УДК 004.94; 517.9:519.6

МЕТОДИ Й АЛГОРИТМИ РОЙОВОГО ІНТЕЛЕКТУ ДЛЯ ЗНАХОДЖЕННЯ РОЗВ'ЯЗКІВ ОПТИМІЗАЦІЙНИХ ЗАДАЧ МОДЕЛЮВАННЯ ПОВЕДІНКИ СКЛАДНИХ ОБ'ЄКТІВ І СИСТЕМ

В. С. Татенко[1], В. В. Хайдуров[1]

[1] Навчально-науковий Фізико-технічний інститут

Анотація

Сучасний ройовий інтелект має значний потенціал для застосування в енергетичній галузі через свою здатність до оптимізації та розв'язання складних проблем. У роботі розглянуті й модифіковані відомі ефективні методи й алгоритми ройового інтелекту (алгоритм оптимізації роєм частинок, бджолиний алгоритм оптимізації, метод диференціальної еволюції) для пошуку розв'язків багатовимірних екстремальних задач з обмеженнями і без обмежень. Продемонстрована ефективність роботи модифікованих методів на різних класичних і прикладних задачах, які використовуються в проєктуванні елементів складних об'єктів та їх систем. Проведено порівняльний аналіз результатів роботи даних методів.

Ключові слова: оптимізація, ройовий інтелект, математичне моделювання, складні об'єкти та системи

Вступ

З розвитком сучасних обчислювальних систем, її прикладного застосування для розв'язування оптимізаційних задач, виникає потреба у проєктуванні складних систем у теплоенергетиці, математичному моделюванні. Сучасна теорія оптимізаційних методів з урахуванням нових її застосувань зазнала суттєвих змін, які полягають в розробленні нових і модифікації існуючих методів й алгоритмів пошуку розв'язків оптимізаційних задач. Значний інтерес нині акцентований до алгоритмів і методів ройового інтелекту, які знаходять все більше прикладних застосувань в сучасній науці й техніці. Ройовий інтелект – це концепція, яка натхненна спостереженнями за поведінкою колоній тварин у природі, таких як мурашки, бджоли та рої птахів. Цей підхід до штучного інтелекту та оптимізації базується на моделюванні поведінки індивідуальних агентів, які взаємодіють між собою та з навколишнім середовищем. Колективна поведінка цих агентів активно використовується для вирішення складних завдань та пошуку оптимальних рішень без централізованого керування. Застосування ройового інтелекту в науці та техніці включає різні галузі, такі як оптимізація, робототехніка, мережі передачі даних і навіть машинне навчання. Наприклад, алгоритми оптимізації на основі ройового інтелекту можуть використовуватися для вирішення задач маршрутизації, пошуку оптимальних рішень у складних просторах параметрів або керування розподіленими системами. У галузі машинного навчання ройові методи й алгоритми можуть бути застосовані для створення ефективних методів глибокого навчання, зокрема, у задачах оптимізації функцій втрат та пошуку гіперпараметрів.

1. Постановка задачі

Більшість прикладних задач оптимізації зводяться до знаходження глобальних екстремумів функцій у класичній постановці, яка подається так [1]:

$$y = F(X) \to min\,(max), X \in G, G \subset R^n,$$

де G – простір пошуку значень аргументів X, n – розмірність простору пошуку глобального екстремуму функції. X – факторні (незалежні) змінні, які встановлюють причинно-наслідковий зв'язок з залежною (результуючою) змінною y. На функцію $F(X)$ часто накладаються функціональні обмеження, які подаються у вигляді таких функціональних залежностей:

$$g_i(X) \le S_i, i \in \overline{1;K},$$
$$g_i(X) = S_i, i \in \overline{K+1;R},$$

де R – загальна кількість функціональних обмежень.

