КІБЕРНЕТИКА та КОМП'ЮТЕРНІ ТЕХНОЛОГІЇ

УДК 519.854:004.023 DOI:10.34229/2707-451X.21.3.1

Н.М. ГУЛАЄВА, В.П. ШИЛО, М.М. ГЛИБОВЕЦЬ

ГЕНЕТИЧНІ АЛГОРИТМИ ЯК ОБЧИСЛЮВАЛЬНІ МЕТОДИ СКІНЧЕННОВИМІРНОЇ ОПТИМІЗАЦІЇ

Дано огляд основних класів екстремальних задач та алгоритмів їх розв'язання. Окреслено клас задач, які можуть бути розв'язані за допомогою генетичних алгоритмів, та показано спосіб зведення заданої оптимізаційної задачі до задачі, розв'язуваної за допомогою генетичного алгоритму. Показано, що генетичні алгоритми розв'язання екстремальних задач є стохастичними чисельними методами прямого пошуку.

Ключові слова: *задача математичного програмування, задача безумовної оптимізації, задача умовної оптимізації, задача багатоекстремальної оптимізації, чисельні методи, генетичні алгоритми, метаевристичні алгоритми.*

Постановка задачі оптимізації. В загальному випадку під скінченновимірною екстремальною задачею розуміють задачу відшукання точної верхньої чи нижньої межі деякої функції $f: X \to R$, де $X \subseteq R^n$.

Нагадаємо основні визначення.

Функція $f: X \to R$, $X \subseteq R^n$, називається *обмеженою знизу* (*зверху*) на множині X, якщо існує дійсне число M таке, що $\forall x \in X: f(x) \geq M\,(f(x) \leq M)$.

Нехай функція $f(x)$ – *обмежена знизу* на множині $X \subseteq R^n$. Тоді дійсне число f^* називається *точною нижньою межею* функції $f(x)$ на X, якщо виконуються дві умови:

1) $\forall x \in X: f^* \leq f(x)$;

2) для будь-якого як завгодно малого числа $\varepsilon > 0$ знайдеться точка $x_\varepsilon \in X$ така, що $f(x_\varepsilon) \leq f^* + \varepsilon$.

Якщо функція $f(x)$ – не обмежена знизу на X, то точна нижня межа вважається $f^* = -\infty$. Точна нижня межа f^* функції $f(x)$ на X позначається $\inf\limits_{x \in X} f(x)$.

Аналогічно вводяться поняття *точної верхньої межі* функції $f(x)$ на $X \subseteq R^n$. Якщо функція $f(x)$ – не обмежена зверху на X, то точна верхня межа вважається $f^* = +\infty$. Точна верхня межа f^* функції $f(x)$ на X позначається $\sup\limits_{x \in X} f(x)$.

Точка $x^* \in X$ називається *точкою локального мінімуму* функції $f(x)$ на множині X, якщо існує число $\varepsilon > 0$ таке, що

$$\forall x \in X \cap O_\varepsilon(x^*): f(x^*) \leq f(x), \qquad (1)$$

де $O_\varepsilon(x^*)$ – ε-окіл точки x^*.

Точка $x^* \in X$ називається *точкою глобального мінімуму* функції $f(x)$ на множині X, якщо

$$\forall x \in X: f(x^*) \leq f(x). \qquad (2)$$

При цьому записують: $f(x^*) = \min\limits_{x \in X} f(x)$ або $x^* = \arg\min\limits_{x \in X} f(x)$.

Множину всіх точок глобального мінімуму позначають так:

$$\operatorname{Arg}\min\limits_{x \in X} f(x) = \{x^* \in X \mid f(x^*) = \min\limits_{x \in X} f(x)\}.$$

Точка глобального мінімуму завжди є точкою локального мінімуму, але не навпаки. Якщо нерівність в (1), (2) виконується як строга за $x \neq x^*$, то x^* називається відповідно *точкою строгого локального* чи *строгого глобального мінімуму*.

Замінюючи в наведених означеннях слово «мінімум» на «максимум» і знак нерівності в (1), (2) на протилежний, отримаємо поняття *точок глобального* та *локального максимумів*, а також поняття *точок строгого локального* та *строгого глобального максимумів*. Зауважимо, що максимум і мінімум об'єднуються єдиним терміном *екстремум*.

Очевидно, що $\inf\limits_{x \in X} f(x)$ та $\sup\limits_{x \in X} f(x)$ завжди існують, в той час як $\min\limits_{x \in X} f(x)$ та $\max\limits_{x \in X} f(x)$ існують не завжди. Залежно від властивостей множини X та функції $f(x)$ множина всіх точок глобального мінімуму (максимуму) $\operatorname{Arg}\min\limits_{x \in X} f(x)$ ($\operatorname{Arg}\max\limits_{x \in X} f(x)$) може бути скінченною, нескінченною або пустою.

