МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«УЖГОРОДСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ»
МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
КАФЕДРА КІБЕРНЕТИКИ І ПРИКЛАДНОЇ МАТЕМАТИКИ

Л.Ф. Гуляницький, О.Ю.Мулеса

-

МЕТОДИ КОМБІНАТОРНОЇ ОПТИМІЗАЦІЇ

Ужгород –2015

Л.Ф.Гуляницький, О.Ю.Мулеса Методи комбінаторної оптимізації: теоретичні відомості. –Ужгород, 2015.– 25 с.

Рекомендовано до друку кафедрою кібернетики і прикладної математики ДВНЗ "Ужгородський національний університет", протокол № 7 від 31 серпня 2015 р.

Рекомендовано до друку методичною комісією математичного факультету ДВНЗ "Ужгородський національний університет", протокол № 7 від 31 серпня 2015 р.

Рецензенти: **Гече Ф.Е.** доктор технічних наук, професор, завідувач кафедри кібернетики і прикладної математики (ДВНЗ "Ужгородський національний університет")

Антосяк П.П., кандидат фізико-математичних наук, доцент кафедри системного аналізу і теорії оптимізації (ДВНЗ "Ужгородський національний університет")

ЗМІСТ

1. МОДЕЛІ КОМБІНАТОРНОЇ ОПТИМІЗАЦІЇ

1.1. Загальна постановка і класифікація задач оптимізації

Довільну задачу оптимізації (не лише комбінаторної) у загальному випадку можна подати кортежем

$$< f, X, \Pi, D, ext >,$$

де $f: X \to R^1$ – задана цільова функція задачі, R^1 – числова пряма, X – простір розв'язків задачі (простір пошуку), П – предикат, який визначає підмножину $D \subseteq X$ припустимих варіантів розв'язку згідно наявних обмежуючих умов, $extr \in \{\min, \max\}$ – напрям оптимізації.

У цих позначеннях задачу оптимізації можна переписати у вигляді:

необхідно знайти $x_* \in D \subseteq X$ таке, що

$$x_* = \arg \underset{x \in D \subseteq X}{ext} f(x). \tag{1.1}$$

Під простором розуміється множина X, в якій вводяться певні співвідношення між елементами (відстані, околи, сусідство тощо), а предикат $\Pi(x)$ визначає множину припустимих розв'язків так:

$$\Pi(x) = \begin{cases} 1, & x \in D, \\ 0, & x \notin D. \end{cases}$$

Значно рідше зустрічаються задачі, в яких необхідно знайти всю множину розв'язків, які відповідають екстремальному значенню цільової функції, – ця множина позначається так: $Arg \underset{x \in D \subseteq X}{ext} f(x)$.

Варто підкреслити, що дуже рідко в оптимізаційних задачах вимагається знайти власне екстремальне значення цільової функції, тоді задача набуває вигляду:

необхідно знайти

$$\underset{x \in D \subseteq X}{ext}\ f(x). \qquad (1.2)$$

Хоча задачі (1.1) і (1.2) – це дві різні задачі, історично склалося так, що в ряді застосувань (наприклад, в соціально-економічних дослідженнях) задачу оптимізації записують у формі (1.2), але насправді розуміють як проблему пошуку саме аргументу екстремуму (1.1), а не лише відповідного значення цільової функції.

Всюди в подальшому, якщо особливо не обумовлене протилежне, будемо для визначеності розглядати задачі на мінімум.

Цільова функція визначається аналітичним (чи іншим) способом задання та конкретними набором числових даних: $f(x)=f(x|c)$, де c – набір даних задачі (вхід задачі).

Означення 1.1. ***Індивідуальною задачею оптимізації*** називається пара ($f(x|c)$, X).

Тоді під ***задачею оптимізації*** розуміють сукупність усіх можливих індивідуальних задач.

В залежності від виду простору X розрізняють задачі неперервної, комбінаторної та змішаної оптимізації. Перший клас утворюють задачі, в яких простір X є неперервним (континуальним). Щодо другого класу, то тут існують різні підходи до означення як самого поняття ЗКО, так і його підкласів – власне комбінаторних задач, а також задач дискретного та цілочислового програмування. Більш детальна класифікація подана на рис. 1.1, де ЗНЛП – це задачі неперервного лінійного програмування, ЗБП – задачі (пвсевдо)булевого програмування, ЗЦЛП – задачі цілочислового лінійного програмування.

