УДК 519.6:004.8

PACS 02.60.-x, 02.60.Pn, 02.70.Wz

ТАЛІМОНЧИК Анна Сергіївна
Черкаський національний університет
імені Богдана Хмельницького, студентка
e-mail: anna.talimonchyk@gmail.com

КРАСНОШЛИК Наталія Олександрівна
Черкаський національний університет
імені Богдана Хмельницького, к. т. н.,
доцент, доцент кафедри прикладної
математики та інформатики
e-mail: wlik007@ukr.net

РОЗВ'ЯЗУВАННЯ ПРИКЛАДНИХ ЗАДАЧ УМОВНОЇ ОПТИМІЗАЦІЇ ЗА ДОПОМОГОЮ МЕТОДІВ РОЙОВОГО ІНТЕЛЕКТУ

Анотація. *У роботі розглянуто метод бактеріальної оптимізації і метод оптимізації зграєю вовків, які відносяться до методів ройового інтелекту, а також метод диференціальної еволюції. Дані методи відносяться до біоінспірованих алгоритмів оптимізації, які інтенсивно розвиваються в останні роки. Метою даної роботи є застосування методів ройового інтелекту та методу диференціальної еволюції до розв'язування задач умовної оптимізації інженерного спрямування. Наведено постановку прикладних задач оптимізації конструкції редуктора і конструкції натяжної/компресійної пружини. Для їх чисельного розв'язування використано метод штрафних функцій, який полягає у переході до задачі безумовної оптимізації. Проведено порівняння наближених розв'язків, отриманих розглянутими методами ройового інтелекту і методом диференціальної еволюції, та розв'язків, отриманих іншими авторами. Встановлено ефективність застосування зазначених методів до задач такого типу, оскільки вони не накладають обмежень до вигляду цільової функції, що важливо при використанні штрафних функцій, і зажди знаходять наближений глобальний мінімум.*

Ключові слова: *методи ройового інтелекту, метод бактеріальної оптимізації, метод оптимізації зграєю вовків, метод диференціальної еволюції, задача умовної оптимізації.*

Вступ

Розв'язування багатьох актуальних проблем в різних прикладних та фундаментальних науках зводиться до задач глобальної оптимізації. Чисельні методи оптимізації можна умовно розділити на детерміністичні і стохастичні. Детерміністичні методи, як правило, вимагають знаходження градієнта цільової функції та залежать від вибору початкових значень. Однак цільова функція у задачах глобальної оптимізації може бути недиференційовною, розривною, мультимодальною, або не мати аналітичного опису, тим самим ускладнюючи розв'язування задачі. Для ефективного розв'язування подібних задач використовують стохастичні методи, які не накладають додаткових обмежень на постановку задачі [1-2].

Стохастичні пошукові алгоритми почали інтенсивно розвиватися у 1980-х рр. [3]. Серед таких алгоритмів важливе місце посідають алгоритми, натхненні природою. Ще у середині минулого століття Д. Холланд уперше запропонував використовувати механізм еволюції органічного світу для оптимізації технічних систем. Пізніше з'явилися інші алгоритми, засновані на ідеї «природних обчислень». З'явився напрямок – ройовий інтелект. До нього віднесли такі поняття, як «мурашиний» та «бджолиний» рій, а також алгоритм рою частинок [4].

Ройовий інтелект (англ. Swarm intelligence) описує комплексну колективну поведінку децентралізованої системи, що самоорганізовується. Системи колективного інтелекту, як правило, складаються із множини агентів (багатоагентна система), що локально взаємодіють між собою та з навколишнім середовищем. Самі агенти зазвичай досить прості, але всі разом, локально взаємодіючи, створюють так званий колективний інтелект [5]. Ідеї їх поведінки виходять від природи, а особливо, від біологічних систем. Кожен агент слідує дуже простим правилам і, незважаючи на те, що немає якоїсь централізованої системи управління, поведінки, яка б вказувала кожному з них на те, що йому слід робити, локальні і, в деякій мірі, випадкові взаємодії призводять до виникнення інтелектуальної глобальної поведінки, неконтрольованого окремими агентами.

Алгоритми, засновані на методах ройового інтелекту належать до більш широкого класу біоінспірованих алгоритмів. Біоінспіровані алгоритми становлять більшість усіх алгоритмів, натхненний природою. Таким чином, можна сказати, що алгоритми, засновані на методах ройового інтелекту є підмножиною біоінспірованих алгоритмів, в той час як біоінспіровані алгоритми є підмножиною алгоритмів, натхненних природою.

Деякі біоінспіровані алгоритми не використовують ройовий інтелект. Наприклад, генетичні алгоритми – біоінспіровані, але не на основі ройового інтелекту. Проте не так легко класифікувати певні алгоритми, такі як метод диференціальної еволюції. Метод диференціальної еволюції немає прямого зв'язку з будь-якою біологічною поведінкою, але він має деяку схожість із генетичними алгоритмами, а також має ключове слово «еволюція», тому його можемо віднести до категорії біоінспірірованних алгоритмів.

Об'єктом дослідження у роботі виступать біоінспіровані алгоритми оптимізації.

Метою даної роботи є застосування методів ройового інтелекту та методу диференціальної еволюції до розв'язування задач умовної оптимізації інженерного спрямування.

1. Опис методів ройового інтелекту

Нехай задана цільова функція $f(x) = f(x_1, x_2, \ldots, x_n)$ на множині допустимих значень $D \subseteq x^n$. Необхідно знайти умовний глобальний мінімум функції $f(x)$ на множині D, тобто знайти таку точку $x^* \in D$, що:

$$f(x^*) = \min_{x \in D} f(x). \quad (1)$$

де $x = (x_1, x_2, ..., x_n)^T$, $D = \{x \mid x_i \in [a_i, b_i], i \in \overline{1,n}\}$.