В загальному випадку *скінченновимірну екстремальну задачу* записують так:

$$f(x) \to \inf(\sup), x \in X \subseteq R^n. \tag{3}$$

Найбільш широко вживаними трактуваннями поняття «розв'язок» задачі (3) є такі:

A) визначити точну нижню (верхню) межу $f^* = \inf\limits_{x \in X} f(x)$ ($f^* = \sup\limits_{x \in X} f(x)$);

B) визначити точну нижню (верхню) межу $f^* = \inf\limits_{x \in X} f(x)$ ($f^* = \sup\limits_{x \in X} f(x)$), якщо множина точок глобального мінімуму (максимуму) не є пустою, знайти хоча б одну точку $x^* = \arg\min\limits_{x \in X} f(x)$ ($x^* = \arg\max\limits_{x \in X} f(x)$);

C) визначити точну нижню (верхню) межу $f^* = \inf\limits_{x \in X} f(x)$ ($f^* = \sup\limits_{x \in X} f(x)$), якщо множина точок глобального мінімуму (максимуму) не є пустою, знайти всі точки $\operatorname{Arg}\min\limits_{x \in X} f(x)$ ($\operatorname{Arg}\max\limits_{x \in X} f(x)$);

D) знайти всі точки локальних мінімумів (максимумів) та значення функції у цих точках.

Часто буває відомо, що множина точок $\operatorname{Arg}\min\limits_{x \in X} f(x) \neq \varnothing$ ($\operatorname{Arg}\max\limits_{x \in X} f(x) \neq \varnothing$). У цьому разі визначення екстремальної задачі та поняття розв'язку цієї задачі відрізняється.

Задачу мінімізації функції $f(x)$ на множині X записують у вигляді

$$f(x) \to \min, x \in X \text{ або } \min\limits_{x \in X} f(x). \tag{4}$$

Функція $f(x)$ називається *цільова функція*, множина X називається *допустима множина*, будь-який елемент $x \in X$ називається *допустима точка* цієї множини. Якщо $X = R^n$,

то задача пошуку мінімуму називається *задача безумовної мінімізації*; якщо $X \subset R^n$, то задача пошуку мінімуму називається *задача умовної мінімізації* або *задача з обмеженнями на змінні*.

Поняття розв'язку задачі (4) за аналогією з розв'язком задачі (3) визначають так:

B') знайти хоча б одну точку $x^* = \arg\min\limits_{x \in X} f(x)$;

C') знайти всі точки $\operatorname{Arg}\min\limits_{x \in X} f(x)$;

D') знайти всі точки локальних мінімумів та значення функції у цих точках.

Зауважимо, що у варіантах C'), D') загалом вимагається знайти не одну точку, а цілу множину точок, тому задачу (4) у варіантах C'), D') ще називають *задачею багатоекстремальної оптимізації* (*multimodal optimization*). Тут слід відмітити певну неоднозначність у термінології: у вітчизняній літературі часто під задачею багатоекстремальної оптимізації розуміють задачу в постановці B') для випадку, коли цільова функція $f(x)$ має багато екстремумів.

За аналогією з (4) задачу максимізації функції $f(x)$ на множині X записують у вигляді

$$f(x) \to \max, x \in X \text{ або } \max_{x \in X} f(x). \tag{5}$$

Очевидно, що $\max\limits_{x \in X} f(x) = -\min\limits_{x \in X}(-f(x))$, отже, для задачі максимізації (5) та задачі мінімізації $-f(x) \to \min,\ x \in X$, множини глобальних та локальних екстремумів збігаються, таким чином, задача максимізації – еквівалентна задачі мінімізації. Тому надалі розглядатимемо лише задачі мінімізації.

Класифікація задач оптимізації. Класифікацію задач оптимізації проводять за такими ознаками: тип і властивості цільової функції, способи подання та властивості допустимої множини. Залежно від класу задачі, на основі апріорної інформації про властивості задачі обирають конкретний метод її розв'язання. Серед основних класів оптимізаційних задач виділимо такі.

Класичну задачу на умовний екстремум: допустима множина X визначається системою рівнянь. Як правило, задачу записують у вигляді

$$f(x) \to \min(\max),\ g_i(x) = 0, i = \overline{1,m}.$$

Найпоширеніший метод розв'язання цієї задачі за умови неперервності та неперервної диференційовності функцій $f(x)$, $g_i(x)$, $i = \overline{1,m}$, це метод множників Лагранжа.

Задачу математичного програмування: допустима множина X визначається системою нерівностей та рівнянь. Цю задачу записують у вигляді

$$f(x) \to \min(\max),\ g_i(x) \le 0, i = \overline{1,k},\ g_i(x) \ge 0, i = \overline{k+1,m},\ g_i(x) = 0, i = \overline{m+1,s},\ x \in P \subset R^n.$$

Залежно від властивостей функцій $f(x)$, $g_i(x)$ серед задач математичного програмування виділяють такі класи задач.

