

СУЧАСНІ МЕТОДИ РОЙОВОГО ІНТЕЛЕКТУ ДЛЯ АНАЛІЗУ СКЛАДНИХ ОБ'ЄКТІВ І СИСТЕМ

М. С. Красноручка^{1,а}, В. В. Хайдуров^{1,2,б}

¹ Навчально-науковий Фізико-технічний інститут

² Інститут загальної енергетики НАН України

Анотація

У даній роботі було проведено дослідження сучасних методів ройового інтелекту, серед яких були наступні алгоритми: алгоритму «Півень-Курка-Курча», алгоритм оптимізації тигрового жука, багатостратегічний покращений ройовий алгоритм пошуку медуз. Їх було протестовано на унімодальних та мультимодальних функціях глобальної оптимізації та досліджено ефективність роботи для кожної з них.

Ключові слова: ройовий інтелект, глобальний оптимум, популяція, вхідні параметри, ітерації, тестування

Вступ

Оптимізаційні задачі є в багатьох галузях науки та техніки. Алгоритми оптимізації можна поділити на детерміновані та точні алгоритми, еволюційні та алгоритми ройового інтелекту. У даній роботі розглядалися саме методи ройового інтелекту. Цей підхід демонструє високу надійність завдяки природі, де колективна поведінка компенсує недоліки окремих елементів. Також важливою перевагою є їхня гнучкість та здатність функціонувати в динамічних середовищах, відстежуючи зміни оптимумів без необхідності змінювати алгоритм. Алгоритми ройового інтелекту будуються на основі таких ключових принципів [1]:

- **децентралізація** – у таких системах відсутній єдиний лідер;
- **самоорганізація** – агенти в рої самостійно адаптуються до змін середовища;
- **виникнення** – колективна поведінка системи виникає внаслідок взаємодії між окремими агентами;
- **стигмергія** – агенти обмінюються інформацією опосередковано, шляхом модифікації навколишнього середовища;
- **адаптивність та надійність** – ройові системи здатні адаптуватися до змін умов та залишаються працездатними при виході з ладу окремих елементів.

1. Постановка задачі

1.1. Формулювання оптимізаційної проблеми

Задачу оптимізації зазвичай записують у вигляді цільової функції $f(x)$, яку потрібно мінімізувати

(або максимізувати), і яка визначає мету задачі. Вектор проектування x є кандидатним розв'язком задачі. Він складається з кількох змінних:

$$x = \{x_1, \dots, x_D\}.$$

Параметр D визначає розмірність задачі. Також простір пошуку обмежено $[lb, ub]$, де

$$lb_i \leq x_i \leq ub_i \quad \text{для кожного } i \in \{1, \dots, D\}.$$

Задача оптимізації тоді описується як процес знаходження такого вектора x^* , щоб виконується наступна умова:

$$f(x^*) \leq f(x) \quad \forall x.$$

У такому випадку розв'язок називається *глобальним оптимумом*.

Однак у більшості практичних задач оптимізації отримати точний розв'язок достатньо складно. Тож достатньо іноді наблизитися до нього. Крім того, більшість практичних задач оптимізації мають обмеження на простір пошуку. Ці обмеження можуть бути виражені у вигляді функцій рівностей та нерівностей:

$$g(x) \leq 0.$$

Коли немає інших обмежень, окрім меж проектних змінних, формальне формулювання задачі оптимізації для мінімізації виглядає так:

$$\min_x f(x).$$

2. Алгоритми ройового інтелекту

2.1. Алгоритм «Півень-Курка-Курча»

Принцип роботи алгоритму полягає в моделюванні колективної взаємодії та ієрархії курей під час пошуку їжі [2]. Коли півні дорослішають, то прагнуть до

