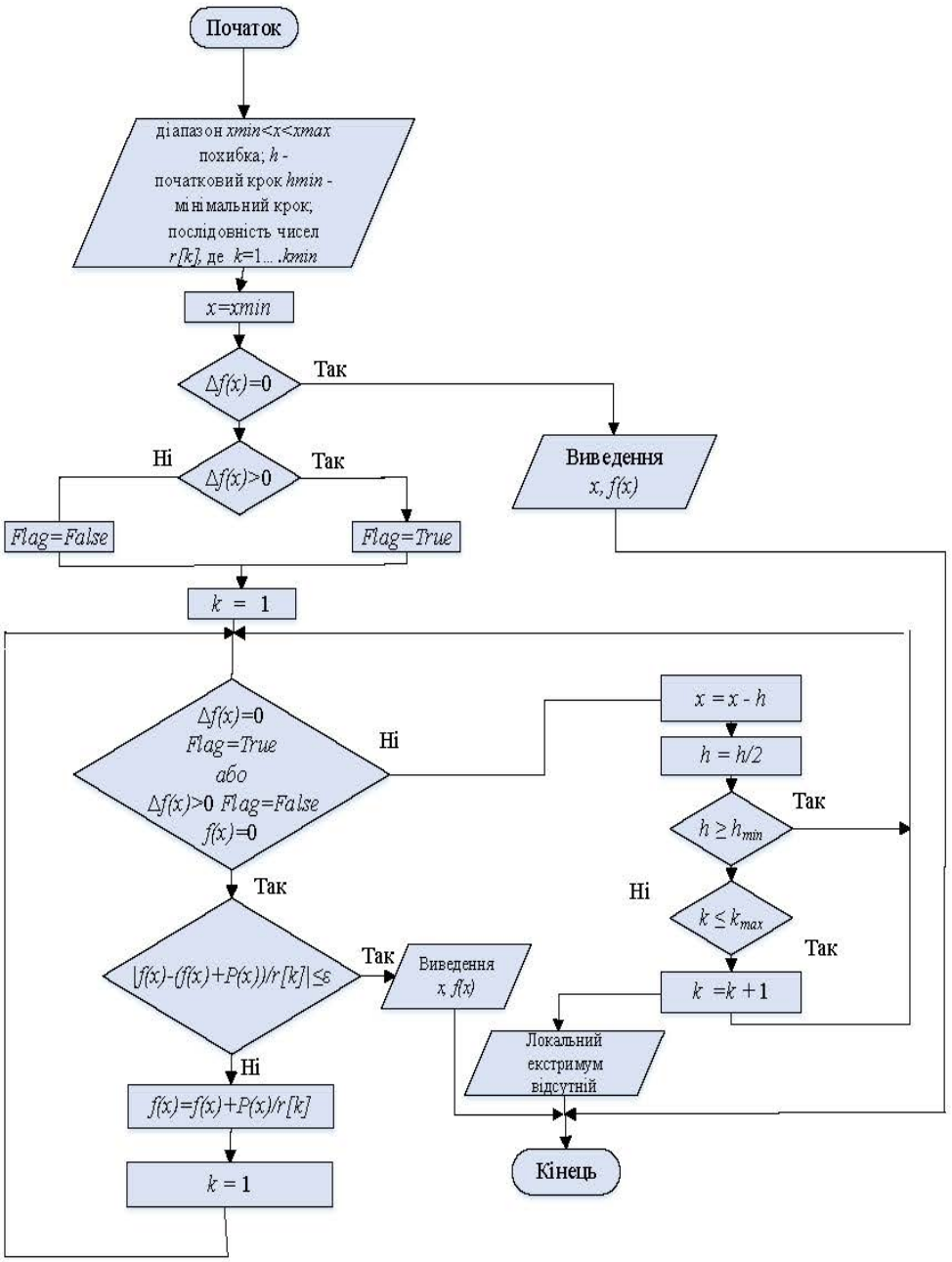


Рисунок 12.5 – Алгоритм методу градієнтного спуску (підйому)

Повнокроковий градієнтний метод

В цьому методі кожний крок градієнтного підйому або спуску виконується максимально можливої довжини, яка забезпечує необхідний напрямок зміни значення функції (тобто її збільшення або зменшення). Інакше кажучи, на півпрямій, що виходить з чергової точки x^{i-1} в напрямку градієнта (при підйомі) або

антиградієнта (при спусканні) відшукується точка абсолютного максимуму або мінімуму, яка і вибирається як наступна точка x^i.

Метод спряжених градієнтів

Для випадку, коли функція $f(x)$ – квадратний поліном, розроблена точна процедура, при повних кроках якої точки екстремуму досягаються з будь-якої початкової точки x^0 за n кроків (де n – розмірність простору). Напрямок a^{i+1} кроку від точки x^i до точки x^{i+1} задається в цьому методі такою рекурентною формулою:

$$a^{i+1} = \pm\nabla f(x^i) + \frac{\left|\nabla f(x^i)\right|^2}{\left|\nabla f(x^{i-1})\right|^2} a^i, \quad i = 1,2,...,$$

де $\left|\nabla f(x)\right|$ означає довжину вектора $\nabla f(x)$.

Знак плюс вибирається в тому випадку, коли розглядається задача максимізації, а знак мінус – у випадку розв'язання задачі мінімізації. Такий же вибір знака здійснюється і для напрямку a^1 першого кроку $(i = 0)$. Воно збігається з напрямком $\pm\nabla f(x^0)$.

Методи зведення загальної задачі оптимізації до задачі без обмежень

Чисельні методи гладкої оптимізації, які описані вище, в чистому вигляді застосовуються звичайно для задач без обмежень. Задачі загального вигляду (тобто задачі з обмеженнями) зводять до задач без обмежень за рахунок зміни цільової функції. Нехай, наприклад, потрібно знайти максимум функції $f(x)$ при обмеженнях $p_i(x) \geq 0 \quad (i = 1,..., m)$. В так званому методі штрафних функцій будується функція $P(x)$, яку називають штрафною функцією і яка всередині припустимої області M приймає нульове значення, а поза нею – від'ємне. Найчастіше за таку функцію вибирають функцію вигляду

$$P(x) = -\sum_{i=1}^{m} \left|\min\left[p_i(x), 0\right]\right| \quad l \geq 1,$$

після чого будують послідовність додатних чисел $r_1 > r_2 > ... > r_k > ...$, яка збігається до нуля.

При заміні цільової функції $f(x)$ функцією

$$F_k(x) = f(x) + \frac{1}{r_k} P(x), \quad k = 1,2,...$$

розв'язують задачу без обмежень для цієї функції, починаючи з $k = 1$. Якщо розв'язок x^k належить припустимій області, тобто якщо всі $g_k(x^k) \geq 0$, то x^k є

розв'язком початкової задачі з обмеженням. В протилежному випадку замінюють k на $k+1$ і розв'язують задачу для нової цільової функції $F_k(x)$.

Алгоритм розв'язання такої задачі наведений на рисунку 12.6.