Задача пошуку максимуму функції $f(x)$ зводиться до задачі пошуку мінімуму:

$$f(x^*) = \max_{x \in D} f(x) = -\min_{x \in D}(-f(x)).$$

Метод бактеріальної оптимізації / метод оптимізації переміщенням бактерій (bacterial foraging optimization algorithm) заснований на моделюванні поведінки одного виду бактерій – E. Coli, яка є однією з найбільш вивчених. Даний метод був запропонований Ліу (Liu Y.) і Пассіні (Passino K.M.) у 2002 р., і моделює хемотаксичну поведінку бактерій, які рухаються під впливом поживних речовин, присутніх у навколишньому середовищі.

Використовуючи механізми пересування (такий як джгутики) бактерії можуть переміщуватися у навколишньому середовищі, як хаотично (перекидатися і обертатися), так і спрямовано (плавно). Залежно від міжбактеріальної взаємодії, бактерії можуть роїтися навколо джерела їжі (тобто притягатися), і/або можуть боротися одна з одною (тобто відштовхуватися).

Алгоритм використовує наступні три процеси:

1) хемотаксис – бактерії рухаються під впливом поживних речовин і міжбактеріальної взаємодії;

2) репродукція – лише ті бактерії, які були ефективні протягом їх життя, можуть потрапити у наступне покоління;

3) видалення-розкид – із малою ймовірністю деякі бактерії замінюються бактеріями, які згенеровані випадковим чином.

Життєздатність бактерій, що використовується в алгоритмі означає, як багато бактерія може спожити поживних речовин протягом життя і наскільки ефективно вона може боротися з шкідливими речовинами.

Алгоритм методу бактеріальної оптимізації можна записати так [6]:

1. Ініціалізація
   1.1 Задаємо параметри притягування $\alpha_{attr}, \beta_{attr}$ та відштовхування α_{repel}, β_{repel}; параметр δ для генерації нової позиції бактерії, ймовірність видалення – розкиду p_{em}, при чому $0 < \alpha_{attr} < 1, 0 < \beta_{attr} < 1,\ 0 < \alpha_{repel} \left\langle 1, \beta_{repel} \right\rangle 0, 0 < \delta < 1$.

   При малих значеннях β_{attr} і великих значеннях β_{repel} популяція схильна до великого розсіювання у просторі пошуку, а при великих значеннях β_{attr} і малих β_{repel} популяція схильна до групування у невеликих областях пошуку.

   1.2 Задаємо максимальне число ітерацій видалення-розкиду N_{ed}, репродукції N_{re}, хемотаксису N_c, рівномірного руху N_{swim}, розміру популяції K, довжину вектору позиції бактерії M, мінімальних і максимальних значень для вектору позиції $x_j^{min}, x_j^{max}, j \in \overline{1, M}$.

   1.3 Задаємо цільову функцію: $f(x) = f_1(x) + f_2(x) \to min$, де $f_1(x)$ – фактична цільова функція, яка відображає кількість спожитих поживних речовин, $f_2(x)$ – додаткова цільова функція, яка відображає взаємодію бактерії з іншими бактеріями,

$$f_2(x) = \sum_{i=1}^{K} J(x, x_j) = -\alpha_{attr} \sum_{i=1}^{K} \exp\left(-\beta_{attr} \sum_{j=1}^{M} \left(x_j - x_{ij} \right)^2 \right) + \alpha_{repel} \sum_{i=1}^{K} \exp\left(-\beta_{repel} \sum_{j=1}^{M} \left(x_j - x_{ij} \right)^2 \right),$$

   де x – вектор позиції бактерії.

   1.4 Створення випадковим чином вектору кращої позиції:

   $x^* = \left(x_1^*, \ldots, x_M^* \right),\ x_j^* = x_j^{min} + \left(x_j^{max} - x_j^{min} \right) \cdot rand()$.

   1.5 Створюємо вихідну популяцію P:

   1.5.1 Номер бактерії $k = 1, P = \%$.

   1.5.2 Випадковим чином створюємо вектор позиції x_k:

   $$x_k = \left(x_{k1}, \ldots, x_{kM} \right),$$

   $$x_{kj} = x_j^{min} + \left(x_j^{max} - x_j^{min} \right) \cdot rand().$$

   1.5.3 Якщо $x_k \notin P$, то $P = P \bigcup \{ x_k \}, k = k + 1$.

   1.5.4 Якщо $k \le K$, то перейдемо на крок 1.5.2.

2. Номер кроку видалення-розкиду $n_{ed} = 1$.