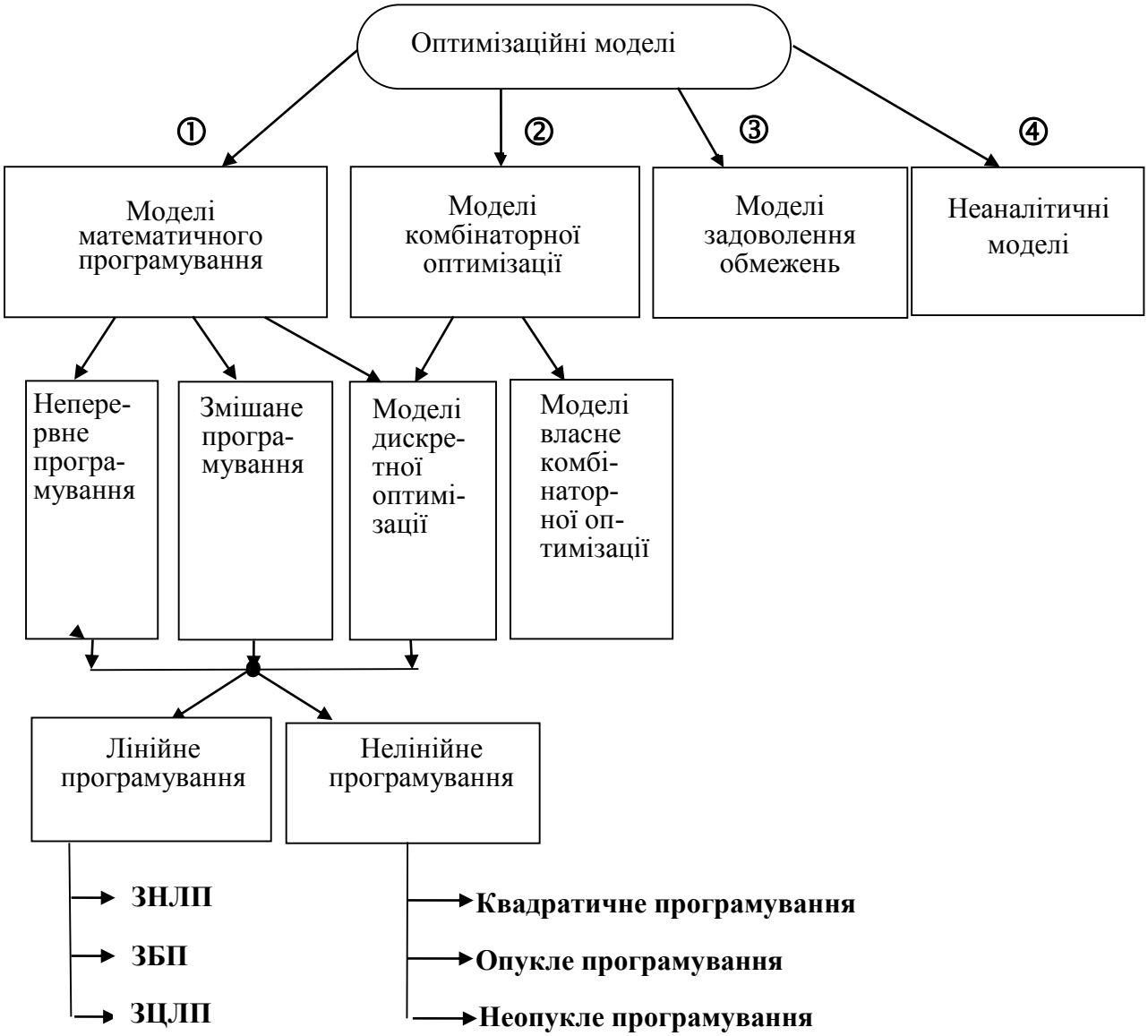

Рис. 1.1. Класифікація оптимізаційних моделей

Часто під *дискретною оптимізацією* розуміють задачу вигляду (1.1), в якій $X = D_1 \times ... \times D_n$ – це *n*-вимірний простір дійсних чисел, – $x = (x_1,...,x_n)$, $x_i \in D_i$, $i = 1,...,n$, причому хоча б одне із D_i є дискретним підпростором.

1.2. Формалізація задач комбінаторної оптимізації

Перш ніж перейти до формалізації ЗКО, введемо ряд понять.

У якості досліджуваного простору вважатимемо вибрану множину (назвемо її твірною), між елементами якої існують певні співвідношення. Нагадаємо, що булеаном 2^X довільної множини X називається множина всіх її підмножин.

Означення 1.2. ***Дискретний простір*** – це такий простір, у якому твірна множина X є скінченною ($\|X\| < \infty$) або нескінченою без граничних точок.

Граничні точки ще називають *точками конденсації, дотику* або *накопичення*.

Нехай в просторі X тим чи іншим чином введена система околів O, тобто для довільного $x \in X$ визначене сімейство множин $o^\sigma(x) \subseteq 2^X$, $\sigma \in I$, де I – множина індексів околів, причому $x \in o^\sigma(x)$ для всіх значень σ, а $O = \{o^\sigma(x) : x \in X, \sigma \in I\}$.

Найчастіше використовуються такі види околів:

– метричні;

– топологічні;

– алгоритмічні;

– дескриптивні.

Перші два види добре відомі у математиці. Утворення алгоритмічних околів описується певною процедурою, наприклад, 2-заміни в ЗК. Дескриптивні визначаються шляхом задання в аналітичній чи в описовій формі (наприклад, шляхом переліку для кожного $x \in X$ елементів, які належать його околам).

Нехай X – метричний простір, тобто на ньому введена метрика $d(x,y)$.

Означення 1.3. Під метричним околом заданого радіусу $\rho>0$ розуміють множину

$$L_\rho(x) = \{y \in X : d(x,y) \le \rho\}.$$

Для більшості дискретних просторів $\rho \in \{1,2,3,\ldots,diam\, X\}$, де діаметр простору X – це відстань між найбільш віддаленими точками:

$$diam\, X = \max_{\forall x,y \in X} \{d(x,y)\}.$$

Приклад. Нехай $X \subseteq \{0,1\}^n$ – n-вимірний метричний простір векторів, компонентами яких є булеві змінні, $x = (x_1,\ldots,x_n) \in X$, $x_i \in \{0,1\}$, $i = 1,\ldots,n$, $d(x,y) = \sum_{i=1}^{n} |x_i - y_i|$, $x, y \in X$ – функція відстані на X. Тоді метричний окіл одиничного радіусу $L_1(x)$ визначається так: $L_1(x) = \{y \in X : d(x,y) = 1\}$, тобто, це множина булевих векторів, які відрізняються не більше ніж на одну компоненту.