^аmarymarusya0@gmail.com

^бallif0111@gmail.com

розмноження. Тому вони намагаються знайти якомога більше їжі, щоб скликати туди самок. Процес пошуку більшої поживи поблизу математично можна описати як випадковий рух від початкової точки:

$$x_i(t+1) = x_i(t) + dance \cdot r_1,$$

де $x_i(t+1)$ та $x_i(t)$ – положення i -го півня на ітерації $t+1$ та поточній ітерації відповідно, $dance$ – це постійний параметр, який у цій роботі приймає значення 0,8, а r_1 – випадковий вектор з рівномірним розподілом. Якщо не знаходить, то переміщуються далі за принципом польоту Леві. Кури групуються біля півня з найбільшою кількістю їжі, досліджуючи навколишній простір у пошуках оптимальних умов, що описується:

$$x_i(t+1) = x_i(t) + a \cdot r_2 [x_b(t) - x_i(t)], a = 2,$$

де $maxIter$ – максимальна кількість ітерацій. r_2 – випадкове число від 0 до 1.

Кури-матері шукають їжу для курчат, а за нестачі їжі вони можуть слідувати за іншим півнем:

$$x_i(t+1) = x_c(t) + a \cdot r_3 [x_c(t) - x_i(t)],$$

де $x_c(t)$ представляє випадково вибраного кандидата серед півнів, r_3 – випадкове число від 0 до 1.

Курчата використовують будь-яку можливість знайти їжу та можуть клювати здобич, яка значно більша за них. Більшість часу курчата слідували за своїми матерями і випадково вибирали, куди йти:

$$x_i(t+1) = x_h(t) + a \cdot r_4 [x_h(t) - x_i(t)].$$

Іноді курчата не впізнавали свою матір, тому $x_h(t)$ був випадково вибраним кандидатом серед курей. Іноді курчата крокували своїм власним шляхом і починали пошук там з невеликою ймовірністю p_1 :

$$x_i(t+1) = r_5(ub - lb) + lb,$$

де $[lb, ub]$ – область визначення задачі (область допустимих значень аргументів функції), r_5 – випадкове число з гаусівського розподілу.

Курчата уникають найгірших позицій, як описано нижче:

$$x_i(t+1) = x_i(t) + a \cdot r_6 [x_w(t) - x_i(t)],$$

де $x_w(t)$ представляє найгіршу позицію на поточній ітерації.

2.2. Алгоритм тигрових жуків

Першим етапом є створення початкової популяції випадковим чином у просторі (1) [3]:

$$TB_i^j = L + (U - L) \cdot \text{rand}, \quad (1)$$

де TB_i^j – це j -та координата i -го рішення, що апроксимує ТВ, L – нижня межа в цьому вимірі, U – верхня межа, а rand – випадкове число від 0 до 1.

Початкова популяція може бути сформована матрицею (2):

$$P = \begin{bmatrix} TB_1^1 & TB_1^2 & \dots & TB_1^D \\ TB_2^1 & TB_2^2 & \dots & TB_2^D \\ \vdots & \vdots & \ddots & \vdots \\ TB_n^1 & TB_n^2 & \dots & TB_n^D \end{bmatrix}, \quad (2)$$

де n – кількість жуків, а D – розмірність кожного рішення.

Оцінка популяції подається у виді:

$$f(P) = \begin{bmatrix} f(TB_1^1) & TB_1^2 & \dots & TB_1^D \\ f(TB_2^1) & TB_2^2 & \dots & TB_2^D \\ \vdots & \vdots & \ddots & \vdots \\ f(TB_n^1) & TB_n^2 & \dots & TB_n^D \end{bmatrix}. \quad (3)$$

Нори створюються біля найбільш оптимальних рішень за допомогою рівняння (4), а найкращі рішення стають орієнтирами:

$$H(TB_i) = \left[1 - \exp\left(-\frac{f(TB_i)}{f(W(t))}\right) \right] \cdot H_m, \quad (4)$$

де $H(TB_i)$ – кількість нір біля TB_i ; H_m – максимальна кількість нір, які копає жук на ділянці; $f(TB_i)$ – придатність TB_i ; $f(W(t))$ – оптимум на ітерації t . H змінюється від найкращого ТВ до найгіршого.