Складність цільової функції $F(X)$ і відповідних обмежень $g_i(X)$ визначається конкретним технічним завданням. Від вигляду $F(X)$ і $g_i(X)$ залежить вибір оптимізаційного методу, який буде застосовуватись для пошуку $F\left(X^*\right)$. Слід зазначити, що залежність виглядає наступним чином:

$$y = F(X) \to min$$

Такі функціональні залежності виникають у регресійному (лінійному і нелінійному) аналізі, у задачах класифікації, кластеризації даних, у побудові

ефективних математичних моделей керування процесами, прогнозуванні часових рядів, модернізації об'єктів і систем, а також при розробленні складних енергетичних комплексів в умовах сьогодення. У роботі розглядатимуться завдання пошуку глобального мінімуму задач без функціональних обмежень на цільову функцію/функціонал, розглядаються задачі з обмеженнями у вигляді функцій, а також методи їх пошуку.

2. Методи ройового інтелекту для розв'язання задач багатовимірної глобальної оптимізації

2.1. Алгоритм оптимізації роєм частинок (*PSO*)

PSO використовує рій частинок, де кожна частинка представляє потенційне вирішення проблеми. Спочатку всі частинки рою займають випадкове положення у просторі пошуку розв'язку задачі та мають маленькі випадкові швидкості [2]. На заключних ітераціях множина частинок збігається до одного або кількох оптимумів, які є глобальними (якщо їх кілька, а не один). Поведінка частинки в гіперпросторі пошуку рішення постійно підлаштовується відповідно до свого досвіду та досвіду своїх сусідів. Крім цього, кожна частинка пам'ятає свою найкращу позицію з досягнутим локальним найкращим значенням цільової (фітнес) функції і знає найкращу позицію частинок своїх сусідів, де було досягнуто глобального на даний момент оптимуму функції. У процесі пошуку частинки рою обмінюються інформацією про досягнуті кращі результати та змінюють свої позиції та швидкості за певними правилами на основі наявної на даний момент інформації про локальні та глобальні досягнення. При цьому глобальний найкращий результат відомий усім частинкам і він коригується в тому випадку, коли деяка частинка рою знаходить кращу позицію з результатом, що перевершує поточний глобальний оптимум. Кожна частинка рою підпорядковується досить простим правилам поведінки, які враховують локальний успіх кожної особини та глобальний оптимум усіх особин (або деякої множини сусідів) рою.

Кожна i-а частинка характеризується ітерацією n, своєї позицією $x_i(n)$ у гіперпросторі та швидкістю руху $v_i(n)$. Швидкість i-ї частинки обчислюється як:

$$v_i(n+1) = v_i(n) + \alpha_1\left(x_i^{best}(n) - x_i(n)\right) r_1 + \\ +\alpha_2\left(x^*(n) - x_i(n)\right) r_2, \quad (1)$$

позиція i-ї частинки обчислюється у вигляді

$$x_i(n+1) = x_i(n) + v_i(n+1), \quad (2)$$

де

$x_i(n) = \left(x_{i1}(n), x_{i2}(n), \ldots, x_{iM}(n)\right)$ – позиція i-ї частинки на ітерації n;

$x_i^{best}(n) = \left(x_{i1}^{best}(n), x_{i2}^{best}(n), \ldots, x_{iM}^{best}(n)\right)$ – найкраща позиція і-ї частинки (персональна найкраща позиція);

$x^*(n) = \left(x_1^*(n), x_2^*(n), \ldots, x_M^*(n)\right)$ – найкраща позиція по всій популяції (глобальна краща позиція);

$v_i(n) = \left(v_{i1}(n), v_{i2}(n), \ldots, v_{iM}(n)\right)$ – вектор швидкості i-ї частинки на ітерації;

α_1, α_2 – додатні коефіцієнти прискорення, що регулюють внесок когнітивної та соціальної компонент;

$r_1 = \left(r_{11}, r_{12}, \ldots, r_{1M}\right), r_2 = \left(r_{21}, r_{22}, \ldots, r_{2M}\right)$ – вектори випадкових чисел, що вносять елемент випадковості у процес пошуку.