На основі введених позначень означення 1.2 можна перефразувати так.

Означення 1.4. Простір X називається ***дискретним***, якщо

$$\forall x \in X \ \exists\ o^{\sigma}(x) \in O : \{x\} \equiv o^{\sigma}(x).$$

Таким чином, для кожної точки дискретного простору існує окіл, який складається лише із цієї точки – саме в такому разі точку називають ізольованою. Тому інколи використовують наступне означення.

Означення 1.5. Простір X називається ***дискретним***, якщо він складається лише з ізольованих точок.

Неважко показати еквівалентність всіх трьох наведених означень дискретного простору.

Варіант розв'язку $x_* \in S$ називається ***субоптимальним розв'язком*** задачі (1.1), якщо

$$f(x_*) \le f(y), \quad \forall y \in S. \tag{1.4}$$

Якщо S співпадає з (метричним) околом точки x, тобто $S = o^{\sigma}(x)$, де $o^{\sigma}(x)$ – це деякий окіл точки x, то такий субоптимальний розв'язок

називається ***локально-оптимальним*** або просто ***локальним розв'язком*** задачі (1.1).

Введені означення дозволяють формально визначити поняття глобального екстремуму. Якщо умова (1.4) виконується для $S = X$, то точка x_* називається ***глобальним*** або ***точним*** розв'язком задачі (1.1).

Якщо система околів така, що будь-який локально-оптимальний розв'язок є глобальним, то кажуть, що така система околів є *точною*.

Часто під ЗКО розуміють проблему пошуку екстремумів заданої цільової функції вигляду (1.1), коли X – комбінаторний простір. Під комбінаторним простором розуміється сукупність комбінаторних об’єктів певного типу, утворених із елементів заданої скінченної множини (твірна множина), хоча формального означення не дається.

Нехай X – простір з околами: $X=(X, O)$.

Означення 1.6. ***Базисними околами*** довільної точки $x \in X$ будемо називати множину

$$B_x = \{o^\tau(x) \in O: \|o^\tau(x)\| > 1 \ \& \ \nexists \gamma: 1 < \|o^\gamma(x)\| < \|o^\tau(x)\|\}.$$

Тобто, базисні околи – це такі нетривіальні (неодноточкові) околи точки, що мають найменшу потужність із всіх околів цієї точки.

Неважко бачити, що всі базисні околи однієї точки мають однакову потужність, а для деяких точок простору таких околів може і не існувати.

Означення 1.7. Простір X називається ***локально скінченним*** (у комбінаторному розумінні), якщо всі базисні околи його точок скінченні:

$$\forall \ x \in X, \ o^\tau(x) \in B_x \Rightarrow \| o^\tau(x) \| < \infty.$$

Означення 1.8. ***Комбінаторним простором*** назвемо дискретний локально-скінченний в комбінаторному розумінні простір, який має не більше ніж злічену кількість елементів.

Повертаючись до оптимізаційної задачі (1.1), дамо наступне означення.

Означення 1.9. Задача (1.1) називається ***ЗКО***, якщо простір її розв'язків X – це комбінаторний простір.

1.3. Обчислювальна складність задач комбінаторної оптимізації

При дослідженні складності алгоритмів задач оптимізації останні подаються у *формі розпізнавання*, яку можна визначити так:

для даної індивідуальної ЗКО виду (1.1) і цілого числа M визначити, чи існує такий припустимий розв'язок $x \in D \subseteq X$, *що* $f(x) \le M$.

Зрозуміло, що для задачі максимізації нерівність мала б вигляд $f(x) \ge M$.

Можна показати, що ця форма постановки задач має таку ж трудомісткість, як і відповідний оптимізаційний варіант.

Розрізняють два класи проблем: *P* та *NP*. Ці класи можна визначити більш точно за допомогою будь-якого формального визначення алгоритмів, такого як машина Тюрінга. Не заглиблюючись в більш формальний опис, для подальшого буде нам достатньо визначити їх так.

Клас *P* складають задачі, для розв'язання яких відомі алгоритми з поліноміальною складністю (polynomial-time algorithms). До поліноміальних відносять алгоритми, складність (трудомісткість) яких обмежена поліномом від розміру входу ЗКО. Отже, це клас відносно простих задач, для яких існують ефективні алгоритми.

До **класу *NP*** (nondeterministic polynomial), який здається більш широким, відносять задачі, що можуть бути розв'язаними недетермінованим поліноміальним алгоритмом. Такі алгоритми мають дві фази: на першій знаходиться припустимий варіант розв'язку, а на

другій цей варіант перевіряється детермінованим поліноміальним алгоритмом верифікації.

Означення 1.10. ЗКО π у варіанті розпізнавання є ***NP-повною*** **(***NP*-complete**)**, якщо виконуються дві умови:

1) $\pi \in NP$;

2) всі задачі із *NP* можуть бути зведеними до π за допомогою поліноміального алгоритму.

Зауважимо, що на практиці для верифікації п.2 означення 1.10 зазвичай лише показують, що деяку відому *NP*-повну задачу можна поліноміально перетворити в досліджувану задачу.

Інколи можна показати, що всі задачі із *NP* поліноміально зводяться до деякої задачі π, але не вдається довести, що $\pi \in NP$. Такі задачі відносять до числа *NP*-складних.