Задачу лінійного програмування: функції $f(x)$, $g_i(x)$, $i = \overline{1,s}$ – лінійні. Основний метод розв'язування задач цього класу – симплекс-метод.

Задачу нелінійного програмування: функції $f(x)$ та/або $g_i(x)$ – нелінійні. Серед цих задач найбільш добре вивченими є задачі *опуклого програмування*. Залежно від диференційованості цільової функції розглядають задачі *диференційованої* або *гладкої оптимізації*, задачі *недиференційованої* або *негладкої оптимізації*. Для розв'язування задач негладкої опуклої оптимізації використовують, наприклад, субградієнтні і ε-субградієнтні методи, метод проекції субградієнта

тощо. Серед задач опуклого програмування виділяють задачі *квадратичного програмування* та задачі *сепарабельного програмування*.

Важливе місце серед прикладних задач математичного програмування посідають задачі *дискретної оптимізації*. В цих задачах допустима множина $X \subset R^n$ – дискретна. Якщо всі невідомі змінні можуть набувати лише цілочислових значень, $X \subset Z^n$, йде мова про задачу *цілочислового програмування*. Умова цілочисельності обумовлена, перш за все, практичною природою задач, адже в умовах реальності багато ресурсів може набувати лише цілих значень (верстати, машини, рулони тощо). Якщо $X = \{0,1\}^n$, йде мова про задачу *булевої оптимізації*, при цьому, у разі $f(x) \in R$, говорять про *псевдобулеву оптимізацію*. В задачах *комбінаторної оптимізації* значення цільової функції залежать від певної комбінації об'єктів зі скінченного набору, їх розміщення або способу упорядкування. Незважаючи на поширеність задач комбінаторної оптимізації, формальне визначення цього класу задач запропоновано нещодавно в [1]. Серед методів розв'язання задач дискретного програмування виділимо метод відтинання, метод гілок і меж, метод динамічного програмування, послідовний аналіз варіантів («київський віник»). Ґрунтовний огляд сучасного стану та методів розв'язання задач дискретної оптимізації дано в [1–3].

Існують також інші класи задач оптимізації: параметричні, детерміновані, стохастичні, нечіткі та інші. Бібліографію методів розв'язання різних класів задач математичного програмування можна знайти в [4].

Класифікація методів розв'язання задач оптимізації. Методи розв'язання задач оптимізації класифікують за різними характеристиками. Один з найпоширеніших це поділ методів оптимізації на такі:

1) *аналітичні*, які ґрунтуються на використанні умов екстремуму;

2) *методи обчислювальної оптимізації* (*чисельні методи*), в яких для організації процесу пошуку розв'язку проводять вимірювання (обчислення) значень локальних характеристик цільових функцій та функцій обмежень (тобто значень функцій, градієнтів, гессіанів).

Аналітичні методи не завжди застосовні на практиці, оскільки вимоги, що накладаються на цільову функцію та функції-обмеження (аналітичний вид функції, кускова гладкість, можливість обчислення похідної та ін.) на практиці виконуються дуже рідко. Прикладами аналітичних методів є методи Лагранжа, Куна – Таккера.

Чисельний метод у загальному випадку є ітераційною процедурою, яка в точках пошукового простору здійснює обчислення певних характеристик функції (значення функції, її градієнта, гессіана). Залежно від того, які саме характеристики функцій обчислюються, чисельні методи поділяють на методи прямого пошуку, першого порядку (градієнтні), другого порядку. Слідуючи [5], називатимемо операцію обчислення характеристик функції у точці *пошуковим випробуванням*, а сукупність значень характеристик у цій точці – *результатом випробування* або *пошуковою інформацією*. В методах *прямого пошуку* в точках пошукового простору обчислюють тільки значення цільової функції, тобто результатом випробування є значення цільової функції у точці випробування. За [5], прямий чисельний метод c розв'язання оптимізаційної задачі (4) з класу Ф можна подати набором

$$c = \langle \{G_k\}, \{E_k\}, \{H_k\} \rangle, \tag{6}$$

де $\{G_k\}$, $k = 1,2,\ldots$ – сім'я функціоналів, яка описує сукупність правил вибору точок випробувань, $\{E_k\}$, $k = 1,2,\ldots$ – послідовність відображень, яка задає сукупність правил побудови наближеного розв'язку, $\{H_k\}$, $k = 1,2,\ldots$ – сукупність правил зупинки обчислювального процесу.

Загальна схема прямого чисельного методу така.

1. Обрати точку першого випробування: $x^1 = G_1(\Phi) \in X$.