3. Номер кроку репродукції $n_{re}=1$.
4. Хемотаксис:
   4.1 Номер кроку хемотаксису $n_c=1$.
   4.2 Номер бактерії $k=1$.
   4.3 $s_{k,n_c}=f\left(x_k\right)$.
   4.4 Номер кроку рівномірного руху $n_{swim}=1$.
   4.5 Хаотичний рух (генерація вектору позиції x^{new} від вектору x_k):
   4.5.1 $x_j^{new}=x_{kj}+\delta\cdot\left(x_j^{max}-x_j^{min}\right)\left(-1+2\cdot rand()\right), j\in\overline{1,M}$.
   $x_j^{new}=\max\left\{x_j^{min},x_j^{new}\right\}, x_j^{new}=\min\left\{x_j^{max},x_j^{new}\right\}, j\in\overline{1,M}$
   4.6 Якщо $f\left(x^{new}\right)<f\left(x_k\right)$, то $x_k=x^{new}$, перехід на крок 4.8.
   4.7 Якщо $n_{swim}<N_{swim}$, то $n_{swim}=n_{swim}+1$, перехід на крок 4.5.
   4.8 Якщо $k<K$, то $k=k+1$, перехід на крок 4.3.
   4.9 Якщо $n_c<N_c$, то $n_c=n_c+1$, перехід на крок 4.2.
5. Визначити найкращу бактерію за значенням цільової функції $x^*=\underset{k}{\operatorname{argmin}} f\left(x_k\right)$.
6. Якщо $f\left(x_{k^*}\right)<f\left(x^*\right)$, то $x^*=x_{k^*}$.
7. Репродукція:
   7.1 Обчислити життєздатність бактерій:
   $S_k=\sum_{n_c=1}^{N_c}s_{k,n_c}$, $k\in\overline{1,K}$.
   7.2 Упорядкувати P за життєздатністю, тобто $S_k<S_{k+1}$.
   7.3 Бактерії з найбільшою життєздатністю помирають, а кожна з тих, які залишились ділиться на дві бактерії, тобто $x_{\frac{K}{2}+k}=x_k, k\in\overline{1,K/2}$.
8. Якщо $n_{re}<N_{re}$, то $n_{re}=n_{re}+1$, перехід на крок 4.
9. Видалення-розкид:
   9.1 Номер бактерії $k=1$.
   9.2 Якщо $rand()\le p_{ed}$, то бактерія ініціалізується, тобто
   $x_{kj}=x_j^{min}+\left(x_j^{max}-x_j^{min}\right)\cdot rand(), j\in\overline{1,M}$.
   9.3 Якщо $k<K$, то $k=k+1$, перехід на крок 9.2.
10. Якщо $n_{ed}<N_{ed}$, то $n_{ed}=n_{ed}+1$, перехід на крок 3.

Розв'язком є x^*.

Метод оптимізації зграєю вовків (wolf pack search) запропонований у 2007 р., його ідея запозичена із соціальної поведінки зграї вовків. Для вовків типовий сімейний спосіб життя: вони живуть зграями – сімейними групами, що складаються з пари «вожаків», їх родичів, а також сторонніх одиноких вовків. Всередині зграї спостерігається строго окреслена ієрархія, на вершині якої знаходиться «вожак» зграї, направляючий інших особин на пошук жертви. Вовки «досліджують» місцевість на наявність здобичі, коли хтось із них почує запах жертви, починається її пошук. Чим сильніше відчувається запах, тим ближче вовки до жертви. Таким чином, вони

переміщуються у напрямку посилення запаху жертви. Причому, вовки поділяються на невеликі групи, і кожна група здійснює пошук в якомусь певному напрямку, відмінному від напрямів інших груп. У підсумку, коли один із вовків знайде жертву, він подає сигнал «вожаку» та іншим, щоб поділитися здобиччю з вовками зі зграї [2, 7]. Метод оптимізації зграєю вовків імітує процес їх полювання.

Припустимо, що місцевість, де полюють вовки – це пошукова область у задачі оптимізації, а частинки – це вовки. Нехай спочатку згенеровано N вовків в евклідовому просторі розмірністю d, тобто позицію кожного вовка представлено у виді вектора $x_i = (x_1,...,x_d)$, який визначає його координати в просторі.

Таким чином, зграя (популяція) являє собою множину потенційних розв'язків, координати яких, оновлюються на кожній ітерації, поки не знайдеться оптимальний розв'язок. Цільова функція $f(x)$ характеризує настільки сильно відчувається запах жертви, координати жертви і є шуканою оптимальною точкою. «Вовки» виконують пошук оптимальної точки об'єднуючись у групи, які рухаються в різних напрямках і обмінюються між собою інформацією.

Алгоритм пошуку можна охарактеризувати за допомогою трьох правил.

1. «Вовк» із найкращим значенням цільової функції на даному кроці – вожак. Якщо на наступній ітерації знайдеться інша особина з кращим значенням цільової функції, ніж у вожака, то зграя «обирає» нового вожака відповідно.

2. Решта вовків вивчають місцевість на наявність жертви, $f(x_i)$ характеризує, як сильно відчувається запах жертви i-м вовком. Тоді величина $GBest$ характеризує як сильно відчувається запах жертви вожаком зграї.

3. Вожак зграї «повідомляє» решті «вовків» у зграї про своє місцезнаходження, як найближчу відстань до жертви, щоб ті переміщалися в напрямку до нього. На цьому етапі вожака розглядають, як жертву, тобто ціль до якої варто наблизитися. Тоді вовки зграї переміщуються в напрямку вожака з кроком $step$, що спочатку визначено, причому d-а координата i-го вовка на $k+1$- й ітерації обраховується за формулою:

$$x_{id}^{k+1} = x_{id}^{k} + step \cdot \frac{(Gbest_d^k - x_{id}^k)}{\left\| Gbest_d^k - x_{id}^k \right\|}, \qquad (2)$$

де $Gbest_d^k$ – d-та координата вожака, визначеного за k попередніх ітерацій, $\|..\|$ – норма для простору пошуку.

Диференціальна еволюція (*Differential Evolution*) – алгоритм багатовимірної оптимізації, який відноситься до класу стохастичних методів оптимізації і використовує деякі ідеї генетичних алгоритмів.