Означення 1.11. Задача відноситься до класу ***NP-складних*** (*NP*-hard), якщо до неї поліноміально зводяться всі задачі із класу *NP*.

Для потреб практики важливо, щоб трудомісткість оптимального алгоритму була обмежена поліномом від вхідних даних задачі. На сьогодні предметом дискусій є питання, чи *P=NP*? У разі негативної відповіді теоретично не існує поліноміального алгоритму, тому при розв`язанні таких задач відразу слід зосередитися на розробці потужних наближених алгоритмів, які хоча б і не гарантують отримання оптимального розв`язку, здатні знаходити оптимальний чи близький до нього розв`язок за прийнятний час.

Насамкінець зазначимо, що викладене справедливе не лише для ЗКО, які розглядалися в формі розпізнавання, а і для ряду інших проблем, що формулюються відразу як задачі розпізнавання (наприклад, задача виконуватості булевої форми чи задача про гамільтонів цикл, в якій слід вияснити, чи існує в заданому графі цикл, що проходить через кожну вершину рівно один раз).

1.4. Приклади моделей комбінаторної оптимізації

1) Задача комівояжера (ЗК)

Дано множину з N міст, відстані між якими відомі. Необхідно знайти найвигідніший замкнутий маршрут, що проходить через задані міста рівно по одному разу.

Математична постановка задачі є такою: Нехай дано граф $G = (V, E)$, де V – множина вершин, E – множина ребер, d_{ij} – вага ребра. Необхідно знайти Гамільтонів цикл мінімальної ваги, тобто замкнутий цикл у графі, що містить усі вершини та передбачає відвідування кожної вершини лише один раз.

2) Задача (псевдо)булевого лінійного програмування

Загальна постановка задачі: знайти такий булевий вектор, для якого досягається

$$\max \sum_{i=1}^{n} c_i x_i$$

за обмежень

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i,$$

де $x=(x_1,\ldots,x_n)$, $x_i \in \{0,1\}$, – булевий вектор довжини n, а $c=(c_1,\ldots c_n)$, $b=(b_1,\ldots b_n)$, $(a_{ij})_{n\times n}$ – задані числові вектори і матриця.

Строго кажучи, це є задача псевдобулевої оптимізації, оскільки цільова функція не є булева, але історично вона отримала назву задачі булевого програмування.

3) Лінійна задача про призначення.

Дано n виконавців та n робіт. Будь-який виконавець може бути призначений до виконанняя будь-якої роботи. Виконання виконавцем роботи пов’язане з витратами, які залежать від того який виконавець виконує роботу. Необхідно виконати всі роботи, призначивши для

кожної роботи лише одного виконавця, тобто, знайти таке призначення робіт для виконавців, так, щоб загальні витати були мінімальними.

Математична постановка задачі може бути виконана таким чином: оптимізувати задану цільову функцію з врахуванням обмежень:

$$\begin{aligned} &\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \to \min \\ &\sum_{j=1}^{n} x_{ij} = 1 \ \left(\forall i = \overline{1,n}\right), \\ &\sum_{i=1}^{n} x_{ij} = 1 \ \left(\forall j = \overline{1,n}\right), \end{aligned} \qquad (1.5)$$

де c_{ij} – вартість призначення i-го виконавця на j-ту роботу $\left(i,\ j = \overline{1,n}\right)$,

$x_{ij} = \begin{cases} 1, \text{ якщо виконавець } i \text{ призначається на роботу } j, \\ 0, \text{ в іншому випадку.} \end{cases}$

Матрицю, що задовільняє умовам (1.5), інколи називають перестановочною матрицею.

Нехай $p=(p_1,...,p_n)$ – довільна перестановка всіх n елементів множини $\{1, ..., n\}$. Будемо вважати, що значення i-го елемента перестановки p_i містить номер роботи, на виконання якої призначений i-й виконавець. Тобто, якщо $p_i=3$, то це означає, що i-го виконавця призначено на третю роботу.

Тоді в термінах перестановок цільова функція матиме вигляд:

$$f(p) = \sum_{i=1}^{n} c_{ip_i} \to \min,$$
$$C = (c_{ij})_{n\times n}.$$

Перестановка дозволяє описувати будь-які можливі варіанти в цій задачі, тому при її використанні немає явних обмежень.

4) Квадратична задача про призначення

Нехай дано множину з n об'єктів та множину з n їх місць призначення. Відомі відстані між місцями призначення та інтенсивність

потоків ресурсів між об'єктами. Задача полягає у розміщенні всіх об'єктів у різних місцях призначення такми чином, щоб сума відстаней, помножених на відповідні потоки була мінімальною.

Математична постановка задачі є такою. Задана матриця $C=(c_{ij})_{n\times n}$, $c_{ij}\geq 0$ – інтенсивність потоків між об'єктами (виконавцями, що призначаються), $D=(d_{st})_{n\times n}, d_{st}\geq 0$ – відстань між місцями призначення (роботами). Необхідно розмістити n об'єктів на n місцях так, щоб виконувалися умови:

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{s=1}^{n}\sum_{t=1}^{n}c_{ij}d_{st}x_{is}x_{jt}\to\min,$$

$$\sum_{s=1}^{n}x_{is}=\sum_{t=1}^{n}x_{jt}=1,$$

$$x_{kl}=\begin{cases}1, \textit{якщо } k-\textit{ий об'єкт розміщений в } l-\textit{е місце призначення};\\ 0, \textit{в протилежному випадку.}\end{cases}$$

В термінах перестановок цільова функція має вид:

$$f(p)=\sum_{i=1}^{n}\sum_{j=1}^{n}c_{ij}d_{p_ip_j}\to\min,$$

де $p=(p_1,...,p_n)$ – перестановка з n елементів над множиною $\{1, ..., n\}$, така що p_i містить номер місця призначення на яке розміщений i -й об'єкт.