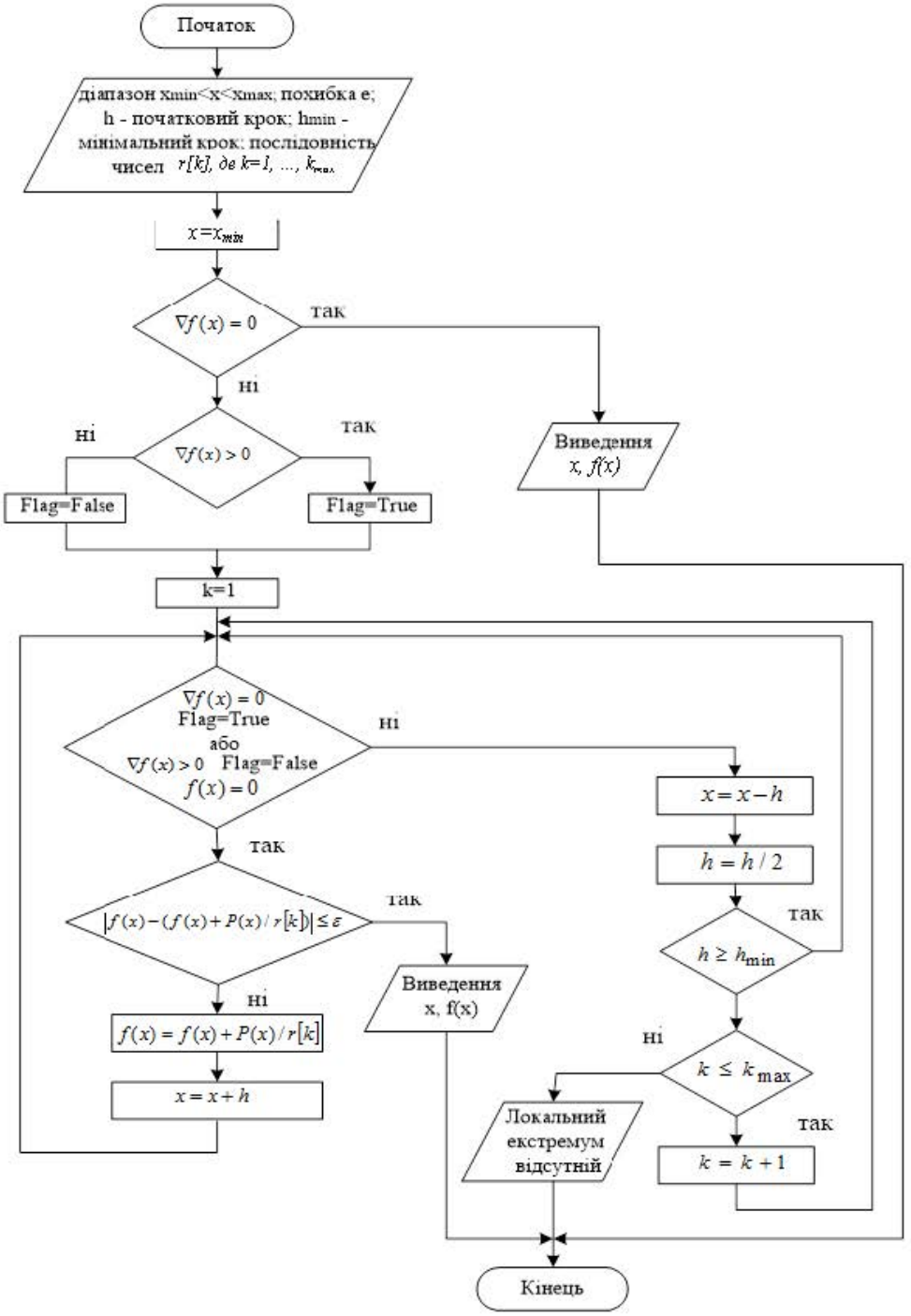


Рисунок 12.6 – Алгоритм методу штрафних функцій

В методі бар'єрів при розв'язанні задачі max $f(x)$ при $p_i(x) \geq 0 \ \ (i = 1,...,m)$ цільову функцію $f(x)$ замінюють функцією $F_k(x) = f(x) + r_k B(x), \ r_k > 0$, де бар'єрна функція $B(x)$ характеризується властивістю прямувати до $-\infty$ при наближенні до границі припустимої області M зсередини. Таку функцію можна задати, наприклад, формулою:

$$B(x) = -\sum_{i=1}^{m} \frac{1}{p_i(x)},$$

якщо область M задана нерівностями $p_i(x) \geq 0 \ \ (i = 1,...,m)$.

Таким же чином, як і в методі штрафних функцій, вибирають послідовність $\{ r_k \}$ додатних чисел $r_1 > r_2 > ... > r_k > ...,$ яка збігається до нуля.

Розв'язуються задачі без обмежень для цільових функцій $F_k(x) = f(x) + r_k B(x)$ при $k = 1,2,...$

Якщо при цьому використовується той або інший чисельний метод, в якому початкова точка вибирається всередині M, то наявність бар'єра забезпечить належність екстремальної точки x^k (при її існуванні) області M. Наявність екстремуму забезпечується умовами обмеженості в області M і неперервності функцій $f(x)$ і $B(x)$.

12.1.4 Опукла оптимізація

Задача оптимізації може бути значно спрощена при накладанні на цільову функцію $f(x)$ і припустиму область M деяких обмежень. Одним з таких обмежень є так звана умова опуклості.

Множина M евклідового простору називається опуклою, якщо для будь-яких двох точок x^1 і x^2 цієї множини всі точки відрізку $[x^1, x^2]$, що їх з'єднує, також належать множині M.

Функція $f(x)$ опукла, або, точніше, опукла донизу, якщо вона визначена на опуклій множині M і для будь-яких двох точок x^1 і x^2 цієї множини значення $f(x)$ функції в будь-якій точці x відрізка $[x^1, x^2]$, не перевищують значення в тій же точці, що визначена на даному відрізку лінійної функції із значеннями $f(x_1)$ і $f(x_2)$ на його кінцевих точках. Це, в загальному випадку, виражається нерівністю

$$\alpha f(x^1) + (1-\alpha) f(x^2) \geq f(\alpha x^1 + (1-\alpha) x^2) \text{ при } 0 \leq \alpha \leq 1.$$

Очевидно, що при $0 \leq \alpha \leq 1$ вираз $\alpha x^1 + (1-\alpha) x^2$ задає різні точки відрізка $[x^1, x^2]$.

Для опуклої функції $f(x)$ в будь-якій точці x^0, що лежить в області визначення M функції, існує вектор $r(x^2)$, для якого при будь-якому x і M виконується нерівність $f(x)-f(x^0)\geq r(x^0)\cdot(x-x^0)$.

В правій частині цієї нерівності стоїть скалярний добуток векторів $r(x^0)$ і $x-x^0$, тобто добуток довжин цих векторів на косинус кута між ними. Вектор $r(x^0)$, що задовольняє цю властивість, називається узагальненим градієнтом (субградієнтом) функції $f(x)$ в точці x^0. Звичайний градієнт (у випадку його існування і відмінності від нуля) є також узагальненим градієнтом, причому єдиним, в розглядуваній точці. Якщо функція $f(x)$ не диференційовна в деякій точці x^0, в ній існує деяка множина узагальнених градієнтів, яку називають субградієнтною множиною функції $f(x)$ в точці x^0.

Простий субградієнтний метод опуклої оптимізації

Для реалізації субградієнтної оптимізації в випадку задачі без обмежень процес може починатися з будь-якої точки x^0. Зсув чергової точки x^j до наступної точки x^{j+1} здійснюється на відстань l_i в напрямку будь-якого узагальненого градієнта функції $f(x)$ в точці x^j в випадку задачі максимізації вгнутої функції і в зворотному напрямку – у випадку задачі мінімізації опуклої функції. Якщо кроки l_i зсувів обираються таким чином, що $l\to 0$ при $i\to\infty$, а ряд $l_0+l_1+l_2+...$ розбігається, то послідовність $\{x^0,x^1,...,x^m,...\}$ збігається до множини екстремуму заданої функції $f(x)$ за умови, що множина обмежена.