2. Нехай обрано точку k-го випробування $x^k = G_k(\Phi) \in X$, $k \geq 1$. Після обчислення значення цільової функції у цій точці $y^k = f(x^k)$, отримаємо пошукову (апостеріорну) інформацію про функцію f: $\omega_k = \{(x^1, y^1), (x^2, y^2), ..., (x^k, y^k)\}$. Отримана інформація дозволяє звузити клас Φ, до якого належить функція $f(x)$, до множини $\Phi(\omega_k) = \{\varphi \in \Phi : \varphi(x^i) = y^i, 1 \leq i \leq k\}$.

3. Визначити поточну оцінку екстремуму (наближений розв'язок): $e_k = E_k(\Phi, \omega_k)$.

4. Обчислити точку наступного випробування: $x^{k+1} = G_{k+1}(\Phi, \omega_k) \in X$.

5. Визначити величину $h_k = H_k(\Phi, \omega_k) \in \{0,1\}$. Якщо $h_k = 1$, то збільшити номер кроку k на одиницю та перейти на п. 2. Інакше ($h_k = 0$) зупинити обчислення. Розв'язок задачі має оцінку e_k .

Наприклад [5], метод перебору значень по вузлах рівномірної сітки для розв'язування одновимірної задачі мінімізації

$$f(x) \to \min, x \in [a,b] \subseteq R,$$

можна описати так (n – кількість вузлів сітки): Φ – клас функцій, визначених на відрізку [a, b] та обчислюваних у кожній точці цього відрізку, $G_1(\Phi) = a$, $G_{k+1}(\Phi, \omega_k) = a + k\frac{b-1}{n}$, $k \geq 1$,

$H_k(\Phi, \omega_k) = \begin{cases} 0, k = n+1 \\ 1, k < n+1 \end{cases}$, $E_k(\Phi, \omega_k) = f_k^* = \min\limits_{1 \leq i \leq k} f(x^i)$.

Генетичні алгоритми як чисельні методи оптимізації. Генетичні алгоритми, в основу яких покладено ідею еволюції живої природи, широко використовуються для розв'язання різноманітних задач оптимізації. Генетичні алгоритми не вимагають аналітичного завдання цільової функції та не накладають жодних обмежень на її властивості (неперервність, диференційовність та ін.), єдиним обмеженням, що накладається на цільову функцію, є обчислюваність у кожній точці пошукового простору. Серед основних властивостей генетичних алгоритмів зазначимо такі [6]: понятійний апарат запозичений з біології, кожен допустимий розв'язок задачі кодується у вигляді хромосоми та представляється особиною популяції, для оцінювання якості розв'язку обчислюють його коефіцієнт пристосованості (за допомогою функції пристосованості), пошук розв'язків здійснюється одночасно цілою популяцією особин, причому для генерації нового розв'язку використовують аналоги природних генетичних операторів. В загальному випадку генетичні алгоритми не можуть бути застосовними до задач, для яких множина $\operatorname{Arg}\min\limits_{x \in X} f(x) = \varnothing$, отже, генетичні алгоритми застосовні лише до задач в постановці B'), C'), D').

Розглянемо задачу безумовної оптимізації

$$f(x) \to \min, x \in X = R^n. \quad (7)$$

Для розв'язування задачі (7) за допомогою генетичного алгоритму необхідно визначити:

1) відображення (бієктивне) $\mu : X \to S$, яке зіставляє кожному допустимому розв'язку задачі $x \in X = R^n$ код $s \in S$; очевидно, обернене відображення $\mu^{-1} : S \to X$ за кодом $s \in S$ відновлюватиме розв'язок $x \in X$. Множина S може збігатись з множиною X або бути визначеною, наприклад, як множина послідовностей довжини n елементів певної множини;

2) функцію пристосованості $F(s), s \in S$, яка має задовольняти таким вимогам [6]:

- адекватність задачі: $\forall s_1, s_2, x_1 = \mu^{-1}(s_1), x_2 = \mu^{-1}(s_2)$ виконується $F(s_1) < F(s_2) \Leftrightarrow$ $\Leftrightarrow f(x_1) < f(x_2)$, де s_1, s_2 – закодовані значення розв'язків x_1, x_2, F – функція пристосованості, f – цільова функція оптимізаційної задачі;
- маловитратність (невисока складність обчислення);
- різноманітний рельєф.

Часто покладають $F(s) \equiv f(x)$, де s – хромосома, що кодує значення розв'язку x ($s = \mu(x)$).

Розв'язування задачі (7) зводиться до розв'язування за допомогою генетичного алгоритму задачі

$$F(s) \to \min, s \in S. \tag{8}$$

Якщо s^* – розв'язок задачі (8), то розв'язком задачі (7) буде $x^* = \mu^{-1}(s^*)$.