Жук викопує $H(TB_i)$ нір навколо себе, згідно з (4), створюючи нові рішення. Стандартне відхилення поступово зменшується за допомогою функції відстані (5):

$$SD(t) = SD_0 - \left| \frac{p}{\pi} \cdot \arctan\left(\frac{p}{\pi} \cdot t\right) \right|, \quad (5)$$

де p – коефіцієнт контролю зменшення нір та їх розподіл; SD_0 – початкове стандартне відхилення; t – лічильник ітерацій.

Ймовірність виживання розв'язку можна описати рівнянням (6):

$$p(TB_i) = 1 - \frac{1}{\sum_{i=1}^n \frac{f(TB_i)}{f(W(t))}} \times \frac{f(TB_i)}{f(W(t))}. \quad (6)$$

Якщо $r \leq p(TB_i)$, то отвір і жук залишаються; інакше – видаляються.

Останніми етапами є схрещування. Рівняння (7) демонструє пошук у регіоні з найвищою ймовірністю отримання оптимума:

$$TB_i^j = \alpha \cdot TB_i^j + \beta \cdot (B(t) - \bar{B}) \cdot \text{rand}. \quad (7)$$

Тут $B(t)$ – координати найкращого жука, $\bar{B}$ – середнє значення розв'язків, α – коефіцієнт зменшення впливу позиції, а β – коефіцієнт посилення впливу області пошуку, α та β обчислюються в рівняннях (8) та (9).

$$\alpha = 1 - \left(\frac{t}{\text{MaxT}} \right)^2, \quad (8)$$

$$\beta = 1 - \alpha, \quad (9)$$

де MaxT – максимальна кількість ітерацій алгоритму тигрового жука, а t – поточна ітерація.

У пошуку партнера $T B_i$ рухається у напрямку випадково обраних із популяції $T B_l$ та $T B_k$. Рівняння (10) виражає напрямок руху $T B_i$ до двох інших жуків

$$T B_i^j = T B_i^j + (T B_k^j - T B_l^j) \cdot \text{rand}. \quad (10)$$

Вибір двох жуків $T B_k$ та $T B_l$, за якими слідує $T B_i$ впливає на поведінку пошуку та запобігає застряганню в локальному оптимумі.

2.3. Покращений алгоритм пошуку медуз

Щоб підвищити точність алгоритму пошуку медуз є три стратегії [4]:

- 1) введення синусоїдальних та косинусоїдальних факторів навчання;
- 2) введення оператора локального виходу;
- 3) застосування навчання на основі опозиції та квазі-опозиції.

Основні кроки алгоритму пошуку медуз:

- 1) визначити функцію пристосованості, створити початкову популяцію з N особин;
- 2) обчислити та порівняти цільові значення всіх кандидатів. Зберегти оптимальну знайдену позицію та цільове значення;
- 3) обчислити функцію контролю часу $C(t)$ за. Якщо $C(t) > 0,5$, кандидат рухається за течією. Інакше перейти до кроку 4;
- 4) якщо $\text{rand}(0, 1) > 1 - C(t)$, виконати рух Типу А, інакше виконати рух Типу В. Тут
 - (а) **Рух типу А:** пасивний рух, кандидат рухається навколо своєї поточної позиції, і його положення може оновлюватися за рівнянням (18):

$$P_i(t+1) = P_i(t) + \gamma \times r_3 \times (Ub - Lb),$$

де Ub і Lb – це верхня та нижня межі області пошуку відповідно, $\gamma > 0$ – коефіцієнт руху, $\gamma = 0, 1$, $r_3 = \text{rand}(0; 1)$.