Метод диференціальної еволюції призначений для знаходження глобального екстремуму недиференційованих, нелінійних, мультимодальних функцій від багатьох змінних. Метод був запропонований Р. Сторном і К. Прайсом, а вперше опублікований ними у 1995 р. [3].

В початковому вигляді, метод можна описати наступним чином. Спочатку генерується деяка множина векторів, які називають поколінням. Під векторами розуміють точки n-вимірного простору, в яких визначена цільова функція, яку потрібно мінімізувати. На кожній ітерації методу генерується нове покоління векторів, випадковим чином комбінуючи вектори із попереднього покоління. Число векторів у кожному поколінні однакове і є одним із параметрів методу.

Нове покоління векторів генерується наступним чином. Для кожного вектора x_i зі старого покоління обирається три різних випадкових вектори v_1, v_2, v_3 за виключення самого вектора x_i, і генерують так званий мутантний вектор v.

Над мутантним вектором виконується операція «схрещення» (crossover), яка полягає в тому, що деякі його координати замінюються відповідними координатами вихідного вектору x_i (кожна координата замінюється з деякою ймовірністю, яка також є параметром методу). Вектор, отриманий після схрещення, називається пробним вектором. Якщо він виявився кращим, ніж вектор x_i (тобто значення цільової функції стало меншим), то в новому поколінні вектор x_i замінюється пробним вектором, інакше – залишається x_i [3, 6].

Алгоритм методу диференціальної еволюції можна записати так:

Крок 1. Ініціалізація

1.1 Задамо параметр кросинговера p^c, параметри δ і α для генерації хромосоми, причому $\delta \in [0;1], \alpha \in [0;1]$.

1.2 Задамо максимальну кількість ітерацій N, розмірності популяції K, довжини хромосоми M, максимальне число різниць L, мінімальне і максимальне значення для хромосоми $x_j^{min}, x_j^{max}, j \in \overline{1, M}$.

1.3 Задаємо цільову функцію, де x – хромосома.

1.4 Випадковим чином створюємо кращу хромосому $x^* = \left(x_1^*, \ldots, x_M^*\right)$, $x_j^* = x_j^{min} + \left(x_j^{max} - x_j^{min}\right) \cdot rand()$, де $rand()$ – функція, що повертає рівномірно розподілене випадкове число в діапазоні $[0;1]$.

1.5 Створюємо вихідну популяцію P:

1.5.1 Номер хромосоми $k = 1, P = \%$.

1.5.2 Випадковим чином створюємо хромосоми x_k, $x_k = \left(x_{k1}, \ldots, x_{kM}\right)$,
$x_{kj} = x_j^{min} + \left(x_j^{max} - x_j^{min}\right) \cdot rand()$.

1.5.3 Якщо $x_k \notin P$, то $P = P \bigcup \{x_k\}, k = k + 1$.

1.5.4 Якщо $k \le K$, то перейдемо на крок 1.5.2.

1.6 Визначаємо кращу хромосому за цільовою функцією $k^* = \arg\min f\left(x_k\right)$.

Крок 2. Номер ітерації $n = 1$.

Крок 3. $k = 1$.

Крок 4. $x^{cur} = x_k$.

Крок 5. Виконаємо арифметичний кросинговер:

5.1 двоє батьків обираємо випадковим чином
$i1 = round\left(1 + \left(K - 1\right) \cdot rand()\right)$, $i2 = round\left(1 + \left(K - 1\right) \cdot rand()\right)$,
при чому $i1 \ne i2$, де $round()$ – функція, яка округлює число до найближчого цілого.

5.2 виконуємо схрещення двох батьків, тобто для батьків $i1$ та $i2$, нащадок буде представлений $(x_{ij};\sigma_{ij})$:

$x_{ij}=a_k x_{i1k}+(1-x_k)\cdot x_{i2k}$, $\sigma_{ij}=a_k\sigma_{i1k}+(1-\sigma_k)\cdot\sigma_{i2k}$,

де a_k – випадкове число, $a_k\in[0,1]$.

5.3 $x_j^{cur}=\max\left\{x_j^{min},x_j^{cur}\right\}, x_j^{cur}=\min\left\{x_j^{max},x_j^{cur}\right\}$.

Крок 6. Якщо $f(x_{k^*})<f(x^*)$, то $x^*=x_{k^*}$

Крок 7. Якщо $n<N$, то $n=n+1$, перейти на крок 3

Розв'язком є x^*.

2. Постановка прикладних задач умовної оптимізації

Задача оптимізації конструкції редуктора

Задача оптимізації конструкції редуктора полягає у мінімізації ваги редуктора (рис. 1) з урахуванням обмежень на вигини зубців колеса, поверхневої напруги, поперечні відхилення валів і напруги у валах.

Наведемо основні конструктивні параметри редуктора [8]:

- b – ширина обшивки (x_1);
- m – модуль колеса (x_2);
- z – число зубів колеса (x_3 – ціла змінна);
- l_1 – довжини першого валу між підшипниками (x_5);
- l_2 – довжини другого валу між підшипниками (x_4);
- d_1 – діаметр першого валу (x_6);
- d_2 – діаметр другого валу (x_7).

Введемо наступні позначення:

- σ_g – фактична напруга вигину зубів шестерні;
- p_s – фактична поверхнева стискувальна напруга;
- B – коефіцієнт пружності (залежить від модуля пружності) ;
- f_1, f_2 – відхилення для валів 1 і 2;
- P – передана сила;
- M_g, M_s – моменти згинання та обертання для валів 1 і 2 $M_z=\sqrt{M_g^2+0.75M_s^2}$;
- W_{x_1}, W_{x_2} – сила опору перерізу.