Генетичні алгоритми природно розглядати як чисельні методи прямого пошуку. Розглянемо, наприклад, генетичний алгоритм турнірного витиснення з гауссовою мутацією, запропонований в [7] для розв'язування задач (4), (5) в постановці D'), $X = X_1 \times X_2 \times ... \times X_n$, $X_i = [a_i, b_i] \subseteq R \ \forall i = \overline{1,n}$. Схему цього алгоритму для розв'язання задач в постановці B') можна подати так.

1. Кодування в дійсних числах ($S = X$): кожен ген хромосоми $x = (x_1, x_2, ..., x_n) \in R^n$ – дійсна змінна $x_i \in [a_i, b_i] \subseteq R$.
2. Генерація початкової популяції за рівномірним розподілом.
3. Обчислення коефіцієнта пристосованості всіх особин популяції, функція пристосованості збігається з цільовою функцією.
4. Породження L нащадків кожною особиною шляхом застосування оператора гауссової мутації до кожного гена хромосоми: $x_i = x_i + N_i(0, \sigma)$, де $N_i(0, \sigma)$ – стандартна нормально розподілена випадкова величина.
5. Обчислення коефіцієнта пристосованості отриманих нащадків.
6. Формування нової популяції за допомогою турнірного відбору, група для кожного турніру формується з батька та усіх його нащадків.
7. Якщо кількість ітерацій дорівнює G, завершення роботи алгоритму (розв'язком задачі є найкраща особина популяції), інакше перехід на крок 4.

Наведений алгоритм можна подати як паралельне виконання N екземплярів алгоритмів, кожен з яких описується схемою прямого чисельного методу c (6):

Ф – клас функцій, визначених на R^n та обчислюваних у кожній точці,

$G_1(\Phi) = \xi$, де ξ – послідовність випадкових дійсних чисел,

$$G_{k+1}(\Phi, \omega_k) = \arg\min \left\{ f(x^k), \min_{i=1,2,...,L} \{ f(x_1^k + N_{1,i}^k(0,\sigma), ..., x_n^k + N_{n,i}^k(0,\sigma)) \} \right\}, \ k \geq 1,$$

$$H_k(\Phi, \omega_k) = \begin{Bmatrix} 0, k = G \\ 1, k < G \end{Bmatrix},$$

$E_k(\Phi, \omega_k) = f_k^* = \min\limits_{1 \leq l \leq N} f(x^{k,l})$, де $x^{k,l}$ – точка випробування l-го алгоритму на k-ій ітерації.

Генетичний алгоритм турнірного витиснення з гауссовою мутацією легко транслюється у схему прямого чисельного методу завдяки таким особливостям: кодування у дійсних числах (кожен параметр задачі задається відповідним геном хромосоми), відсутність оператора кросинговеру, локальний відбір особин у наступну популяцію (проводять турнірні змагання між батьками та їх власними нащадками). В загальному випадку $S \neq X$, для породження нащадків використовується не лише оператор мутації, але й кросинговер, для реалізації відбору слід урахувати інформацію

про всіх особин популяції. Тому для подання довільного генетичного алгоритму як прямого чисельного методу наведена схема прямого чисельного методу *c* (6) вимагає певних модифікацій. При проведенні модифікації схеми важливо також урахувати паралелізм генетичних алгоритмів, адже в генетичних алгоритмах пошук розв'язків здійснюється одночасно популяцією особин. Іншими словами, на кожній ітерації роботи генетичного алгоритму аналізується множина точок пошукового простору, кількість таких точок дорівнює розміру популяції та є сталою. Отже, для представлення довільного генетичного алгоритму розв'язування задачі (8) як чисельного методу необхідно, щоб у схемі прямого чисельного методу *c* (6) на кожному кроці розглядалась не одна, а *N* точок випробування. Пропоновану модифікацію схеми *c* подано далі.

1. Обрати множину точок першого випробування (в термінах генетичного алгоритму: згенерувати початкову популяцію): $< s_1^1, s_2^1, ..., s_N^1 >$, $s_i^1 = G_1(\Phi) = \xi_i$, де ξ_i – випадкова величина (залежно від множини *S* це може бути випадкове дійсне число, послідовність довжини *n* випадкових дійсних чисел, випадкова двійкова послідовність довжини *n* та ін.).

2. Нехай обрано множину точок *k*-го випробування $< s_1^k, s_2^k, ..., s_N^k >$, $k \geq 1$. Після обчислення значення функції пристосованості в кожній з цих точок $y_i^k = F(s_i^k)$ отримаємо пошукову (апостеріорну) інформацію про функцію *F*: $\omega_k = \{(s_1^k, y_1^k), (s_2^k, y_2^k), ..., (s_N^k, y_N^k)\}$. Звернемо увагу, що апостеріорна інформація про функцію *F* містить тільки інформацію, отриману на поточному кроці роботи алгоритму (на відміну від оригінальної схеми *c*, в якій запам'ятовується вся інформація, отримана від початку роботи алгоритму).