Перший доданок в (1) $v_i(n)$ зберігає попередній напрямок швидкості i-ї частин і може розглядатися як момент, який перешкоджає різкій зміні напрямку швидкості і виступає в ролі інерційної компоненти (*inertia velocity*). Когнітивна компонента (*cognitive velocity*) $\alpha_1\left(x_i^{best}(n) - x_i(n)\right) r_1$ визначає характеристики частинки щодо її передісторії, яка зберігає кращу позицію даної частинки. Ефект цього доданка в тому, що він намагається повернути частинку назад у кращу досягнуту позицію. Третій доданок $\alpha_2\left(x^*(n) - x_i(n)\right) r_2$ – визначає соціальну компоненту (*social velocity*), яка характеризує частинку щодо своїх сусідів. Ефект соціальної компоненти полягає в тому, що вона намагається спрямувати кожну частинку у бік досягнутого роєм (або його деяким найближчим оточенням) глобального оптимуму. Зміщення позиції частинки здійснюється на основі (2).

2.2. Бджолиний алгоритм глобального пошуку

Бджолиний алгоритм заснований на поведінці медоносних бджіл [3]. Він базується на поведінці бджіл, що розшукують їжу, та є розширенням системи бджіл. Виділяють фазу бджоли-працівника (зайнятого фуражиру) та бджоли розвідника. Метою алгоритму є визначення місцезнаходження хороших ділянок у просторі пошуку. Бджоли-розвідники виконують випадковий пошук. Знайдені добрі (за значенням цільової функції/функціоналу) ділянки досліджуються за допомогою локального пошуку. Рішенню відповідає позиція бджоли, що знаходиться на певній ділянці.

2.3. Метод диференціальної еволюції

Основною ідеєю методу диференціальної еволюції є комбінування мутації та схрещування для ефективного пошуку оптимальних рішень у багатовимірних просторах параметрів [4]. Метод використовує деякі ідеї генетичних алгоритмів, але, на відміну від останніх, не передбачає роботу з бінарним кодом.

Задача: $F(X) \to min, X = X_1, X_2, \ldots, X_M$.

Основні етапи методу є такими:

1. Для кожної хромосоми $\overline{x_i}$ в популяції,

$$\overline{x_i} = \left(\overline{x_{i1}}, \overline{x_{i2}}, \ldots, \overline{x_{iM}}\right)$$

(N – розмірність популяції) обираються три інші випадкові члени цієї популяції x_1, x_2, x_3,

$$\overline{x_i} \neq x_1, \overline{x_i} \neq x_2, \overline{x_i} \neq x_3,$$
$$x_1 \neq x_2 \neq x_3.$$

2. Генерується мутантний вектор:

$$v = x_1 + F\left(x_2 - x_3\right), F \in (0; 2).$$

3. Різниця векторів $x_2 - x_3$ масштабується за визначеним користувачем гіперпараметром F, $F \in (0; 2)$.
4. Формуємо пробний вектор на основі схрещування $\overline{x_i}$ з мутантним вектором v: для кожної координати хромосоми генерується випадкове число $r \in (0; 1)$ за нормальним розподілом. Якщо $r < P$, P – задана константа (ще один параметр алгоритму, $P \in (0; 1)$), то відповідна координата хромосоми замінюється на

$$v_i = \overline{x_{ij}}, j \in \overline{1; M},$$

M – розмірність простору пошуку (кількість незалежних змінних фітнес-функції).
5. Якщо $f\left(\overline{x_i}\right) > f(v)$, $\overline{x_i} = v$.

Усі вищенаведені кроки виконуються до критерію зупинки методу (класичні, як і в інших алгоритмах і методах ройового інтелекту).