Для обчислення субградієнтів застосовуються різні способи. Одним з них є зведення задачі обчислення субградієнта функції, що задана складним виразом, до обчислення субградієнтів окремих компонентів цього виразу. Для подібного зведення особливо часто застосовуються такі властивості опуклих функцій.

Властивість 1. Якщо функції $f_1(x),...,f_m(x)$ опуклі, то опуклою буде і будь-яка їх лінійна комбінація $f(x)=\sum_{i=1}^{m}a_i f_i(x)$ з невід'ємними коефіцієнтами $a_i(a_i\geq 0, i=1,...,m)$, а субградієнт $S(x)$ функції $f(x)$ дорівнює лінійній комбінації $\sum_{i=1}^{m}a_i S_i(x)$ субградієнтів $S_i(x)$ функцій $f_i(x)$.

Властивість 2. Якщо функції $f_1(x),...,f_m(x)$ опуклі, то опуклою буде і функція $f(x)=\max f_i(x)$, і всі субградієнтні множини $S_i(x^0)$ функцій $f_i(x)$, для яких $f(x^0)=f(x)$, входять в субградієнтну множину $S(x^0)$ функції $f(x)$ в будь-якій заданій точці $x=x^0$.

Методи розтягання простору

Простий субградієнтний метод, що розглянутий в попередньому розділі, може виявитися повільно збіжним, якщо області рівня цільової функції сильно витягнуті в одному напрямку, або, як часто при цьому говориться, мають вигляд вузьких і довгих ярів. Ця ситуація характерна для багатьох задач негладкої оптимізації. Для поліпшення збіжності методу в подібних випадках використовують методи розтягання простору.

Розтягання простору зводиться до заміни змінних $x_1,...,x_k$ в цільовій функції $f(x)=f(x_1,...,x_n)$ деякими їх лінійними комбінаціями. Особливо просто виконується розтягання в напрямку однієї з координатних осей. Змінна x_i, що відповідає цій осі, замінюється при цьому на $k^{-1}x_i$, де k – коефіцієнт розтягання, а решта залишається без змінювання.

12.1.5 Негладка оптимізація за методом координатного спуску (підйому)

При оптимізації недиференційованих функцій можна відмовитись від будь-яких узагальнень понять градієнта та використовувати лише кроки вздовж осей координат. З цією метою досліджують значення цільової функції $f(x_1,...,x_n)$ на черговому кроці оптимізації з приростом $\pm\delta_i$ однієї з координат і переходять від точки $(x_1,...,x_n)$ до точки $(x_1,...,x_{i-1},x_i\pm\delta_i,x_{i+1},...,x_n)$. Знак приросту вибирають таким чином, щоб значення функції $f(x_1,...,x_n)$ змінювалось в потрібному напрямку (збільшувалось при максимізації і зменшувалось при мінімізації). Величини δ_i вибирають звичайно як в покрокових методах та забезпечують максимально можлива зміна функції в обраному напрямку.

Зсуви здійснюються по черзі по всіх осях координат, причому після зсуву по останній осі x_n знову повертаються до першої осі x_1. Якщо повторити цей процес достатньо багато разів, як правило, можна знайти як завгодно точне наближення деякого екстремуму заданої негладкої функції в тому разі, коли виконується припущення, що такий екстремум існує.

12.1.6 Стохастична оптимізація

Іноді корисно замінити зсуви вздовж координатних осей зсувами (з відповідним знаком) вздовж випадково обраного на кожному кроці напрямку. При достатньо загальних припущеннях внаслідок таких зсувів з імовірністю одиниця за достатньо велике число кроків екстремальна точка може бути знайдена з будь-яким необхідним ступенем точності. Перевагою методу є простота кожного окремого кроку оптимізації. Методи випадкового пошуку дозволяють знаходити глобальний екстремум багатоекстремальних функцій. Більшість з них не має обмежень на вигляд критерію оптимізації, таких як вимога диференційовності (як у градієнтних методів).

Для невеликої кількості параметрів оптимізації (звичайно 1–5 параметрів) кількість кроків, порівняно з методами, які забезпечують пошук задовільних напрямків руху, у подібних простих методах, як правило, значно більша. Проте для великої кількості параметрів (більше 10) методи випадкового пошуку забезпечують швидший результат. Для середньої кількості параметрів (5–10) перевага у швидкості детермінованих або стохастичних методів залежить від конкретного вигляду критерію оптимізації.

Простий випадковий пошук

Нехай нам необхідно вирішити задачу мінімізації функції $f(\bar{x})$ за умови, що $\bar{x} \in [\bar{A}, \bar{B}]$.

У даній області за рівномірним розподілом ймовірності обираємо випадкову точку $\bar{x}_1$ (рис. 12.7) і обчислюємо в ній значення функції $y_1 = f(\bar{x}_1)$. Потім вибираємо таким же чином випадкову точку $\bar{x}_2$ і обчислюємо $y_1 = f(\bar{x}_2)$. Запам'ятовуємо мінімальне з цих значень і точку, в якій значення функції мінімально. Далі генеруємо нову точку. Робимо N експериментів, після чого кращу точку беремо як розв'язок задачі (в якій функція має мінімальне значення).

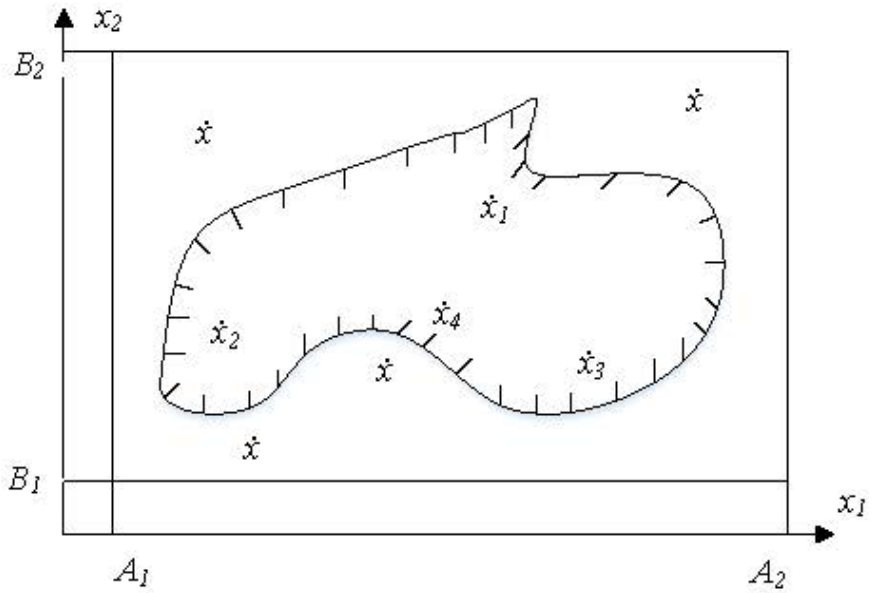


Рисунок 12.7 – Простий випадковий пошук

Ймовірність потрапляння в цей окіл при одному випробуванні дорівнює $P_в = \frac{V_в}{V}$, де $V = \prod_{i=1}^{n}(B_i - A_i)$ $V_в = \prod_{i=1}^{n} \varepsilon_i$. Імовірність непотрапляння дорівнює

$1-P_\varepsilon$. Випробування незалежні, тому ймовірність непотрапляння за N експериментів дорівнює $(1-P_\varepsilon)^N$.

Ймовірність того, що ми знайдемо розв'язок за N випробувань: $P=1-(1-P_\varepsilon)^N$.