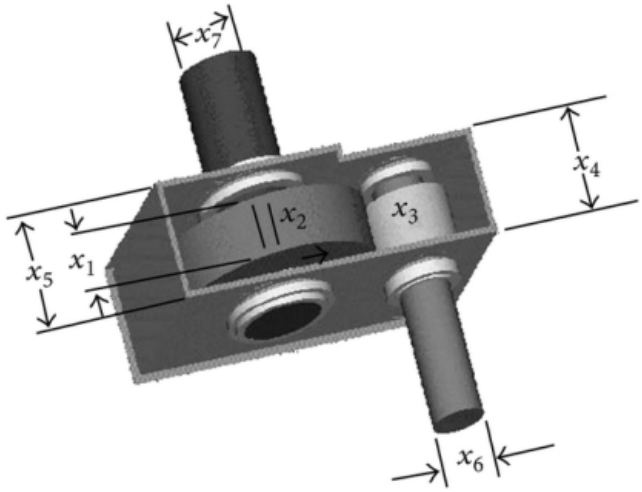

Рис. 1 Конструкція редуктора [8]

При цьому повинні виконуватись наступні обмеження:

- умова вигину: $\sigma_g = \frac{2M_g}{bm^2z} \le k_g$.
- обмеження напруги при стисненні: $p_{s^2} = \frac{2BM}{m^2z^2b} \le p_{a^2}$.
- мінімальна кількість зубів: $z \ge 17$.
- відносні умови ширини обшивки: $5 \le \frac{b}{m} \le 12$.
- умова розміру : $m\left(z_1 + z_2\right) \le 160$.
- поперечне відхилення вала 1 через навантаження: $f_1 = \frac{1}{48} \cdot \frac{Pl_1^3}{El_1} \le f_{01} = 0.001$.
- поперечне відхилення вала 2 через навантаження: $f_2 = \frac{1}{48} \cdot \frac{Pl_2^3}{El_2} \le f_{02} = 0.001$.
- запасний стан напруги для валу 1: $\sigma_{g1} = \frac{M_{z_1}}{w_{x1}} \le k_{g1}$.
- запасний стан напруги для валу 2: $\sigma_{g2} = \frac{M_{z_2}}{w_{x2}} \le k_{g1}$.
- стан продуктивності: $d_2 \ge 5$.
- умови конструкції: $1.5d_1 + 1.9 \le l_1$, $1.1d_2 + 1.9 \le l_2$.

В результаті отримаємо задачу нелінійної оптимізації:

$$f\left(x\right) = 0.7854x_1x_2^2\left(3.3333x_3^2 + 14.9334x_3 - 43.0934\right) - 1.508x_1\left(x_6^2 + x_7^2\right) + \\ +7.4777\left(x_6^3 + x_7^3\right) + 0.7854\left(x_4x_6^2 + x_5x_7^2\right) \to min \qquad (2)$$

$$g_1(x)=\frac{27}{x_1x_2^2x_3}-1\le 0,$$

$$g_2(x)=\frac{397.5}{x_1x_2^2x_3^2}-1\le 0,$$

$$g_3(x)=\frac{1.93x_4^3}{x_2x_3x_6^4}-1\le 0,$$

$$g_4(x)=\frac{1.93}{x_2x_3x_7^4}-1\le 0,$$

$$g_5(x)=\frac{1.0}{110x_6^3}\sqrt{\left(\frac{745.0x_4}{x_2x_3}\right)^2+16.9\times10^6}-1\le 0,$$

$$g_6(x)=\frac{1.0}{85x_7^3}\sqrt{\left(\frac{745.0x_5}{x_2x_3}\right)^2+157.5\times10^6}-1\le 0,$$

$$g_7(x)=\frac{x_2x_3}{40}-1\le 0,$$

$$g_8(x)=\frac{5x_2}{x_1}-1\le 0,$$

$$g_9(x)=\frac{x_1}{12x_2}-1\le 0,$$

$$g_{10}(x)=\frac{1.5x_6+1.9}{x_4}-1\le 0,$$

$$g_{11}(x)=\frac{1.1x_7+1.9}{x_5}-1\le 0,$$

$$2.6\le x_1\le 3.6\ ,0.7\le x_2\le 0.8,$$

$$17\le x_3\le 28,7.3\le x_4\le 8.3,$$

$$7.8\le x_5\le 8.3,2.9\le x_6\le 3.9,5.0\le x_7\le 5.5\ .$$

Задача оптимізації конструкції натяжної/компресійної пружини

Задача полягає у мінімізації ваги натяжної/компресійної пружини, конструкцію якої наведено на рис. 2, з урахуванням обмежень мінімального відхилення, напруги зсуву, частоти перенапруги та обмежень зовнішнього діаметра і конструктивних змінних.

Перерахуємо основні конструктивні параметри:

- d – діаметр дроту (x_1);
- D – середній діаметр котушки (x_2);
- N – кількість активних котушок (x_3).