2.4. Комбінація детермінованих і стохастичних методів/алгоритмів для знаходження розв'язків складних задач

У складних науково-технічних прикладних задачах цільова функція або функціонал може обчислюватись порівняно довго навіть на сучасних обчислювальних системах. Стохастичні методи й алгоритми глобальної оптимізації можуть знаходити глобальні екстремуми цільових функцій, не вимагаючи додаткової про них інформації, зокрема, диференційовність цільових функцій. Звісно ж, що кількість викликів процедури знаходження значення цільової функції у такому разі зростає. У такому разі зростає й загальна кількість обчислень. Це означає, що зростає процесорний час пошуку глобального екстремуму цільової функції. У такому разі розробляють методи й алгоритми, які передбачають детерміновану складову, а також і стохастичну складову. Детермінована складова працює швидко і ефективно уточнює знайдений розв'язок, а стохастична складова не дає методу/алгоритму застрягати в локальних екстремумах.

3. Практичні результати роботи

3.1. Модифікація методів/алгоритмів ройового інтелекту

В обчислювальній математиці кожна модифікація методу/алгоритму оптимізації передбачає роботу в кількох напрямках:

1. Зменшення загальної кількості ітерацій для досягнення певної похибки. Позитивний ефект зменшення загальної кількості ітерацій призводить до зменшення загального процесорного часу розв'язання конкретної задачі або вирішення конкретного завдання. Це в свою чергу зменшує загальне навантаження на систему.
2. Підвищення точності обчислень. Позитивний ефект від цього полягає у прискоренні збіжності методу/алгоритму оптимізації. Це ж у свою чергу знову призводить до зменшення кількості ітерацій, а значить – до зменшення процесорного часу, який необхідний для вирішення поставленої задачі або розв'язання конкретного завдання.

У даній роботі пропонується кілька модифікацій описаних вище алгоритмів.

1. *Модифікація положення найгіршого елемента популяції в описаних вище алгоритмах і методах ройового інтелекту.* Найгірший елемент популяції x_j^{worst} замінити на усереднене положення елемента по всій популяції x_j, $\forall j \in \overline{1; M}$ за такою формулою:

$$x_j^{Average} = \frac{1}{N}\sum_{i=1}^{N} x_{ij}, \forall j \in \overline{1; M}, \quad (3)$$

де N – загальна кількість елементів у популяції, а M – розмірність простору пошуку. Такий підхід можна виконувати на кожній ітерації або періодично – кожні K ітерацій.
2. *Модифікація положення елементів популяції до найкращого з кроком h.* Оновлення кожного елемента популяції виконується за формулою:

$$x_{ij} = x_{ij} + h \cdot \frac{x_{Best,j} - x_{ij}}{\|x_{Best,j} - x_{ij}\|}, \quad (4)$$

де x_{Best} – кращий вектор популяції за значенням цільової функції. Слід зазначити, що крок руху h можна визначати пропорційно (за лінійним законом) околу популяції, яка збігатиметься до глобального оптимуму з плином часу.
3. *Зміна гіперпараметрів методів/алгоритмів оптимізації.* Такий підхід передбачає зміну одного або кількох гіперпараметрів у процесі роботи програми. Наприклад, у алгоритмі диференціальної еволюції можна параметр F замінити на випадкове дійсне число у діапазоні від 0 до 2 через K ітерацій. У бджолиному алгоритмі присутня залежність зміни параметра $\eta(n)$ від кількості ітерацій n. У такому випадку для оптимізації функцій великих розмірностей виникає проблема, яка полягає у завчасному зменшенні цього параметра. Для того, щоб таке не відбулось до знаходження глобального екстремуму функції або функціоналу такому випадку передбачається

$$\eta(n) = \eta_{max}\alpha^{[n/K]}, \quad (5)$$

де $[s]$ – ціла частина від числа s, K – період, який визначає те, через скільки ітерацій буде оновлене значення параметра η.

У методі диференціальної еволюції пропонується брати гіперпараметр F випадковим числом від 0 до 2 кожні K ітерацій. Це обумовлено тим, що на кожній ітерації метод знаходить ближчі значення до глобального оптимуму функції або функціоналу.