Звідси неважко отримати оцінку необхідного числа випробувань N для визначення мінімуму з необхідною точністю:

$$N \geq \frac{\ln(1-P)}{\ln(1-P_\varepsilon)}.$$

При розв'язанні екстремальних задач на областях зі складною геометрією зазвичай вписують цю область в *n*-вимірний паралелепіпед. А далі генерують в цьому *n*-вимірному паралелепіпеді випадкові точки за рівномірним закон, залишаючи тільки ті, які потрапляють в допустиму область.

Розрізняють ненаправлений випадковий пошук без самонавчання, направлений випадковий пошук, направлений пошук з самонавчанням.

Ненаправлений випадковий пошук (або метод статичних випробувань, метод Монте-Карло) полягає в багаторазовому моделюванні незалежних випадкових варіантів рішень з області допустимих, обчисленні в кожному з них критерію оптимізації і запам'ятовуванні найближчого до екстремуму. Метод Монте-Карло відносять до числа універсальних, оскільки він дозволяє вирішувати багатоекстремальні завдання загального вигляду з пошуку глобального екстремуму. Основний недолік методу полягає в необхідності проведення великого числа випробувань для знаходження розв'язку.

Направлений випадковий пошук без самонавчання є модернізацією ненаправленого випадкового пошуку. В основі методу лежить використання результатів пошуку попередніх кроків.

Направлений випадковий пошук з самонавчанням полягає в додаванні алгоритмів самонавчання, які коригують вектор передісторії (випадковий вектор *E*) переважно в напрямку вдалого попереднього кроку або в напрямку, зворотному попередньому невдалому кроці, у разі нелінійної цільової функції, наприклад шляхом перебудови ймовірностей P_1, ... , Pi, Pn складових випадкового вектору $E = (e_1, ..., e_i, ..., e_n)$. Так, при цільовій функції, близькій до лінійної, ймовірність кроку у вдалому напрямку збільшується, і навпаки.

Алгоритм парної проби. У даному алгоритмі чітко розділені пробний і робочий кроки.

Нехай x^{-k} – знайдене на *k*-му кроці найменше значення функції $f(\bar{x})$, що мінімізується. За рівномірним законом генерується випадковий одиничний вектор $\bar{\xi}$ і по обидві сторони від вихідної точки x^{-k} робляться дві проби: проводи-

мо обчислення функції в точках $\bar{x}_{1,2}^{k} = x^{-k} \pm g \cdot \bar{\xi}$, де g - величина пробного кроку.

Робочий крок робиться в напрямку найменшого значення цільової функції. Чергове наближення визначається співвідношенням

$$x^{-k+1} = x^{-k} + \Delta x^{-k} = \bar{x}^{k} + \alpha \cdot \bar{\xi} \cdot sign(f(x^{-k} - g\bar{\xi}) - f(x^{-k} + g\bar{\xi}).$$

Особливістю даного алгоритму є його підвищена тенденція до "блукання". Навіть знайшовши екстремум, алгоритм може відвести процес пошуку в сторону.

Алгоритм найкращої проби. На k-му кроці ми маємо точку x^{-k}. Генерується m випадкових одиничних векторів $\bar{\xi}_1,...,\bar{\xi}_m$. Робляться пробні кроки в напрямках $g \cdot \bar{\xi}_1,..., g \cdot \bar{\xi}_m$ і в точках $x^{-k} + g \cdot \bar{\xi}_1,..., x^{-k} + g \cdot \bar{\xi}_m$ обчислюються значення функції. Вибирається той крок, який призводить до найбільшого зменшення функції: $\bar{\xi}^{*} = \arg \min_{i=1,m} f(x^{-k} + g \cdot \bar{\xi}_1)$. І в даному напрямку робиться крок

$$\Delta x^{-k} = \lambda \cdot \bar{\xi}^{*}.$$

Параметр λ може визначатися як результат мінімізації у напрямку, що визначається найкращою пробою, або вибиратися за певним законом.

Зі збільшенням числа проб обраний напрям наближається до напрямку $-\nabla f(\bar{x})$.

Якщо функція $f(\bar{x})$ близька до лінійної, то є можливість прискорити пошук, вибираючи разом з найкращою пробою і найгіршу. Тоді робочий крок можна робити або в напрямку найкращого, або в напрямку протилежному найгіршій пробі.

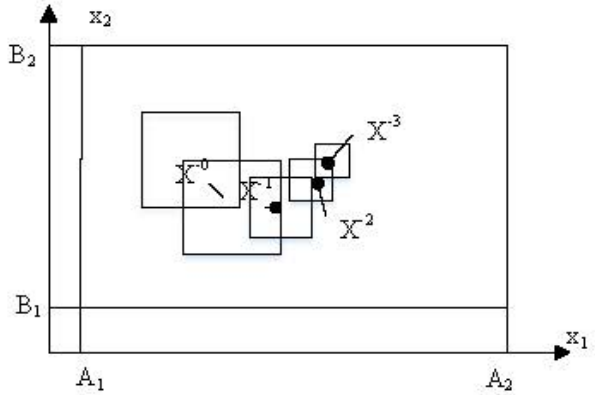


Рисунок 12.8 – Послідовність пошуку за алгоритмом найкращої проби

Векторний випадковий пошук

Метод статистичного градієнта. З вихідного стану x^{-k} робиться m незалежних проб $g \cdot \bar{\xi}_1, ..., g \cdot \bar{\xi}_m$ у m випадкових напрямках, а потім обчислюються відповідні значення функції, що мінімізується в цих точках. Для кожної проби запам'ятовуємо приріст функції

$$\Delta f_j = f(x^{-k} + g \cdot \bar{\xi}_j) - f(x^{-k}).$$

Після цього формуємо векторну суму

$$\Delta \bar{f} = \sum_{j=1}^{m} \bar{\xi}_j \cdot \Delta f_j .$$

У межі при $m \to \infty$ напрямок $\Delta \bar{f}$ збігається з напрямком градієнта цільової функції. При остаточному m вектор $\Delta \bar{f}$ являє собою статистичну оцінку напрямки градієнта. У напрямку $\Delta \bar{f}$ робиться робочий крок і, в результаті, чергове наближення визначається відношенням

$$\bar{x}^{k+1} = \bar{x}^{k} - \lambda \cdot \frac{\Delta \bar{f}}{\left\| \bar{f} \right\|} .$$

При виборі оптимального значення λ, яке мінімізує функцію в заданому напрямку, ми отримуємо статистичний варіант методу найшвидшого спуску. Суттєва перевага перед детермінованими алгоритмами полягає в можливості прийняття рішення про напрямок робочого кроку при $m < n$. При $m=n$ і невипадкових ортогональних робочих кроках, спрямованих уздовж осей координат, алгоритм вироджується в градієнтний метод.

Алгоритм найкращої проби з напрямним гіперквадратом. Усередині допустимої області будується гіперквадрат. У цьому гіперквадраті випадковим чином розкидається m точок $\bar{x}_1, ..., \bar{x}_m$, в яких обчислюються значення функції. Серед збудованих точок вибираємо найкращу. Таким чином, на 1-му етапі координати випадкових точок задовольняють нерівності $a_i^1 \le x_i \le b_i^1$, $i = \overline{1,n}$, і $x^{-1} = \arg \min_{j=1,m} \{ f(\bar{x}_j) \}$ – точка з мінімальним значенням цільової функції.

Спираючись на цю точку, будуємо новий гіперквадрат. Точка, в якій досягається мінімум функції на k-му етапі, береться як центр нового гіперквадрата на *(k+1)*-му етапі.

Координати вершин гіперквадрата на *(k+1)*-му етапі визначаються співвідношеннями

$$a_i^{k+1} = x_i^{k+1} - \frac{b_i^k - a_i^{k+1}}{2},\ b_i^{k+1} = x_i^{k+1} - \frac{b_i^k - a_i^{k+1}}{2},$$

де x^{-k} – найкраща точка в гіперквадраті на k-му етапі.