Наведемо математичну постановку цієї задачі [9]:

$$f(x)=(x_3+2)x_2x_1^2\to min, \tag{3}$$

Повинні виконуватись наступні обмеження:

- відхилення: $g_1(x) = 1 - \frac{x_2^3 x_3}{7.178 x_1^4} \le 0,$
- напруга стиснення: $g_2(x) = \frac{4x_2^2 - x_1 x_2}{12.566\left(x_2 x_1^3\right) - x_1^4} + \frac{1}{5.108 x_1^2} - 1 \le 0,$
- частота: $g_3(x) = 1 - \frac{140.45 x_1}{x_2^2 x_3} \le 0,$
- зовнішній діаметр: $g_4(x) = \frac{x_2 + x_1}{1.5} - 1 \le 0.$

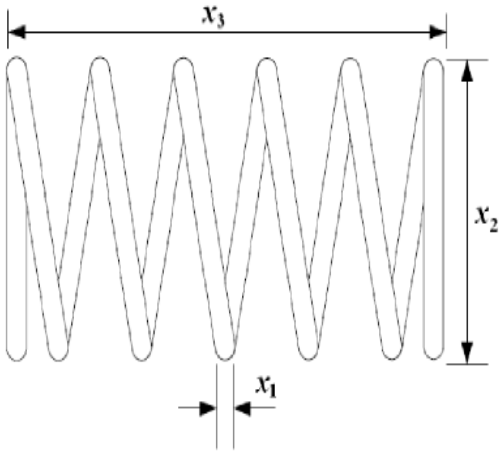

Рис. 2 Конструкція натяжної/компресійної пружини [9]

Нижня та верхня межі проектних змінних вибираються таким чином:

$$0.005 \le x_1 \le 2.0, 0.25 \le x_2 \le 1.3, 2.0 \le x_3 \le 15.0.$$

3. Застосування розглянутих методів ройового інтелекту до розв'язування задач умовної оптимізації

Для розв'язання наведених прикладних задач (2)-(3) використаємо метод штрафних функцій, тобто переходимо до еквівалентної задачі безумовної мінімізації:

$$F(x) = f(x) + \delta(x|X) \to min, \quad x \in R^n, \tag{4}$$

де $\delta(x|X) = \begin{cases} 0, якщо\ x \in X, \\ +\infty, якщо\ x \notin X. \end{cases}$ – індикаторна функція множини X, а множина X визначається обмеженнями нерівностями і/або рівностями.

Розглянуті методи ройового інтелекту, а саме: метод бактеріальної оптимізації (BFAO), метод оптимізації зграєю вовків (WPS), а також метод диференціальної еволюції (DE) були реалізовані у середовищі MATLAB R2014b. Проведемо порівняльний аналіз даних методів при розв'язуванні прикладних задач умовної оптимізації. При цьому кожен метод застосували 10 разів, та обирали кращий результат серед отриманих.

Результати обчислювальних експериментів для задачі оптимізації конструкції редуктора, а також розв'язки, отримані іншими авторами наведено у табл. 1.

Таблиця 1

Метод	Змінні конструкції							$f(x)$
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	
WPS	3.15750	0.709276	17.000000	7.442233	7.963778	3.350354	5.286743	2.9048e+03
DE	3.23800	0.700381	17.00000	7.307671	7.803623	3.350870	5.286763	2.8954e+03
BFAO	3.84610	0.665184	15.895074	7.700917	8.221207	3.430979	5.311854	2.8295e+03
Akhtar et al. [9]	3.50612	0.700006	17.000000	7.549126	7.859330	3.365576	5.289773	3.0081e+03
Huang [9]	3.49565	0.700000	17.000000	7.300000	7.712038	3.343372	5.285352	2.9901e+03
Lu and Kim [9]	3.50000	0.700000	17.00000	7.30000	7.670396	3.542421	5.245814	3.0195e+03

Результати обчислювальних експериментів для задачі оптимізації конструкції натяжної/компресійної пружини, а також розв'язки, отримані іншими авторами за допомогою методу рою часток, генетичного алгоритму наведено у табл. 2.

Таблиця 2

Метод	Змінні конструкції			$f(x)$
	x_1	x_2	x_3	
WPS	0.051350587825182	0.406569118257710	7.430889946267433	0.010110622691547
DE	0.050000000000000	0.374432870716765	8.546569316461159	0.009872455563440
BFAO	0.062236761287400	0.649865776615723	9.640590240821823	0.029301687855993
PSO (Ha and Wang) [10]	0.051728	0.357644	11.244543	0.0126747
GA (Coello) [10]	0.051480	0.351661	11.632201	0.0127048
Constraint correction (Arora) [10]	0.050000	0.315900	14.250000	0.0128334

Отримані результати показали, що при розв'язуванні задачі оптимізації конструкції редуктора кращий результат отримано методом бактеріального пошуку, а при розв'язуванні задачі оптимізації конструкції натяжної/компресійної пружини – методом диференціальної еволюції. Також варто відмітити, що методом диференціальної еволюції знайдено другий за значенням кращий результат і для попередньої задачі.

Можна зробити висновок, що методи ройового інтелекту можуть ефективно використовуватись для розв'язування задач умовної оптимізації, зокрема і прикладних задач інженерного спрямування. При цьому більш ефективним виявився метод диференціальної еволюції, який відноситься до еволюйних алгоритмів. Отже, розглянуті методи продемонстрували свою ефективність й конкурентоздатність при розв'язуванні складних оптимізаційних задач, і передбачають подальший розвиток, зокрема у напрямку еволюційних стратегій формування популяції.

Висновки

У роботі розглянуто біоінспіровані алгоритми оптимізації, а саме метод

бактеріальної оптимізації і метод оптимізації зграєю вовків, які відносяться до методів ройового інтелекту, а також метод диференціальної еволюції. Дані методи працюють одночасно з великою кількістю потенційних розв'язків. Кожен розв'язок поступово поліпшується і оцінюється, таким чином впливаючи на покращення наступного розв'язку. Представлено загальний опис та покроковий алгоритм для кожного розглянутого методу, здійснено їх програмну реалізацію у середовищі MATLAB.