3. Визначити поточну оцінку екстремуму: $e_k = E_k(\Phi, \omega_k) = \min\limits_{1 \leq i \leq N} y_i^k$.

4. Для обчислення множини точок наступних випробувань до поточної множини точок $< s_1^k, s_2^k, ..., s_N^k >$ застосувати послідовність операторів: відбір до батьківського пулу $< p_1^k, p_2^k, ..., p_N^k >= G_{k+1}^{SEL}(< s_1^k, s_2^k, ..., s_N^k >, \omega_k)$, кросинговер $< c_1^k, c_2^k, ..., c_N^k >= G_{k+1}^{CROS}(< p_1^k, p_2^k, ..., p_N^k >)$, мутація $< s_1^{k+1}, s_2^{k+1}, ..., s_N^{k+1} >= G_{k+1}^{MUT}(< c_1^k, c_2^k, ..., c_N^k >)$. Іншими словами, $G_{k+1}(\Phi, \omega_k)$ – послідовність спеціальних генетичних операторів. Відбір відкидає деякі «невдалі» розв'язки, та, натомість, дублює «вдалі». Кросинговер здійснює обмін частинами розв'язків, а мутація фактично – локальне перетворення розв'язку (збурення).

5. Визначити величину $h_k = H_k(\Phi, \omega_k) \in \{0,1\}$. Якщо $h_k = 1$, то збільшити номер кроку *k* на одиницю та перейти на п. 2. Інакше ($h_k = 0$) зупинити обчислення. Розв'язок задачі має оцінку e_k. В більшості генетичних алгоритмів $H_k(\Phi, \omega_k) = 0$, якщо $k = G$, де *G* – максимальна кількість ітерацій (поколінь), або ідентифіковано збіжність генетичного алгоритму, наприклад,

$$\left| e_k - \sum_{i=1}^{N} y_i^k \Big/ N \right| < \varepsilon .$$

Наведена схема – загальна та потребує уточнень залежно від обраного класу генетичного алгоритму (нові класи генетичних алгоритмів будують шляхом уточнення генетичних операторів). Також, схема описує генетичні алгоритми з генераційним типом репродукції, опис алгоритмів зі стійким типом репродукції вимагає незначної модифікації. Підсумовуючи сказане та враховуючи стохастичність операторів G_{k+1}^{SEL}, G_{k+1}^{CROS}, G_{k+1}^{MUT}, можна говорити про генетичні алгоритми як про стохастичні чисельні методи прямого пошуку.

Генетичні алгоритми в задачах умовної оптимізації. Досі ми обговорювали генетичні алгоритми розв'язання задач безумовної оптимізації (7). В задачах умовної оптимізації допустима множина $X \subset R^n$. Загальні методи урахування обмежень у задачах математичного програмування детально описані, наприклад, в [5]: методи штрафних функцій, множників Лагранжа, параметризації цільової функції та ін. Для врахування обмежень на допустимі розв'язки в генетичних алгоритмах застосовують такі підходи:

1) вибір такого способу кодування, за якого всі значення з множини S декодуються в допустимі точки: $\forall s \in S:\ x = \mu^{-1}(s) \in X \subset R^n$. Зазначимо, що на практиці розробити такий спосіб кодування не завжди вдається;

2) розробка спеціальних генетичних операторів (кросинговер, мутація), застосування яких гарантує породження допустимих розв'язків. Тобто, після застосування генетичного оператора породжений ним розв'язок буде допустимим (нова точка випробування задовольняє обмеженням на розв'язок);

3) застосування модифікації широко використовуваного в інших чисельних методах оптимізації методу штрафних функцій. В цьому разі цільова функція задачі (8) замінюється функцією

$$F_p(s) = F(s) + \gamma P(s), s \in S .$$

Тут $F_p(s)$ – нова цільова функція, $P(s) = \begin{cases} 0, & x = \mu^{-1}(s) \in X \\ > 0, & x = \mu^{-1}(s) \notin X \end{cases}$ – функція штрафу, γ – коефіцієнт штрафу. Така заміна цільової функції дозволяє погіршити оцінку якості розв'язку (оцінку точки випробування), в якому порушуються задані умовою обмеження. Відмінність методу штрафних функцій у генетичних алгоритмах полягає у тому, що значення коефіцієнтів штрафу γ підбираються експертним або експериментальним шляхом і способи підбору цих коефіцієнтів чітко не регламентуються.

Нескладно бачити, що всі зазначені підходи не потребують додаткових змін пропонованої вище схеми подання генетичного алгоритму як чисельного методу прямого пошуку.