У новому гіперквадраті виконуємо ту ж послідовність дій, випадковим чином розкидаючи m точок. В результаті здійснюється спрямоване переміщення гіперквадрата в бік зменшення функції.

В алгоритмі з навчанням сторони гіперквадрата можуть регулюватися відповідно до зміни за деяким правилом, що визначає стратегію зміни сторони гіперквадрата. У цьому випадку координати вершин гіперквадрата на *(k+1)*-му етапі будуть визначатися співвідношеннями

$$a_i^{k+1} = x_i^{k+1} - \frac{b_i^k - a_i^k}{2\alpha},\ b_i^{k+1} = x_i^{k+1} - \frac{b_i^k - a_i^k}{2\alpha}.$$

Добре вибране правило регулювання стороки гіперквадрата призводить до досить ефективного алгоритму пошуку.

Генетичний алгоритм

Генетичний алгоритм (англ. Genetic algorithm) – різновид алгоритму направленого випадкового пошуку, що використовується для вирішення завдань оптимізації та моделювання шляхом випадкового підбору, комбінування і варіації шуканих параметрів.

Завдання формалізується таким чином, щоб його рішення могло бути закодованим у вигляді вектора (“генотипу”) генів, де кожен ген може бути бітом, числом або якимсь іншим об'єктом. У класичних реалізаціях генетичного алгоритму (ГА) передбачається, що генотип має фіксовану довжину. Однак існують варіанти ГА, вільні від цього обмеження.

Відповідно до ГА, деяким, звичайно випадковим, чином створюється множина генотипів початкової популяції. Вони оцінюються з використанням функції пристосованості» (fitness function), яка визначає, наскільки добре генотип наближує критерій оптимальності до екстремуму.

З отриманої множини рішень (“покоління”) з урахуванням значення «пристосованості» обираються рішення, до яких застосовуються «генетичні оператори» (“схрещування” - crossover і “мутація” - mutation), результатом чого є отримання нових рішень. Для них також обчислюється значення пристосованості, і потім проводиться відбір (“селекція”) кращих рішень в наступне покоління.

Цей набір дій повторюється ітеративно, поки не буде виконаний критерій зупинки алгоритму. Таким критерієм може бути:

- знаходження глобального, або субоптимального рішення;
- закінчення числа поколінь, відпущених на еволюцію;
- закінчення часу, відпущеного на еволюцію.

12.1.7 Лінійне програмування

Однією з задач опуклої оптимізації, що найчастіше зустрічаються, є так звана задача лінійного програмування, в якій як цільова функція, так і всі обмеження лінійні. Для розв'язання цієї задачі можуть бути застосовані загальні способи опуклої оптимізації, які описані підрозділі 12.1.4. Але на практиці для цього випадку застосовують інші, більш прості способи, з одним з яких (так званий симплекс-метод) ми зараз ознайомимося.

Постановки задач лінійного програмування, які зустрічаються на практиці, передбачають ще одне додаткове обмеження. Це умова невід'ємності значень всіх змінних. Далі, обмеження типу нерівностей $p_i(x) \geq 0$ або $p_i(x) \leq 0$ введенням додаткових змінних з невід'ємними значеннями перетворюються в рівності $p_i(x) - z_i = 0$ або $p_i(x) + z_i = 0$. Крім того, простою зміною знака цільової функції задача знаходження максимуму зводиться до задачі знаходження мінімуму. Таким чином, в лінійному програмуванні можна обмежитись лише розв'язанням задачі мінімізації.

Стандартна постановка задачі лінійного програмування полягає в тому, щоб: знайти мінімум лінійної функції $d + c_1 x_1 + c_2 x_2 + ... + c_n x_n$ при обмеженнях $b_i + a_{i1} x_1 + a_{i2} x_2 + ... + a_{in} x_n = 0 \ \ (i - 1,..., m), \ \ x_i \geq 0 \ \ (i = 1,..., m)$.

Інакше, йдеться про знаходження невід'ємного розв'язку $x = (x_1,..., x_n)$ даної системи лінійних алгебраїчних рівнянь $b_i + \sum_{j=1}^{n} a_{ij} x_j = 0 \ (i = 1,..., m)$, яке перетворює в мінімум значення даної лінійної функції $d + \sum_{j+1}^{n} c_j x_j$.

Задача лінійного програмування може не мати розв'язку в таких випадках:

1) система S рівнянь $b_i + \sum_{j=1}^{n} a_{ij} x_j = 0 \ (i = 1,..., m)$ несумісна, тобто взагалі не має розв'язків;

2) система S не має жодного від'ємного розв'язку;

3) на множині M невід'ємних розв'язків системи S цільова функція $f = d + \sum_{j+1}^{n} c_j x_j$ може приймати як завгодно великі за абсолютною величиною від'ємні значення, тобто $\min_{M} f = -\infty$.

Симплекс-метод

Один з найбільш розповсюджених методів розв'язування задачі лінійного програмування в стандартній постановці полягає в послідовному застосуванні до лінійних функцій так званих симплексних перетворень, що являють собою

розв'язки одного з обмежувальних рівнянь $b_i + \sum_{j=1}^{n} a_{ij} x_j = 0$ відносно деякого невідомого x_j, і підстановку знайденого таким чином значення x_j до всіх обмежувальних рівнянь і в цільову функцію $f = d + \sum_{j+1}^{n} c_j x_j$.

Транспортна задача

В деяких окремих випадках розв'язання задачі лінійного програмування може бути значно спрощено. Найважливішим з таких випадків є так звана транспортна задача, яку називають іноді задачею про призначення. Суть її полягає в мінімізації цільової функції

$$f = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_j$$

при $m + n$ обмеженнях типу рівнянь

$$\sum_{j=1}^{n} x_{ij} = a_i, \quad \sum_{i=1}^{m} x_{ij} = b_i .$$

Звичайно додатково припускається зберігання рівняння

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j ,$$

інакше задача не буде мати розв'язків.

Назву транспортної задача, що розглядається, одержала тому, що до неї зводиться оптимізація плану перевезень вантажів з m пунктів відправлення з запасами $a_1,...,a_m$ до n пунктів призначення з об'ємами споживання $b_1,...,b_n$. Роль коефіцієнтів c_{ij} в цільовій функції виконують питомі вартості, тобто вартості перевезення однієї одиниці вантажу з пункту i в пункт j. Задача полягає в мінімізації загальної вартості перевезення вантажів за умови, що вантажі повністю вивезені з усіх пунктів відправлення і потреби всіх пунктів призначення виявились повністю задоволеними.

Цілочислове лінійне програмування

Іноді на практиці доводиться зустрічатися з такими задачами лінійного програмування, в яких припустимі лише цілочислові розв'язки.

В загальному випадку одержання цілочислового розв'язку потребує застосування спеціальних методів, серед яких найчастіше вживають методи відтинань та розгалужень і меж.

Методи відтинань базуються на таких ідеях. Спочатку розв'язується звичайна задача лінійного програмування A_0 (з тимчасово відкинутою вимогою цілочислового розв'язку). Якщо координати точки $a^0 = (a_1,...,a_n)$, яка одержана в результаті розв'язання задачі A_0, цілі числа, то ця точка дає розв'язок не тільки задачі A_0, а й початковій задачі цілочислового лінійного програмування, яку ми домовимося позначати через A.

Якщо хоча б одна з координат точки a^0, наприклад, a_j, не є цілим числом, то до початкової задачі цілочислового лінійного програмування додається нове обмеження, що побудоване з використанням інформації, яка є в таблиці симплекс-методу і відповідає кроку, на якому побудований розв'язок a^0.