Перевагою усіх класичних методів й алгоритмів ройового інтелекту є простота реалізації, а також можливість створювати нові інтелектуальні системи на основі вже відомих. Такі інтелектуальні системи працюватимуть швидше і точніше за вже існуючі. Здійснювати модифікацію методів й алгоритмів можна по-різному – не лише так, як описано вище.

3.2. Структура розробленого програмного забезпечення

Розроблено програмний комплекс для задач оптимізації, який включає в себе модуль головного вікна переходу до модулів розв'язання розглянутих у роботі задач оптимізації, 6 модулів (3 модулі для розв'язання задач звичайними методами/алгоритмами та 3 модулі з модифікаціями відповідних методів/ алгоритмів). Кожез модулів програми є автономним, тобто працює незалежно від інших модулів, які знаходять глобальні екстремуми різних прикладних задач.

3.3. Тестування методів й алгоритмів на відомих функціях з обмеженнями і без обмежень

1. *Багатовимірна функція Растригіна.* Першою тестовою функцією є функція Растригіна, яка має такий математичний вигляд [5]:

$$f(X) = An + \sum_{i=1}^{n} \left(x_i^2 - A \cdot \cos\left(2\pi x_i\right)\right), \qquad (6)$$

де $A = 10$, $x_i \in [-5,12; 5,12]$, $i = \overline{1;n}$. Глобальний мінімум в точці $X^* = \left(x_1^*, x_2^*, \ldots, x_n^*\right) = (0; 0; \ldots; 0)$. $f\left(X^*\right) = 0$.

2. *Функція Розенброка, обмежена кубічною кривою і прямою.*

$$\begin{gathered} f(x,y) = (1-x)^2 + \\ +100\left(y - x^2\right)^2 \to min, \\ (x-1)^3 - y + 1 < 0, x + y - 2 < 0 \\ x \in [-1,5; 1,5], y \in [-0,5; 2,5]. \end{gathered} \qquad (7)$$

Глобальний мінімум в точці $\left(x^*, y^*\right) = (1;1)$. $f\left(x^*, y^*\right) = 0$.

3. *Функція Розенброка, обмежена диском.*

$$\begin{gathered} f(x,y) = (1-x)^2 + \\ +100\left(y - x^2\right)^2 \to min, \\ x^2 + y^2 < 2, \\ x \in [-1,5; 1,5], y \in [-1,5; 1,5]. \end{gathered} \qquad (8)$$

Глобальний мінімум в точці $\left(x^*, y^*\right) = (1;1)$. $f\left(x^*, y^*\right) = 0$.

4. *Функція Мішри-Берда з функціональними обмеженнями.*

$$\begin{gathered} f(x,y) = e^{(1-\cos x)^2} \sin y + \\ +e^{(1-\sin y)^2} \cos x + (x-y)^2, \\ (x+5)^2 + (y+5)^2 < 25, \\ x \in [-10; 0], y \in [-6,5; 0]. \end{gathered} \qquad (9)$$

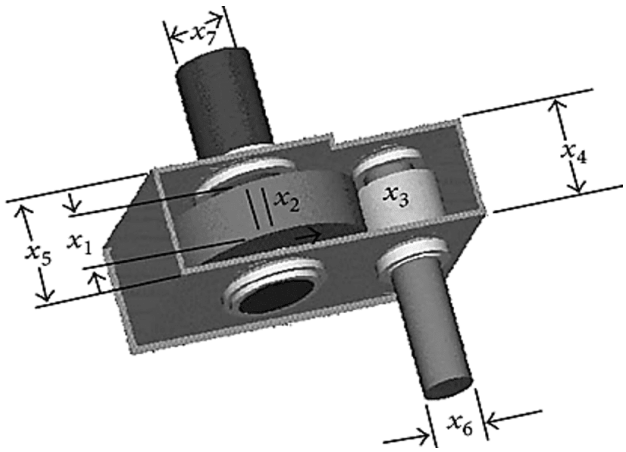

Рис. 1. Геометрія редуктора

Глобальний мінімум в точці $\left(x^*, y^*\right) = (-3,13025; -1,58214)$. $f\left(x^*, y^*\right) = -106,76454$.