Висновки. З огляду на велику кількість різноманітних методів оптимізації науковці все більше уваги приділяють класифікації та систематизації цих методів. Питанню класифікації методів оптимізації присвячено роботи [3, 8–10]. Розглянута нами класифікація методів оптимізації на аналітичні та чисельні аналізує тип задіяних обчислень, тому проста й зрозуміла, але далеко не єдина. Інша популярна класифікація методів оптимізації є класифікація за типом отримуваного розв'язку, в цьому разі алгоритми поділяють на точні (за скінченний час гарантовано повертають оптимальний розв'язок або дають висновок, що розв'язку не існує), наближені (за скінченний час гарантовано повертають розв'язок, для якого може бути отримана асимптотична оцінка точності), евристичні (відсутня оцінка точності отриманого розв'язку та навіть гарантія отримання будь-якого розв'язку за скінченний час). За ступенем математичного обґрунтування алгоритми поділяють на раціональні та евристичні, за принципом прийняття рішень – на детерміновані та стохастичні, за складністю структури – на прості, метаевристичні, гіперевристичні тощо. Часто дослідники як окремий клас розглядають алгоритми, породжені ідеєю імітації природних явищ (різноманітні роєві та еволюційні алгоритми). Хоча ідея виділення в окремий клас алгоритмів, на створення яких авторів надихнуло спостереження за навколишнім середовищем, може здатись цілком природною, така класифікація не розкриває принципів роботи самих алгоритмів, їхньої суті. В роботі [3] пропонується проводити класифікацію методів комбінаторної оптимізації за вісьмома характеристиками: принцип прийняття розв'язку, складність структури, тип траєкторії, що формується алгоритмом, тип використовуваних просторів, вплив на ландшафт пошуку, використання пам'яті, наявність адаптації/навчання, наявність спеціальної моделі задачі. Ця класифікація

є ґрунтовною, але породжує надвелику кількість різних класів та підкласів. Так, за цією класифікацією генетичні алгоритми є стохастичними метаевристичними траєкторно-розривними ітераційними (популяційними) алгоритмами без зміни ландшафту пошуку, без пам'яті, в загальному випадку без адаптації, завдання-орієнтованими.

Все сказане наштовхує на думку, що наразі стан теорії методів розв'язання оптимізаційних задач нагадує стан теорії штучних нейронних мереж кілька років тому: кількість розроблених методів оптимізації є надзвичайно великою та постійно зростає, причому досить поширеною є практика надання нових назв раніше відомим поняттям або незначним модифікаціям раніше розроблених методів без посилання на відповідні роботи. Такий стан справ пояснюється надзвичайною розгалуженістю різних напрямів досліджень і вузькою спеціалізацією дослідників. Особливо проблемним є напрям розробки метаевристичних алгоритмів, де в гонитві за науковою новизною автори створюють все нові алгоритми, не даючи їхнього математичного обґрунтування або більш-менш вичерпного експериментального аналізу і порівняння з вже існуючими алгоритмами. Про необхідність виправлення ситуації вже звітують деякі науковці [10].

Підсумовуючи сказане, констатуємо необхідність перегляду та переосмислення історії виникнення, розробки й модифікацій різних методів оптимізації для узагальнення і систематизації цих методів. В теорії штучних нейронних мереж такою роботою автори вважають роботу [11], в якій дано не просто огляд історії розвитку штучних нейронних мереж, але й аналіз історії основних ідей глибокого навчання, відстеження розвитку конкретних структур та алгоритмів. Автори сподіваються, що наведене представлення генетичного алгоритму (метаевристичного, популяційного, породженого ідеєю імітації природних явищ тощо) як чисельного методу прямого пошуку може стати першим кроком на цьому шляху.