В різних постановках квадратичної задачі про призначення можливі обмежувальні умови: наприклад, зафіксовані позиції деяких елементів чи заборонені позиції.

5) Мінімальне кістякове дерево (МКД)

Кістяковим деревом для неорієнтованого графу G називається його ациклічний підграф, який містить всі вершини графу G. Задача про МКД полягає в пошуку такого кістякового дерева, що сума ваг ребер, які входять у дерево, була б мінімальною.

Математична постановка задачі є такою:

для заданої множини вершин V, ребер E та їх ваг $D = (d_{ij}),\ (i, j) \in E$, необхідно побудувати кістякове дерево із множиною вершин V та підмножиною ребер $U \subseteq E$, яке б мало мінімальну сумарну вагу.

Цільова функція має вигляд:

$$\sum_{(i,j)\in U} d_{ij} \to \min .$$

Додатковою умовою тут може бути те, що локальний степінь вершини не повинен перевищувати деяке задане число. Нагадаємо, що *локальним степенем* вершини називається кількість ребер, інцидентних даній вершині.

Більшість ЗКО – це задачі на скінчених множинах. Але навіть в цьому випадку – не кажучи вже про пошук розв'язку в нескінчених просторах – виникають проблеми із застосуванням точних методів, наприклад методу повного перебору. Серед причин складності перш за все варто відмітити багатоекстремальність цільових функцій, складність урахування обмежених умов, а іноді – і трудомісткість обчислення цільових функцій.

Зазначені аспекти визначають проблемність застосування точних методів для розв'язання прикладних задач і актуальність розробки наближених алгоритмів.

2. МЕТОДИ КОМБІНАТОРНОЇ ОПТИМІЗАЦІЇ. ЗАГАЛЬНИЙ ОПИС

2.1. Класифікація алгоритмів комбінаторної оптимізації

Класифікація алгоритмів комбінаторної оптимізації (АКО) за отримуваним розв'язком. Заради простоти викладу поки вважатимемо, що D ≡ X, тобто розглянемо ЗКО без обмежень: необхідно знайти елемент $x_* \in X$ такий, що

$$x_* = \arg \operatorname*{ext}_{x \in X} f(x). \tag{2.1}$$

Будемо вважати, що АКО – це певна процедура *A,* яка переводить задану підмножину $Z \subset X$ (початкові наближення) у множину $X_* \subset X$:

$$AZ = X_*,$$

тобто X_* – це множина знайдених варіантів розв'язку задачі (2.1).

Існує багато підходів до класифікації АКО – за точністю, за типом використаних просторів, за структурою обчислювальної схеми тощо.

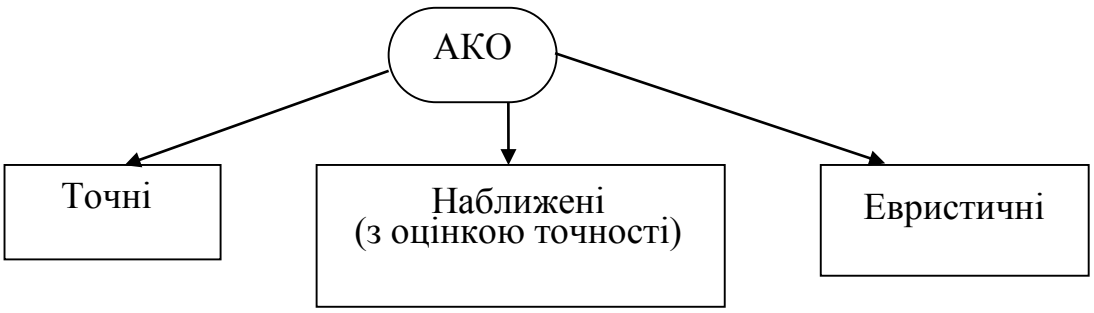

Рис. 2.1. Класифікація АКО за точністю

Точні АКО – це такі методи, які знаходять глобальний розв`язок, тобто для них $X_* \subseteq Arg\,min$.

Наближені АКО діляться на алгоритми:

– з апріорною оцінкою точності;

– з апостеріорною оцінкою точності.

Евристичні алгоритми будуються на основі правдоподібних міркувань (як, наприклад, алгоритм "іди в найближче місто", що використовується для розв'язання ЗК), але нічого не говорять про точність знайденого розв'язку.

Часто дослідники називають наближеними як алгоритми з оцінкою точності, так і евристичні; ми теж у подальшому будемо розглядати такий об'єднаний клас прикладних алгоритмів.

Класифікація АКО за типом обчислювальної схеми. Наближені алгоритми за типом обчислювальної схеми прийнято ділити на послідовні та ітераційні.

Нехай маємо певну множину $Y \supseteq X$.

Послідовні алгоритми (ще називають прямі, конструктивні) – це такі алгоритми, у яких $Y \supset X$ ($Y \neq X$), тобто вони оперують в просторі, що є розширенням простору розв'язків X.

Починаючи "з нуля" чи якогось фрагменту розв'язку, вони поступово формують повний розв'язок.

Приклади послідовних алгоритмів для різних ЗКО:

1) ЗК – алгоритм "іди в найближче місто", який на кожному наступному кроці додає до маршруту вершину, відстань до якої є найменшою із всіх можливих.