В результаті виходить нова задача A_1, до якої знов застосовується та ж сама процедура. Таким чином, методи відтинань будують і розв'язують послідовність лінійних задач $A_0, A_1...,$ кожна з яких відрізняється від попередньої лише одним новим обмеженням.

Загальна задача лінійної оптимізації

В загальному випадку задача дискретної оптимізації не припускає лінійності як цільової функції $f(x)$, так і обмежень $p_i(x) \geq 0$ $(i = 1,...,m)$. Не вимагається також неодмінно цілочисельність координат точок, серед яких знаходиться розв'язок. Важливо лише, щоб вони утворювали дискретну множину M. Звичайно на практиці загальне число N точок виявляється настільки великим, що простий перебір фізично неможливий навіть при використанні найпотужніших ЕОМ. З цієї причини основна задача в дискретній оптимізації зводиться до розробки методів, які направлені на максимально можливе звуження перебору.

У так званих прямих методах дискретної оптимізації звичайно використовуються ті чи інші аналоги (в неперервному випадку) градієнтних методів, що розглядалися вище. До них відносять методи локальної оптимізації, наприклад, метод вектора спаду. Обчислювальну схему цього методу для розв'язання задачі мінімізації дійсної функції $f(x)$, що визначена на деякому просторі M дискретного метричного простору D, можна описати таким чином.

Спочатку вибираємо деяке початкове наближення $x^0 \in M$ і радіус r. В просторі D розглядаємо окіл $O(x^0, r)$ радіуса r з центром в точці x^0. Досліджуючи координати вектора спаду, який характеризує зміни значень цільової функції в точках множини $O(x^0, r) \cap M$ порівняно з $f(x^0)$, визначаємо, чи є x^0 точкою локального мінімуму функції f. Якщо ні, то за допомогою вектора спаду в множині $O(x^0, r) \cap M$ вибираємо точку x^1, для якої значення цільової функції менше, ніж $f(x^0)$. На наступній ітерації знову повторюємо цю ж процедуру,

виходячи вже з x^1 як з центра нового околу, і так далі. Процес обчислень вважається закінченим, якщо одержаний деякий локально оптимальний розв'язок задачі. Алгоритм методу припускає переривання обчислень на будь-якій ітерації з видачею в якості результату останнього одержаного припустимого розв'язку задачі.

В релаксаційних методах звертаються до способу ослаблення (релаксації) обмежень і заміни цільової функції $f(x)$ її мінорантою $f'(x)$ (в задачі мінімізації), тобто функцією, яка задовольняє умову $f'(x) \le f(x)$ для всіх x.

З припустимої області M', що здобута розширенням вихідної області $M(M' > M)$ за рахунок ослаблення обмежень, M' і $f'(x)$ вибираються так, щоб, по-перше, релаксована задача допускала порівняно прості способи розв'язань, і, по-друге, щоб вона найменше відрізнялась від вихідної за цільовою функцією і множиною припустимих розв'язків:

$$\min_{x \in M'} f'(x) \le \min_{x \in M'}(fx) \le \min_{x \in M} f(x)$$

Метод розгалужень і меж (МРМ)

Цей метод дозволяє отримати точний або наближений розв'язок з заданою відносною похибкою за цільовою функцією. Суть МРМ полягає в тому, що замість вихідної задачі оптимізації $A_0 = \{\min f_0(x), x \in M_0\}$ будується і розв'язується послідовність релаксованих задач $\overline{A}_i = \{\min \bar{f}_1(x), x \in \overline{M}_i\}$, $i = 0,1,2...$ Спочатку розв'язується задача для M_0 і $\bar{f}_0(x_0) = f_0(x_0)$, то x_0 – оптимальний розв'язок A_0, і процес закінчується. Інакше $\bar{f}_0(x_0)$ – оцінка знизу для цільової функції на оптимальному розв'язку задачі A_0, і алгоритм починає процес розгалуження вихідної задачі A_0 на декілька задач (безпосередніх нащадків).

12.1.8 Теорія ігор

До методів дискретної оптимізації і лінійного програмування часто доводиться звертатись при аналізі і виборі рішень в конфліктних ситуаціях, тобто при наявності сторін, що мають на меті відмінні (найчастіше – протилежні) цілі.

Стратегією гравця називають сукупність правил, що визначають поведінку гравця від початку гри до її завершення.

Задання стратегій *(A,B)* двох гравців в парній грі повністю визначає її наслідок, тобто виграш одного і програш другого. Гра називається скінченною, якщо в кожного гравця є лише скінченна кількість стратегій. Результати скінченної парної гри з нульовою сумою можна задавати матрицею, рядки і стовпці якої відповідають відмінним стратегіям, а її елементи - відповідні виграші однієї сторони (які дорівнюють програшам іншої).

Розглянемо гру $m \times n$ з матрицею

	B_1	B_2	...	B_n
A_1	a_{11}	a_{12}	...	a_{1n}
A_2	a_{21}	a_{22}	...	a_{2n}
...	...	...	...	...
a_m	a_{m1}	a_{m2}	...	a_{mn}

Якщо перший гравець застосовує стратегію A_i, то другий буде прагнути до того, щоб вибором відповідної стратегії B_j звести виграш першого гравця до мінімуму. Величина цього мінімуму, яку ми позначимо α_i, дорівнює, очевидно, $\min\limits_j a_{ij}$.

З точки зору першого гравця (при будь-яких відповідях супротивника) доцільно прагнути знайти таку стратегію, при якій α_i перетворюється в максимум. Цей максимум, який ми позначимо α, називається нижньою ціною гри. Оскільки значення α обчислюється за формулою $\alpha = \max\limits_i \min\limits_j a_{ij}$, то його називають також максиміном. Йому відповідає максимінна стратегія, дотримуючись якої, перший гравець при будь-яких стратегіях у противника забезпечить собі виграш, який не менше α (залежно від знака α це може бути і програш, який в цьому разі виявиться мінімальним).

Аналогічно визначається мінімальний програш (який може бути в дійсності і виграшем) для другого гравця: $\beta = \min\limits_j \max\limits_i a_{ij}$.

Величина β називається верхньою ціною гри або мінімаксом. Їй відповідають мінімаксні стратегії другого гравця.

Має місце нерівність $\alpha \le \beta$. При $\alpha = \beta$ розв'язок одержується в чистих стратегіях. Для знаходження їх достатньо виділити в платіжній матриці максимінні (тобто такі, що дорівнюють α) елементи. Будь-яка пара рядків і стовпців, на перетині яких знаходиться такий елемент, відповідає парі чистих оптимальних стратегій. При $\alpha < \beta$ перший гравець може істотно збільшити свій середній виграш порівняно з α, якщо він буде користуватись не чистою (однією єдиною) стратегією, а так званою змішаною стратегією. Змішана стратегія C полягає в тому, що при повторі гри здійснюється випадковий вибір стратегії з деякої множини змішаних стратегій, і для кожної змішаної стратегії вказується імовірність її вибору.

12.1.9 Динамічне програмування

На практиці часто доводиться зустрічатись з випадками, коли метою (ціллю) оптимізації є встановлення найкращої послідовності тих чи інших робіт

(виробничих операцій, етапів будівництва різних споруд тощо). З подібним зустрічаються при розв'язанні задач так званого динамічного програмування.