Список літератури

1. Hulianytskyi L., Riasna I. Formalization and classification of combinatorial optimization problems. In honor of Ivan V. Sergienko's 80th birthday. *Optimization Methods and Applications* / Butenko S., Pardalos P., Shylo V. (eds). Cham : Springer, 2017. Vol. 130. P. 239–250. https://doi.org/10.1007/978-3-319-68640-0_11
2. Сергиенко И.В., Шило В.П. Проблемы дискретной оптимизации: сложные задачи, основные подходы к их решению. *Кибернетика и системный анализ*. 2006. Т. 42, № 4. С. 3–25. https://doi.org/10.1007/s10559-006-0086-3
3. Сергиенко И.В., Гуляницкий Л.Ф., Сиренко С.И. Классификация прикладных методов комбинаторной оптимизации. *Кибернетика и системный анализ*. 2009. Т. 45, № 5. С. 71–83. https://doi.org/10.1007/s10559-009-9134-0
4. Жалдак М.І., Триус Ю.В. Основи теорії і методів оптимізації : навчальний посібник. Черкаси : Брама-Україна, 2005. 608 с.
5. Городецкий С.Ю., Гришагин В.А. Нелинейное программирование и многоэкстремальная оптимизация: учебное пособие. Нижний Новгород : Изд-во Нижегородского гос. ун-та, 2007. 489 с.
6. Глибовець М.М., Гулаєва Н.М. Еволюційні алгоритми : підручник. Київ : НаУКМА, 2013. 828 с.
7. Шило В.П., Глибовець М.М., Гулаєва Н.М., Нікіщіхіна К.В. Генетичні алгоритми турнірного витиснення з гауссовою мутацією. *Кибернетика и системный анализ*. 2020. Т. 56, № 2. С. 75–88. https://doi.org/10.1007/s10559-020-00239-4
8. Birattari M., Paquete L., Stützle T., Varrentrapp K. Classification of metaheuristics and design of experiments for the analysis of components: technical report. Darmstadt : Techn. Univ. Darmstadt, 2001. AIDA-01-05. 12 p.
9. Певнева А.Г. Построение классификации методов глобальной оптимизации. *Международный научно-исследовательский журнал*. 2012. № 3. С. 14–23.
10. Sörensen K. Metaheuristics – the metaphor exposed. *International Transactions in Operational Research*. 2015. Vol. 22, N 1. P. 3–18. https://doi.org/10.1111/itor.12001
11. Schmidhuber J. Deep learning in neural networks: an overview. *Neural Networks*. 2015. Vol. 61. P. 85–117. https://doi.org/10.1016/j.neunet.2014.09.003.
12. Эйлер Л. Об упругих кривых. *Метод нахождения кривых линий, обладающих свойствами максимума, либо минимума или решение изопериметрической задачи, взятой в самом широком смысле* / ред. Н.С. Кошляков. М. – Л.: ГТТИ, 1934. С. 447–572. (Серия «Классики естествознания»).

Одержано 02.07.2021

Гулаєва Наталія Михайлівна,
кандидат фізико-математичних наук, доцент кафедри інформатики
факультету інформатики Національного університету «Києво-Могилянська Академія», Київ,
https://orcid.org/0000-0003-4588-0702
ngulayeva@yahoo.com

Шило Володимир Петрович,
доктор фізико-математичних наук, професор, провідний науковий співробітник
Інституту кібернетики імені В.М. Глушкова НАН України, Київ,
v.shylo@gmail.com

Глибовець Микола Миколайович,
доктор фізико-математичних наук, професор кафедри інформатики
факультету інформатики Національного університету «Києво-Могилянська Академія», Київ.
https://orcid.org/0000-0002-3853-2171
glib@ukma.edu.ua

MSC 65K05, 68W50

Nataliya Gulayeva [1]*, Volodymyr Shylo [2], Mykola Glybovets [1]

Genetic Algorithms as Computational Methods for Finite-Dimensional Optimization

[1] *National University of Kyiv-Mohyla Academy*
[2] *V.M. Glushkov Institute of Cybernetics of the NAS of Ukraine, Kyiv*
* *Correspondence: ngulayeva@yahoo.com*

Introduction. As early as 1744, the great Leonhard Euler noted that nothing at all took place in the universe in which some rule of maximum or minimum did not appear [12]. Great many today's scientific and engineering problems faced by humankind are of optimization nature. There exist many different methods developed to solve optimization problems, the number of these methods is estimated to be in the hundreds and continues to grow. A number of approaches to classify optimization methods based on various criteria (e.g. the type of optimization strategy or the type of solution obtained) are proposed, narrower classifications of methods solving specific types of optimization problems (e.g. combinatorial optimization problems or nonlinear programming problems) are also in use. Total number of known optimization method classes amounts to several hundreds. At the same time, methods falling into classes far from each other may often have many common properties and can be reduced to each other by rethinking certain characteristics. In view of the above, the pressing task of the modern science is to develop a general approach to classify optimization methods based on the disclosure of the involved search strategy basic principles, and to systematize existing optimization methods.

The purpose is to show that genetic algorithms, usually classified as metaheuristic, population-based, simulation, etc., are inherently the stochastic numerical methods of direct search.

Results. Alternative statements of optimization problem are given. An overview of existing classifications of optimization problems and basic methods to solve them is provided. The heart of optimization method classification into symbolic (analytical) and numerical ones is described. It is shown that a genetic algorithm scheme can be represented as a scheme of numerical method of direct search. A method to reduce a given optimization problem to a problem solvable by a genetic algorithm is described, and the class of problems that can be solved by genetic algorithms is outlined.

Conclusions. Taking into account the existence of a great number of methods solving optimization problems and approaches to classify them it is necessary to work out a unified approach for optimization method classification and systematization. Reducing the class of genetic algorithms to numerical methods of direct search is the first step in this direction.

Keywords: mathematical programming problem, unconstrained optimization problem, constrained optimization problem, multimodal optimization problem, numerical methods, genetic algorithms, metaheuristic algorithms.