2) КЗП – евристика, відповідно до якої об'єкти (елементи) з інтенсивнішими потоками розміщуються в першу чергу.

3) МКД – евристика, відповідно до якої на кожному наступному кроці в побудований фрагмент кістякового дерева включається та вершина, яка мінімально збільшує його сумарну довжину і не утворює підциклу.

Ітераційні алгоритми – це такі алгоритми, які опрацьовують "повні" розв'язки, тобто для них простір пошуку $Y \equiv X$.

Починаючи з деякого $x^o \in X$, ітераційні алгоритми намагаються його покращувати покроково:

$$x^{(h+1)} = A^{(h)} x^{(h)}, \quad h = 0,1,\ldots,$$

де $A^{(h)}$ – ітераційна процедура (у більшості випадків вона не залежить від кроку h: $A^{(h)} = A$).

Ітераційні методи, які оперують на кожному кроці одним (поточним) розв'язком, називаються *траєкторними*; інколи у зарубіжній літературі такі методи називаються базованими на одному розв'язку чи стані (Single-Solution Based/Single-State Methods), а під траєкторними розуміють такий їх підклас, який породжує послідовність сусідніх розв'язків – траєкторію у просторі пошуку. Уникаючи певних термінологічних ускладнень, будемо всі такі методи називати траєкторними.

Алгоритми, які опрацьовують на кожній ітерації не один, а декілька розв'язків одночасно, називаються *популяційними* (Population-Based Methods).

Отже, для траєкторних алгоритмів $||Z|| = ||X_*|| = 1$, а для популяційних – $||Z|| > 1$, $||X_*|| > 1$.

За *складністю структури* АКО можна виділити:

– "прості" алгоритми;

– гібридні алгоритми;

– метаевристики[1];

– гібридні метаевристики

– гіперевристики.

Гіперевристикою (гіперевристичним алгоритмом) називають метод пошуку, який орієнтований на автоматизацію процесів вибору, комбінування або адаптації чи налаштування декількох більш простих алгоритмів (евристик чи метаевристик) для ефективного розв'язування ЗКО чи їх класів. Це може досягатися як вибором наявних евристик чи їх фрагментів, так і генеруванням нових.

Таким чином, якщо метаевристики та інші алгоритми здійснюють, в основному, пошук у просторі розв'язків ЗКО, то простором пошуку для гіперевристик є множина евристик (більш простих алгоритмів чи їх частин).

[1] Термін введено в F. Glover. Future paths for integer programming and links to artificial intelligence Computers & Operations Research, 13:533–549, 1986.

За *впливом на ландшафт* пошуку більшість АКО можна віднести до таких, що залишають його незмінним. В той же час, є алгоритми, які модифікують цей ландшафт шляхом:

– зміни простору розв'язків (наприклад, послідовні алгоритми);

– зміни цільової чи оцінкової функції (алгоритми керованого локального пошуку);

– варіації системи околів, що використовується при пошуку (алгоритми локального пошуку (ЛП) зі змінними околами, метод вектора спаду з пульсуючими околами).

Якщо робота алгоритму базується на безпосередніх даних ЗКО, то такі АКО відносяться до *задаче-орієнтованих алгоритмів.*

В ряді нових АКО використовуються не стільки прямі дані ЗКО, скільки спеціальна модель задачі (наприклад, феромонна матриця та матриця маршрутів у ОМК) – такі алгоритми отримали назву *моделе-орієнтованих*.

Точні алгоритми поділяються на загальні методи та спеціальні алгоритми.

Загальні методи:

– повний перебір (вичерпний пошук);

– метод гілок і меж (МГіМ);

– метод гілок і відтинань;

– послідовний аналіз варіантів (ПАВ, "київський віник");

– динамічне програмування (метод Беллмана).

Спеціальні алгоритми будуються на основі урахування специфіки задачі оптимізації, що розв'язується.

Приклад – метод Балаша (угорський метод) для розв'язання лінійної задачі про призначення.

2.2. Найбільш вживані на практиці наближені алгоритми.

Необхідність розробки ефективних наближених алгоритмів комбінаторної оптимізації, які і застосовуються у переважній більшості на практиці, визначається рядом обставин, серед яких:

1) практично всі важливі задачі відносяться до числа *NP-складних*, так що точне їх розв'язання вельми проблематично навіть з використанням сучасних і перспективних комп'ютерів;

2) їх цільові функції мають, як правило, велику кількість *локальних екстремумів*;

3) в багатьох прикладних проблемах *дані задаються з певними похибками*, що робить недоцільними ті істотні обчислювальні затрати, які необхідні для знаходження їх точного розв'язку;

4) покладені в основу розробки наближених обчислювальних схем ідеї (метаевристики) дозволяють створювати алгоритми, які можуть розв`язувати *не одну, а цілий клас* близьких за постановою оптимізаційних задач

5) у ряді задач значення цільової функції можуть бути доступними лише в процесі розв'язання задачі або ж вони можуть змінюватися з часом – *динамічні задачі чи on-line задачі*.

Найбільш вживані на практиці наближені алгоритми можна розділити на такі класи (рис. 2.2). Тут МГіМ – це метод гілок і меж, ПАВ – послідовний аналіз варіантів.

Зрозуміло, що сукупність всіх розроблених донині алгоритмів КО значно перевищує перелік наведених на цьому рисунку. Відмітимо ряд метаевристик, деякі із яких перебувають на початковому етапі свого дослідження і апробації.