Наведено постановку прикладних задач умовної оптимізації інженерного спрямування, зокрема задачі оптимізації конструкції редуктора і задача оптимізації конструкції натяжної/компресійної пружини. Для чисельного розв'язування даних задач використано метод штрафних функцій.

Проведено обчислювальні експерименти з порівняння отриманих розв'язків задач умовної оптимізації розглянутими методами ройового інтелекту, а також іншими методами різних авторів.

Встановлено, що наведені методи ройового інтелекту та метод диференціальної еволюції можна застосовувати для розв'язування різних за складністю прикладних задач умовної оптимізації. Зазначені методи не накладають жодних обмежень до вигляду цільової функції, що важливо при використанні штрафних функцій, і зажди знаходять наближений глобальний мінімум.

Список використаної літератури:

1. Красношлык Н.А. Решение задачи глобальной оптимизации модифицированным алгоритмом летучих мышей / Н.А. Красношлык // Радіоелектроніка. Інформатика. Управління. – 2015. – № 4(35). – С. 96-103.
2. Талімончик А.С. Дослідження та вдосконалення методів світлячків і зграї вовків при розв'язуванні задач оптимізації і машинного навчання / А.С. Талімончик, Н.О. Красношлик // Вісник Черкаського університету. Серія «Прикладна математика. Інформатика». – 2016. – № 1-2. – С. 77-89.
3. Карпенко А. П. Современные алгоритмы поисковой оптимизации. Алгоритмы, вдохновленные природой : учебное пособие / А. П. Карпенко. – Москва : Издательство МГТУ им. Н. Э. Баумана, 2014. – 448 с.
4. Фразе-Фразенко О.О. Багатоагентний метод виділення інформативних ознак зображень у системах доступу / О.О. Фразе-Фразенко // Вісник Східноукраїнського національного університету імені Володимира Даля. – 2013. – № 15(1). – С. 210-218.
5. Вікіпедія. Стаття «Колективний інтелект» [Електронний ресурс]: [Веб-сайт]. Електронний ресурс. – Режим доступу: https://uk.wikipedia.org/wiki/Колективний_інтелект (дата звернення 19.02.2018) – Назва з екрана.
6. Скобцов Ю. А. Метаэвристики : монография / Ю. А. Скобцов, Е. Е. Федоров. – Донецк: Изд-во «Ноулидж» (Донецкое отделение), 2013. – 426 с.
7. Yang C. Algorithm of Marriage in Honey Bees Optimization Based on the Wolf Pack Search / Chenguang Yang, Xuyan Tu, Jie Chen // *International Conference on Intelligent Pervasive Computing*: 11 - 13 October 2007, IEEE Computer Society, Washington, USA. – 2007. – p. 462-467.
8. Lin M.-H. Design Optimization of a Speed Reducer Using Deterministic Techniques / Ming-Hua Lin, Jung-Fa Tsai, Nian-Ze Hu, and Shu-Chuan Chang // Mathematical Problems in Engineering. – Vol. 2013. – P. 1-7 (Article ID: 419043).
9. Cagnina L. C. Solving engineering optimization problems with the simple constrained particle swarm optimizer / L. C. Cagnina, S. C. Esquivel, C. A. C. Coello // Informatica. – 2008. – Vol. 32. – № 3. – P. 319-326.
10. Mirjalili S. Grey wolf optimizer / S. Mirjalili, S. M. Mirjalili, A. Lewis // Advances in engineering software. – 2014. – Vol. 69. – P. 46-61.

References:

1. Krasnoshlyk N. O. (2015) A Modified bat algorithm for solving global optimization problem. *Radio Electronics, Computer Science, Control*, № 4/35, 88-95 (in Rus)
2. Talimonchyk A. & Krasnoshlyk N. (2016) Research and improvement of methods glowworm swarm optimization and wolf pack search for solving optimization problems and machine learning. *The Cherkasy University Bulletin. Computer Sciences*, №1-2, 77-89 (in Ukr)

3. Karpenko A. P. (2014) Modern algorithms of search optimization. Algorithms inspired by nature: a tutorial. Moscow: Publishing House of the N.E. Bauman MSTU (in Rus)
4. Fraze-Frazenko A. A. (2013) Multi-agent methods for the isolation of informative features from the image. *The Volodymyr Dahl East-Ukrainian National University Bulletin*, 15 (1), 210-218 (in Ukr)
5. Wikipedia. Article "Collective Intelligence" [Electronic resource]: [Web-site]. – Electronic Data. – Access mode: https://uk.wikipedia.org/wiki/Kolective_intelekts (Last accessed: 19.02.2018). The name of the screen. (in Ukr)
6. Skobtsov Yu. A. & Fedorov E. E. (2013) Metaheuristics: monograph. Donetsk: Publishing house "Knolidzh" (Donetsk branch).
7. Chenguang Yang & Xuyan Tu & Jie Chen (2007) Algorithm of Marriage in Honey Bees Optimization Based on the Wolf Pack Search. *International Conference on Intelligent Pervasive Computing*: 11-13 October 2007, IEEE Computer Society, Washington, USA, 462-467.
8. Ming-Hua Lin, Jung-Fa Tsai, Nian-Ze Hu, and Shu-Chuan Chang (2013) Design Optimization of a Speed Reducer Using Deterministic. *Mathematical Problems in Engineering*, 2013, 1-7 (Article ID: 419043).
9. Cagnina L. C. & Esquivel S. C. & Coello C. A. C. (2008) Solving engineering optimization problems with the simple constrained particle swarm optimizer. *Informatica*, 32(3), 319-326.
10. Mirjalili S. & Mirjalili S. M. & Lewis A. (2014) Grey wolf optimizer. *Advances in engineering software*, 69, 46-61.