Однією з перших задач такого роду, що привернули увагу математиків, була так звана задача про комівояжера (мандрівного торговця). Суть її така: є $n+1$ місто $A_0, A_1, ...A_n$ $(n \geq 1)$ з заданими між ними відстанями $d_{ij}(i, j = 0,1,...n)$. Потрібно, відправившись з A_0, вибрати такий маршрут пересування $A_0, A_{i1}, A_{i2}, ..., A_{in}, A_0$, при якому комівояжер, побувавши в кожному місті по одному разу, повернувся б до вихідного пункту A_0, пройшовши при цьому мінімально можливий сумарний шлях.

Основний спосіб динамічного програмування полягає в знаходженні правил домінування, які дозволяють робити порівняння варіантів розвитку послідовностей і завчасне відсіювання безперспективних варіантів. У ряді випадків в задачах динамічного програмування вдається одержати такі сильні правила домінування, що вони визначають елементи оптимальної послідовності однозначно один за одним. В такому випадку правила домінування називають розв'язувальними правилами.

Розв'язувальні правила звичайно виводяться за допомогою принципу оптимальності Беллмана. Суть принципу оптимальності така. Нехай критерій F (задається формулою або алгоритмом), який дає числову оцінку якості варіанта (послідовності) $A_n = A_{i1}A_{i2}...A_{in}$, можна застосовувати не тільки до всієї послідовності, але і до будь-якого її початкового відрізку $A_R = A_{i1}A_{i2}...A_{iR}$. Послідовність A_n, якій відповідає екстремальне значення критерію F, називається оптимальною. Якщо будь-який початковий відрізок оптимальної послідовності також оптимальний (в класі всіх послідовностей, складених з тих же елементів, і можливо, такий, що має ті ж початок і кінець, що і даний відрізок), то вважають, що для відповідної задачі справедливий принцип оптимальності.

Розглянемо зразок розв'язання задачі про комівояжера методом динамічного програмування:

1. Введення даних про пункти $A_0, ..., A_n$ і відстані між пунктами i та j d_{ij} ($d_{ij} = 0$ при $i=j$).

2. Обчислення всіх можливих варіантів відстаней, що складаються з трьох ділянок $A_0, A_{i1}, A_{i2}, A_{i3}$. Вони групуються за останнім пунктом i з них залишаються ті варіанти, що об'єднують однакові пункти, але мають найменший шлях.

3. До тих варіантів, що залишились додають ще четверту ділянку і повторюють процедуру з пункту 2. Це повторюється для п'ятої, шостої і т. д. дільниць, доки не повертається в пункт A_0. Той варіант (чи варіанти), що залишилися, і визначає найкоротший шлях, яким комівояжеру можна об'їздити всі міста A_i $(i=0,...,n)$, якщо він почне та закінчить свою подорож в A_0.

12.1.10 Варіаційні задачі

В варіаційних задачах місце змінних займають функції, а критерій оптимізації – деяка дійсна функція, що залежить від цих функцій і називається функціоналом.

В найпростішому випадку мають справу з функцією $y(x)$ однієї змінної і функціоналом вигляду

$$I = \int_{a_1}^{a_2} F(x, y, y')dx,$$

де a_1 і a_2 – константи, $F(x,y,z)$ – задана функція трьох дійсних змінних. При цьому припускається, що як сама функція $y=y(x)$, так і її похідна $y' = \frac{d}{dx}y(x)$ неперервні на відрізку $a_1 \le x \le a_2$.

В класичному варіаційному численні доводиться, що екстремум функції (мінімум або максимум) функціонала I досягається на функціях $y(x)$, які задовольняють диференціальне рівняння, тобто рівняння Ейлера:

$$F_y + F_{xy'} - F_{yy'}y' - F_{y'y'}y'' = 0.$$

Тут через F_y і F_{uv} ($u = x, y$ або $y', v = y'$) позначені, відповідно, перша і друга частинні похідні функції $F(x, y, y,')$ за змінною y і за змінними u, v.

12.1.11 Алгоритми на графах

Як вже зазначалося, графи широко використовуються для розв'язання різноманітних задач з використанням комп'ютерів. Це підтверджується тим, що задачам на графах присвячена половина фундаментальної праці Р.Седжвіка "Фундаментальні алгоритми на С".

Основні задачі, які вирішуються на графах:

1) пошук шляху в графі – будь-якого або найкоротшого (для незважених графів найкоротший шлях – шлях, який складається з мінімальної кількості ребер; для зважених графів – це шлях з мінімальною сумарною довжиною ребер);

2) пошук циклів в графі: будь-яких, мінімальної та максимальної довжини;

3) пошук гамільтонового циклу (циклу, який проходить через всі вершини графу) та ейлерового циклу (циклу, який проходить через всі ребра графу);

4) побудова дерев графу, тобто виділення таких вершин і ребер, які утворюють підграф типу дерево, в тому числі кістякове дерево;

5) пошук критичного шляху у графі (шляху аксимальної довжини);

6) задача про максимальний потік: знайти такий розподіл потоків у мережі, який забезпечить максимальний сумарний потік з витоку до стоку.

Розглянемо найпоширеніші алгоритми розв'язання цих задач.

Пошук шляху в незваженому графі

Основні способи пошуку шляху у незважених графах:

1) способи перебору варіантів;

2) пошук в глибину;

3) пошук в ширину.

Переборні методи

Переборні методи основані на генеруванні перестановок вершин і перевірці, чи є згенерована перестановка необхідним шляхом.

Пошук шляху в графі в глибину

Головний інструмент алгоритму пошуку в глибину – стек.

Стек – структура даних, яка діє за правилом: останнє занесене дане першим вилучається (LIFO – last input – first output).

Алгоритм пошуку шляху в глибину ґрунтується на таких кроках:

1) починаючи від початкової вершини, будується шлях у графі, використовуючи ребра з мінімальним номером кінцевої вершини. Послідовність вершин заноситься у стек;

2) якщо на цьому шляху не зустрінеться кінцева вершина, тоді зі стека вилучається остання занесена вершина і береться ребро з більшим номером кінцевої вершини;

3) якщо невикористаних ребер немає, знову зі стека вилучається вершина (робиться крок назад);

4) якщо стек став пустим, а шлях не знайдено, то його немає взагалі.

Переваги та недоліки методу пошуку в глибину: цей метод дозволяє знайти шлях у незваженому графі, знайдений шлях не є найкоротшим, але метод, у середньому, забезпечує найбільшу швидкість.

Пошук шляху у графі в ширину

Цей метод дозволяє знайти найкоротший шлях у незваженому графі.

Основний засіб для здійснення пошуку в ширину – структура даних черга.

Черга – структура даних, яка діє за правилом: перший занесений першим вилучається (FIFO: first input – first output).

Алгоритм пошуку шляху в ширину ґрунтується на таких кроках:

1) вершини при пошуку в ширину заносяться в *чергу*. Елемент черги складається з двох частин: номера вершини та номера елемента черги, звідки до цієї вершини потрапили;

2) до черги заноситься початкова вершина, а у другу частину елемента – 0 (ознака початку шляху);

3) розглядаються *всі* вершини, суміжні з початковою, і заносяться до черги;

4) якщо серед розглянутих вершин немає кінцевої – пересунути вказівник початку черги і розглянути вершини, суміжні з тією, на яку вказує вказівник;

5) дійшовши таким чином до кінцевої вершини, пройти шлях у зворотному порядку, використовуючи другу частину елементів черги.

Пошук шляху в зваженому графі

Якщо в графі враховуються ваги ребер, то, очевидно, це ставить задачу пошуку шляху з мінімальною або максимальною сумарною вагою.

Пошук шляху в зваженому графі методом перебору

Метод перебору є найпростішим і найнеефективнішим. Він дозволяє знайти розв'язок у будь-якому випадку (якщо розв'язок взагалі існує), але кількість варіантів, які необхідно перебрати, найчастіше надто велика.