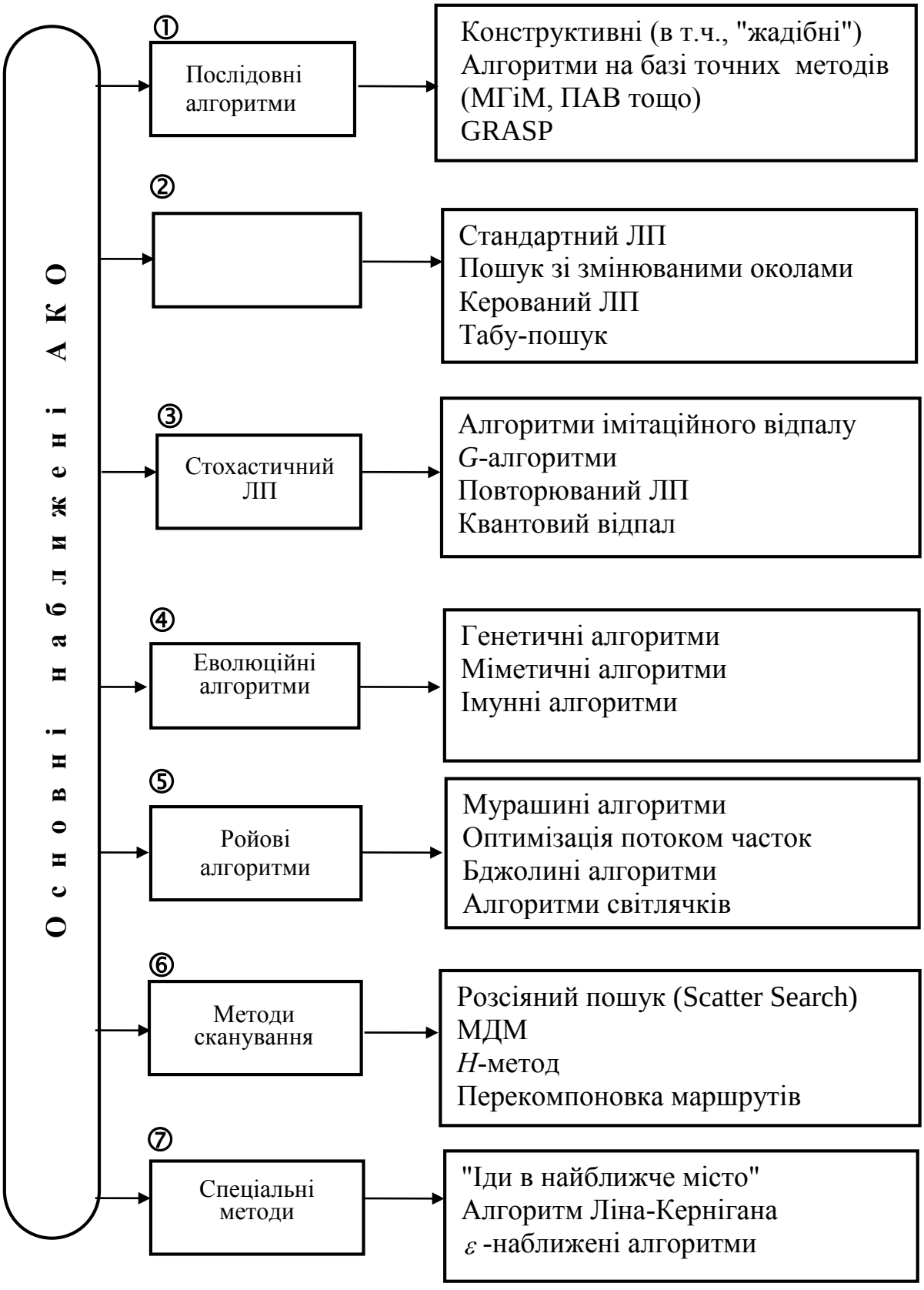

Рис. 2.2. Класифікація основних наближених методів комбінаторної оптимізації

2.3. Конструктивні алгоритми

Основна ідея конструктивних алгоритмів – побудова припустимого варіанту розв`язку задачі *шляхом нарощування* наявного в поточний момент його *часткового фрагменту*.

Більш формально, при застосуванні конструктивних алгоритмів до розв`язання задач виду (2):

- замість простору варіантів X розглядається більш широкий простір $Y \supset X$, для якого X, в більшості випадків, є границею. Наприклад, в ЗКО на перестановках часто Y – це множина всіх розміщень k елементів множини A $(k=1,\ldots, n)$;

- для кожного елементу $y \in Y$ вводиться поняття множини сусідніх елементів – певний аналог поняття околу;

- визначається функція $\varphi : Y \to R^1$, яка оцінює якість $\varphi(y)$ кожного варіанту $y \in Y$ з точки зору можливого значення цільової функції задачі (1) чи (2), причому природною є умова:

$$\varphi(x)=f(x), \ \forall x \in X.$$

Означення. Для довільного елемента $y \in Y \setminus X$ множиною сусідніх елементів є множина

$$N(y)=\{x \in Y: \ \|x\|-\|y\|=1\}.$$

Загальна схема конструктивних алгоритмів складається із таких кроків (заради простоти викладання розглядається задача безумовної оптимізації).

1) Початок. Випадковим чином чи згідно певним міркуванням, які враховують специфіку задачі, вибираємо один елемент множини A і включаємо його в фрагмент розв`язку $y \in Y$, тобто $\|y\|=1$.