TALIMONCHYK Anna,
The Bohdan Khmelnytsky National University of Cherkasy, student
KRASNOSHLYK Nataliya,
The Bohdan Khmelnytsky National University of Cherkasy, PhD, Senior Lecturer

SOLVING ENGINEERING CONSTRAINED OPTIMIZATION PROBLEMS WITH SWARM INTELLIGENCE METHODS

Abstract. Introduction. *Solving many of the actual problems in various applied and fundamental sciences is reduced to the global optimization problem. The stochastic methods, which do not impose additional restrictions on the problem statement, are used to effectively solve such problems. Stochastic search algorithms began to intensively develop in the 1980's. Among such algorithms, nature-inspired algorithms, take on an important place.*

Swarm intelligence describes the complex collective behavior of a self-organized decentralized system. Collective intelligence systems, as a rule, consist of a plurality of agents (multi-agent system) that interact locally with each other and with the environment. The agents themselves are usually quite simple, but all together, locally interacting, create the so-called collective intelligence.

Algorithm based on swarm intelligence methods belong to a wider class nature-inspired algorithms. The bio-inspired algorithms make up most of the algorithms inspired by nature. Some bio-inspired algorithms do not use swarm intelligence. The method of differential evolution is not directly related to any biological behavior, but it has similarities with genetic algorithms, and also has the keyword "evolution", so it can be classified as a category of bio-inspired algorithms.

Purpose. *The aim of this article is to apply swarm intelligence methods and differential evolution to the solution of engineering constrained optimization problems.*

Results. *The paper considers bacterial foraging optimization algorithm and wolf pack search algorithm, which relates to the swarm intelligence, and differential evolution method. These methods simultaneously work with a large number of potential solutions. Each solution is gradually improved and evaluated, thus influencing the improvement of the next solution. A general description and step-by-step algorithm for each method under consideration is presented.*

The article presents the engineering constrained optimization problems, in particular problems of tension/compression spring design and welded beam design. For the numerical solution of these problems, the method of penalty functions is used.

Computational experiments were carried out to compare the obtained solutions of the constrained optimization problems with the considered methods of swarm intelligence, as well as other methods of different authors. The given methods of swarm intelligence and differential evolution can be used to solve various on the complexity problems of constrained optimization, was defined. These methods do not impose any restrictions on the objective function, which is important when using penalty functions, and always find an approximate global minimum.

Conclusion. *Swarm intelligence methods can be effectively used to solve the engineering constrained optimization problems. At the same time, the differential evolution method, which relates to evolutionary algorithms, proved to be more effective. Consequently, the considered methods demonstrated their effectiveness and competitiveness in solving complex optimization problems, and foresee further development.*

Key words*: swarm intelligence methods, bacterial foraging optimization algorithm, wolf pack search, differential evolution, constrained optimization problem.*

Стаття надійшла 21.03.2017
Прийнято до друку 24.04.2017

УДК 519.85

PACS 89.20.Bb; 89.20.Ff; 89.20.Kk; 89.40.-a; 89.65.Lm

СТАНІНА Ольга Дмитрівна
ДВНЗ «Український хіміко-технологічний університет», асистент кафедри інформаційних систем
e-mail: stanina@i.ua

КРИТЕРІЙ ОПТИМАЛЬНОСТІ В ОДНОМУ КЛАСІ НЕПЕРЕРВНИХ БАГАТОЕТАПНИХ ЗАДАЧ ОПТИМАЛЬНОГО РОЗБИТТЯ МНОЖИН

Анотація. *У роботі розглянуто двохетапну задачу розміщення підприємств з неперервно розподіленим ресурсом, як різновид задачі оптимального розбиття множин з розміщенням центрів першого етапу без обмежень на їх потужності. Метою даної роботи є отримання необхідніх та достатніх умов оптімальності для задачі оптимального розбиття множин з розміщенням центрів першого етапу без обмежень на їх потужності. Для двохетапної задачі розміщення підприємств з неперервно розподіленим ресурсом сформульовані необхідні і достатні умови оптимальності. Показано, що така задача через функціонал Лагранжа може бути зведена до задачі оптимізації негладкої функції скінченного числа змінних.*

Ключові слова: *оптимальне розбиття множин, задачі розміщення-розподілу, багатоетапні задачі розбиття, критерій оптимальності*

Вступ

З розвитком виробництва та інфраструктури актуальними становляться задачі розміщення-розподілу [2, 7], в яких, наприклад, потрібно знайти на деякій території оптимальні місця розташування підприємств-виробників і виявити зони їх обслуговування так, щоб мінімізувати транспортні витрати на доставку виробленої продукції до споживачів. На даним момент існує декілька підходів до моделювання та розв'язування таких задач. Один з них приводить до дискретних задач розміщення-розподілу (наприклад, задач розміщення на графі або вибір місць розташування із заданої скінченної кількості точок), інший описує розміщення підприємств на певній території з урахуванням неперервності розподілення продукту у цьому регіоні. Розробка математичних моделей та методів таких задач є колом досліджень різноманітних математичних шкіл [3, 8-10, 12], зокрема, науковою школою член-кореспондента НАН України, доктора фізико-математичних наук, заслуженого діяча науки і техніки України професора О.М. Кісельової [4-6]. Сьогодні в рамках вказаної школи проводяться дослідження, пов'язані з подальшим узагальненням теорії оптимального розбиття множин (ОРМ) та розповсюдженням її на нові класи задач.