Пошук шляху в зваженому графі методом гілок та границь

Метод дозволяє скоротити кількість варіантів у порівнянні з методом перебору. Застосовується в задачах, де критерій пошуку є монотонно зростаючою функцією кількості ребер.

Ідея методу: знаходиться перший шлях, що задовольняє умову задачі. При пошукові іншого шляху з додаванням до нього чергового ребра перевіряємо, чи не став вже цей частковий шлях довшим за початковий (порівнюємо з границею). Якщо став, то далі нарощувати його немає сенсу – він вже заздалегідь гірший, і всі наступні варіанти (гілки), що мають своїм початком цей частковий шлях, не розглядаються.

Алгоритм Флойда-Уоршелла – динамічний алгоритм для знаходження найкоротших відстаней між усіма вершинами зваженого орієнтованого графу.

Нехай вершини графа $G = (V, E)$, $|V| = n$ пронумеровані від 1 до n і введено позначення d_{ij}^k для довжини найкоротшого шляху від i до j, який окрім самих вершин i, j проходить тільки через вершини $1 \ldots k$. Очевидно, що d_{ij}^0 – довжина (вага) ребра (i, j), якщо таке існує (в іншому випадку його довжина може бути позначена як ∞).

Існує два варіанти значення d_{ij}^k, $k \in (1,\ldots,n)$:

1. Найкоротший шлях між i, j не проходить через вершину k, тоді $d_{ij}^k = d_{ij}^{k-1}$.

2. Існує більш короткий шлях між i, j, що проходить через k, тоді він спочатку йде від i до k, а потім від k до j. У цьому випадку, очевидно, $d_{ij}^k = d_{ik}^{k-1} + d_{kj}^{k-1}$.

Таким чином, для знаходження значення функції досить вибрати мінімум з двох позначених значень.

Тоді рекурентна формула для d_{ij}^k має вигляд:

$$d_{ij}^0 \text{ – довжина ребра } (i, j)$$

$$d_{ij}^k = \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k+1}).$$

Алгоритм Флойда-Уоршелла послідовно обчислює всі значення $\forall i, j$ для від 1 до n. Отримані значення d_{ij}^{k} є довжинами найкоротших шляхів між вершинами i, j.

Алгоритм Дейкстри дозволяє знайти найкоротші шляхи від заданої вершини до кожної з решти вершин графу.

Ідея алгоритму: якщо безпосередній шлях від вершини V_i до V_j більший за шлях $V_i \to V_k \to V_j$, то шлях $V_i \to V_j$ заміняють на шлях $V_i \to V_k \to V_j$.

Алгоритм використовує структуру даних масив [3, N], де N – кількість вершин. Кожен стовпець масиву складається з трьох комірок: a – розглянута вершина, b – відстань від початкової вершини до тої, що розглядається, c – попередня вершина, через яку проходить шлях до вершини, що розглядається. На початку роботи алгоритму $\forall c_i = V_0$, де V_0 – початкова вершина; $\forall b_i$ (відстані від початкової вершини до вершини V_i), які безпосередньо з'єднані з початковою вершиною одним ребром, дорівнюють вазі ребра, решта – великому значенню M, яке для даної задачі імітує нескінченність.

Розглянемо приклад виконання алгоритму Дейкстри для графу, матриця ваг якого має вигляд:

(0, 23, 12, M, M, M, M, M),
(23, 0, 25, M, 22, M, M, 35),
(12, 25, 0, 18, M, M, M, M),
(M, M, 18, 0, M, 20, M, M),
(M, 22, M, M, 0, 23, 14, M),
(M, M, M, 20, 23, 0, 24, M),
(M, M, M, M, 14, 24, 0, 16),
(M, 35, M, M, M, M, 16, 0).

Послідовність кроків алгоритму (для випадку початкова вершина – 3, кінцева вершина – 8):

Крок	***i***	**1**	**2**	**3=V_0**	**4**	**5**	**6**	**7**	**8=V_n**
1 (поч. стан)	***a***	0	0	1	0	0	0	0	0
	b	12	25	0	18	999	999	999	999
	c	3	3	0	3	3	3	3	3
2	***a***	1	0	1	0	0	0	0	0
	b	12	25	0	18	999	999	999	999
	c	3	3	0	3	3	3	3	3
3	***a***	1	0	1	1	0	0	0	0
	b	12	25	0	18	999	38	999	999
	c	3	3	0	3	3	4	3	3
4	***a***	1	1	1	1	0	0	0	0
	b	12	25	0	18	47	38	999	60
	c	3	3	0	3	2	4	3	2

5	***a***	1	1	1	1	0	1	0	0
	b	12	25	0	18	47	38	62	60
	c	3	3	0	3	2	4	6	2
6	***a***	1	1	1	1	1	1	0	0
	b	12	25	0	18	47	38	61	60
	c	3	3	0	3	2	4	5	2
7	***a***	1	1	1	1	1	1	0	1
	b	12	25	0	18	47	38	61	60
	c	3	3	0	3	2	4	5	2
8	***a***	1	**1**	**1**	1	1	1	1	**1**
	b	12	25	0	18	47	38	61	60
	c	3	**3**	0	3	2	4	5	**2**

Мінімальний шлях 3-->2-->8, L_{min}= 60.0

Побудова мінімального кістякового дерева

Прикладом використання задачі побудови мінімального кістякового дерева є прокладання телефонної мережі між заданими пунктами.

Найчастіше для побудови мінімального кістякового дерева використовується алгоритм Пріма-Краскала. Ними доведено, що "жадібний алгоритм" дозволяє знайти мінімальне кістякове дерево. Це алгоритм, який на кожному етапі розв'язання задачі обирає найкращий варіант з точки зору критерію задачі.

Алгоритм Пріма-Краскала

Алгоритм належить до категорії «жадібних алгоритмів». Ідея алгоритму: спочатку вибирають найкоротше ребро, а його кінцеві вершини вносять до дерева – це початок дерева. Потім приєднують до нього найкоротше ребро з тих, що виходять з вершин, які належать до дерева, а входять у вершину, яка не належить до дерева. І так далі, поки всі вершини не увійдуть до складу дерева.

Задача може мати декілька розв'язків, якщо є ребра однакової довжини.

Задача про максимальний потік

Найчастіше для розв'язання задачі використовується алгоритм Форда-Фолкерсона:

1) якщо мережа має декілька витоків та декілька стоків, то вона перетворюється в мережу з одним витоком і одним стоком шляхом введення уявних витоку і стоку;

2) початок розв'язання задачі полягає у тривіальному розв'язку (коли всі потоки рівні 0);

3) виконується збільшення потоку за допомогою "збільшувальних ланцюгів". "Збільшувальний ланцюг" – це така послідовність вершин і ребер, яка по-

чинається з витоку і закінчується стоком, і в усіх ребрах на цьому шляху потік менший за пропускну здатність, тобто може бути збільшений.

Доведено, що алгоритм буде сходитись, тобто потоки будуть наближатись до максимального значення, якщо процес збільшення починати з найкоротшого "збільшувального ланцюга".

4) виконується збільшення потоку у "збільшувальному ланцюзі" на величину:

а) якщо граф неорієнтований

$\min(C_{ij} - V_{ij})$ для $V_{ij} > 0$,

$\min(C_{ij} + V_{ij})$ для $V_{ij} < 0$.

б) якщо граф орієнтований, потрібно враховувати, що напрямок дуги змінити не можна.

Оскільки для забезпечення збіжності алгоритму потрібно починати з найкоротшого збільшувального ланцюга, то в основу перебору ланцюгів кладуть метод пошуку в ширину, який не закінчується пошуком першого найкоротшого шляху, а продовжується далі.