Повні тестування описаних методів/алгоритмів та їх модифікацій, що описані у даній роботі, виконано на функціях (6)–(9) у магістерській дисертації першого автора роботи.

Математична модель мінімізації ваги редуктора. Як було зазначено на початку в актуальності роботи, ройовий інтелект активно застосовується для знаходження розв'язків різних прикладних задач. У даній роботі розглядається модель мінімізації ваги редуктора, геометрія якого подана на рис.1. Математичне формулювання оптимізаційної задачі з урахуванням обмежень на вигини зубців колеса, поверхневої напруги, поперечні відхилення валів і напруги у валах подається так [6, 7]:

$$\begin{gathered} f\left(\overline{x}\right) = 0.7854 x_1 x_2^2 \times \\ \times \left(3.3333 x_3^2 + 14.9334 x_3 - 43.0934\right) - \\ -1.508 x_2 \left(x_6^2 + x_7^2\right) + 7.4777 \left(x_6^3 + x_7^3\right) + \\ +0.7854 \left(x_4 x_6^2 + x_5 x_7^2\right) \to min, \\ 2,6 \le x_1 \le 3,6, 0,7 \le x_2 \le 0,8 \\ 17 \le x_3 \le 28, 7,3 \le x_4 \le 8,3 \\ 7,8 \le x_5 \le 8,3, 2,9 \le x_6 \le 3,9, 5 \le x_7 \le 5,5 \end{gathered} \qquad (10)$$

$$\begin{aligned} g_1\left(\overline{x}\right) &= \frac{27}{x_1 x_2^2 x_3} - 1 \le 0, \\ g_2\left(\overline{x}\right) &= \frac{397,5}{x_1 x_2^2 x_3^2} - 1 \le 0, \\ g_3\left(\overline{x}\right) &= \frac{1,93 x_4^3}{x_2 x_3^2 x_6^4} - 1 \le 0, \\ g_4\left(\overline{x}\right) &= \frac{1,93 x_5^3}{x_2 x_3 x_7^4} - 1 \le 0, \\ g_5\left(\overline{x}\right) &= \frac{1}{110 x_6^3} \sqrt{\left(\frac{745 x_4}{x_2 x_3}\right)^2 + \frac{16,9}{10^{-6}}} - 1 \le 0, \\ g_6\left(\overline{x}\right) &= \frac{1}{85 x_7^3} \sqrt{\left(\frac{745 x_5}{x 2 x 3}\right)^2 + \frac{157,5}{10^{-6}}} - 1 \le 0, \\ g_7\left(\overline{x}\right) &= \frac{x_2 x_3}{40} - 1 \le 0, \\ g_8\left(\overline{x}\right) &= \frac{5 x_2}{x_1} - 1 \le 0, \\ g_9\left(\overline{x}\right) &= \frac{x_1}{12 x_2} - 1 \le 0, \\ g_{10}\left(\overline{x}\right) &= \frac{1,5 x_6 + 1,9}{x_4} - 1 \le 0, \\ g_{11}\left(\overline{x}\right) &= \frac{1,1 x_7 + 1,9}{x_5} - 1 \le 0. \end{aligned} \qquad (11)$$

Таблиця 1. Порівняльний аналіз класичного методу диференціальної еволюції та його модифікації (для класичного методу $F = 1$, для модифікованого $F - rand\,(0; 1)$)

Розмір популяції	Простір	К-сть ітерацій (класичний алгоритм)	К-сть ітерацій (модифіковани алгоритм)	Результат цільової ф-ції (класичний алгоритм)	Результат цільової ф-ції (модифікований алгоритм)
$N = 150$	7	43	12	2.986214e+3	2.966142e+3
$N = 150$	7	34	21	2.986055e+3	2.995720e+3
$N = 150$	7	41	27	2.991924e+3	2.996293e+3