Покладаємо $Z=\{y\}$.

2) Основна процедура.

2.1) Будуємо множину сусідніх до всіх чи деяких $z \in Y \setminus X$ і обчислюємо для них значення функції φ.

2.2) Серед всіх побудованих сусідніх вибираємо один чи декілька елементів, якім відповідають кращі значення функції φ.

2.3) Включаємо вибрані елементи в множину Z.

3) Перевірка критерію завершення.

Якщо умова завершення не виконується, то знову виконується основна процедура (п.2), інакше алгоритм припиняє свою роботу.

Критерієм завершення найчастіше виступає умова, що серед елементів множини Z хоча б один елемент (або всі ці елементи) належить простору X. Зрідка викладена схема розширюється шляхом старту в п.1 не з одного елемента, а відразу з декількох.

Наведена схема охоплює різноманітні евристичні алгоритми, що відрізняються правилами відбору елементів в п.2.2, заданням функції φ та визначенням критерію зупину. Конструктивні алгоритми можуть будуватися на основі таких відомих послідовних схем, як послідовний аналіз та відсіювання варіантів чи динамічного програмування.

Найбільшого поширення серед конструктивних алгоритмів набули так звані ***жадібні*** або ***гріді*** алгоритми, в схемі яких в п.2.2 обов`язково вибирається елемент, що доставляє екстремальне (максимальне) значення функції φ.

2.4. Метаевристики

Серед наближених оптимізаційних методів розв’язування ЗКО окремий клас утворюють метаевристики. За своєю природою метаевристики об'єднують простіші евристичні алгоритми в межах обчислювальних схем більш високого рівня, які спрямовані на ефективне вивчення простору пошуку. Отже, *метаевристика* – це метод розв’язування широкого класу обчислювальних задач шляхом такого комбінування існуючих процедур, при якому одна виступає провідною, а інша (чи декілька інших) – як підлегла. У якості як провідної, так і підлеглих процедур зазвичай виступають деякі відомі

евристики чи алгоритми. Якщо складовою метаевристики є певний математичний метод, то вживають термін *матевристика*.

Найбільш часто метаевристичні алгоритми застосовують при розв'язуванні ЗКО, проте вони також можуть бути використані і для таких задач, які зводяться до розв'язування логічних рівнянь.

Одним із таких класів, що розвивається найбільш динамічно, є метаевристики, обчислювальна схема яких навіяна природою.

Аналіз існуючих метаевристик дозволяє здійснити таку класифікацію, яка доповнює і розвиває наведену на рис.2.2.

1. Еволюційні методи.

1.1. Генетичні алгоритми, серед яких виділяють алгоритми з бінарним кодуванням та з дійсним кодуванням, а також ті, які працюють у комбінаторних просторах з кодуванням, яке не зводиться до названих типів.

1.2. Метаевристики га основі штучних імунних систем.

1.3. Методи диференціальної еволюції.

II. Методи "ройового" інтелекту.

1. Оптимізації потоком часток (Particle Swarm Optimization).

2. Оптимізація мурашиними колоніями (Ant Colony Optimization).

3. Метод імітації поведінки бактерій (Bacterial Foraging Optimization).

4. Методи бджолиних колоній (Bees Algorithms, Artificial Bee Colony):

– метод бджолиного рою;

– метод штучної бджолиної колонії.

5. Пошук на основі імітації поведінки зграї риб в пошуках корму (Fish School Search).

6. Алгоритм, який імітує поведінку летючих мишей (Bat-Inspired Algorithm).

7. Метод імітації поведінки світлячків (Glowwarm Swarm Optimization).

8. Алгоритм імітації поведінки жаб (Shuffled Frog-Leaping Algorithm).

III. Методи, які імітують фізичні процеси

1. Метод гравітаційної кінематики (Central Force Optimization).

2. Метод імітаційного відпалу (Simulated Annealing).

3. Адаптивний метод імітаційного відпалу.

4. Метод пошуку гармонії (Harmony Search).

5. Метод, який використовує закон електромагнетизму (Electromagnetism-like Mechanism).

IV. Мультистартові методи

1. Жадібний алгоритм адаптивного випадкового пошуку (Greedy Randomized Adaptive Search Procedure).

2. Метод спрямованого табу-пошуку (Tabu Search).

СПИСОК РЕКОМЕНДОВАНИХ ДЖЕРЕЛ

1. Гуляницький Л.Ф. Диверсифікація пошуку в алгоритмах ОМК // Abstracts of Int. Conf "Problems of Decision Making under Uncertainties (PDMU–2011)" (September 19-23, 2011, Yalta, Ukraine). – Київ, 2011. – P.66–67.

2. Сергиенко И.В. Классификация прикладных методов комбинаторной оптимизации / И.В. Сергиенко, Л.Ф. Гуляницкий, С.И. Сиренко // Кибернетика и системный анализ. — 2009. — № 5. — С. 71-83.

3. Blum, C., Puchinger J., Raid, G. R., Roli A. Hybrid metaheuristics in combinatorial optimization: A survey // Applied Soft Computing. – 2011. – 11, 6. – P. 4135-4151.

4. Hulianytskyi L.F., Sirenko S.I. Cooperative model-based metaheuristics // Electronic Notes in Discrete Mathematics. – 2010. – 36. – P. 33-40.

5. Raidl G.R. A unified view on hybrid metaheuristics // Lect. Notes Computer Sci. – Berlin: Springer-Verlag, 2006. – 4030. – P. 1–12.