Таблиця 2. Значення кращого значення функції та відповідних аргументів, які знайдені модифікованим методом

x_1^*	x_2^*	x_3^*	x_4^*	x_5^*	x_6^*	x_7^*
3.524129	0.7	17	7.352752	7.903315	3.359240	5.291292
3.570157	0.7	17	7.9303545	8.3	3.419805	5.287778
3.510148	0.7	17	7.3	7.812450	3.362408	5.290009

У моделі (10)–(11) x_1 – ширина обшивки; x_2 – модуль колеса; x_3 – кількість зубів колеса (ціла змінна); x_4 – довжина першого валу між підшипниками; x_5 – довжина другого валу між підшипниками; x_6 – діаметр першого валу; x_7 – діаметр другого валу.

У таблиці 1 наведено кілька обчислювальних експериментів, які демонструють ефективність як класичного алгоритму глобальної багатовимірної оптимізації методом диференціальної еволюції, так і його модифікації. Таблиця 2 містить відповідні розв'язки задач для кожного з обчислювальних експериментів (екстремальна точка, яка містить 7 значень координат). Глобальний мінімум $f(x^*) = 2996.348165$.

Висновки

У даній роботі досліджені три актуальні методи й алгоритми ройового інтелекту: алгоритм глобальної оптимізації роєм частинок, бджолиний алгоритм пошуку глобальних розв'язків задач, які подані в екстремальних постановках, метод диференціальної еволюції. Отримано три модифікації розглянутих методів й алгоритмів. Перша модифікація полягає у заміні найгіршого елемента популяції на усереднене положення елемента по всій популяції. Такий підхід ефективно працює, якщо його застосовувати на кожній ітерації або періодично – кожні K ітерацій. Друга модифікація даних методів й алгоритмів полягає у застосуванні принципу детермінованого руху в напрямку до кращого елемента популяції. Крок руху h визначається пропорційно (за лінійним законом) околу популяції, яка збігатиметься до глобального оптимуму з плином часу. Третя ж модифікація полягає у зміні гіперпараметрів методів/алгоритмів оптимізації. Це передбачає зміну одного або кількох гіперпараметрів у процесі роботи програми. Практичні результати статті полягають у тому, що комплексне застосування трьох модифікацій для кожного методу/алгоритму дає переваги при зростанні розмірності пошуку в екстремальній задачі. Для багатовимірних задач модифікації впливають на елементи популяції і дають змогу точніше визначати розв'язок. Це зменшує загальну кількість ітерацій методу/алгоритму, а значить, зменшує час пошуку оптимуму поставленої задачі з заданою точністю.

Перелік використаних джерел

1. *Joaquim R.* Engineering Design Optimization. — 2020. — URL: https://flowlab.groups.et.byu.net/mdobook.pdf.
2. *Martínez F. J., Gonzalo García E.* The Generalized PSO: A New Door to PSO Evolution. — 2008. — С. 1—15. — URL: https://doi.org/10.1155/2008/861275.
3. *Verma B. K., Kumar D.* A review on Artificial Bee Colony algorithm. — 2013. — URL: https://doi.org/10.14419/ijet.v2i3.1030.
4. *Dey N.* Enhancing Differential Evolution Algorithm for Solving Continuous Optimization Problems. — 2013. — С. 1—7. — URL: http://urn.kb.se/resolve?urn=urn:nbn:se:mdh:diva-33466.
5. *Wikipedia.* Multidimensional functions for testing global optimization methods and algorithms. — 2024. — [Online; accessed 25-Match-2024]. https://en.wikipedia.org/wiki/Test_functions_for_optimization/.
6. *Ming-Hua L.* Design Optimization of a Speed Reducer Using Deterministic. Mathematical Problems in Engineering. — 2013.
7. *Cagnina L. C., Esquivel S. C., Coello C. A. C.* Solving engineering optimization problems with the simple constrained particle swarm optimizer. — 2008. — С. 319—326. — URL: https://doi.org/10.1007/978-981-15-0306-1.