- (б) **Рух типу В:** у активному русі випадково вибирається кандидат j . Якщо у положенні P_j більше їжі, ніж у положенні P_i , тоді P_i рухається до P_j , інакше P_i віддаляється.

$$P_i(t+1) = P_i(t) + \overrightarrow{\text{step}};$$

- 5) перевірити, чи не виходить нова позиція за межі простору пошуку. Якщо так – повернутись до протилежної межі;
- 6) якщо нове значення цільової функції краще попереднього – оновити позицію;
- 7) якщо $t < T$, збільшити t та перейти до кроку 3, інакше – до кроку 8;
- 8) повернути найкращу знайдену позицію та її цільове значення.

Фактори ω_1 та ω_2 дозволяють медузі навчатися не лише від випадкової особини, але й від

найкращої в популяції:

$$\omega_1 = 2 \cdot \sin \left[\left(1 - \frac{t}{T} \right) \cdot \frac{\pi}{2} \right],$$

$$\omega_2 = 2 \cdot \cos \left[\left(1 - \frac{t}{T} \right) \cdot \frac{\pi}{2} \right].$$

Оператор локального виходу – це оператор пошуку, який має знаходити нові області пошуку, формує альтернативне рішення поточного $P_i(t+1)$. Методи навчання на основі протилежностей та квазі-протилежного навчання є ефективними засобами підвищення різноманітності кандидатних особин. Застосовуючи ці стратегії для i -ї кандидатної особини P_i у поточній популяції, можна отримати протилежне рішення

$$P_i^e = (P_{i1}^e, P_{i2}^e, \dots, P_{iD}^e)$$

та майже-протилежне рішення

$$\text{hat}P_i = (\hat{P}_{i1}, \hat{P}_{i2}, \dots, \hat{P}_{iD}).$$

Компонентні формули для цих рішень подано нижче:

$$P_{id}^e = Lb_d + Ub_d - P_i^d,$$

$$\hat{P}_{id} = \text{rand} \left(\frac{Lb_d + Ub_d}{2}, Lb_d + Ub_d - P_i^d \right),$$

де P_i^d – d -а координата i -ї особини, Lb_d та Ub_d – нижня та верхня межі простору рішень відповідно для d -го виміру.

Тоді оновлене положення i -ї медузи P_i^{new} визначається за формулою:

$$P_i^{\text{new}} = \begin{cases} P_i^e, & \text{ifrand} < p, \\ \hat{P}_i, & \text{otherwise} \end{cases}, \quad i = \overline{1; N},$$

де p – ймовірність вибору, зазвичай $p = 0,5$.

На основі рівняння (11) генеруються N нових особин. Далі обчислюються значення цільової функції для всіх кандидатів. Після цього з них вибираються найкращі для наступної ітерації.

3. Практичні результати роботи

3.1. Тестування алгоритмів на класичних функціях глобальної оптимізації

Алгоритми було порівняно один з одним та з результатами, які показали генетичний алгоритм, алгоритм рою частинок та методом послідовного квадратичного програмування [5].

F1 – сферична функція

Мінімум $f = 0$, область: $[-100; 100]^D$,

$$f(\mathbf{x}) = \sum_{i=1}^D x_i^2.$$

F2 – еліпсоїдна функція

Мінімум $f = 0$, область: $[-100; 100]^D$,

$$f(\mathbf{x}) = \sum_{i=1}^D i \cdot x_i^2.$$

F3 – функція суми степенівМінімум $f = 0$, область: $[-10; 10]^D$,

$$f(\mathbf{x}) = \sum_{i=1}^D |x_i|^{i+1}.$$

F4 – квінтична функціяМінімум $f = 0$, область: $[-20; 20]^D$,

$$f(\mathbf{x}) = \sum_{i=1}^D |x_i^5 - 3x_i^4 + 4x_i^3 + 2x_i^2 - 10x_i - 4|.$$

F5 – функція спадаючої хвиліМінімум $f = 0$, область: $[-5, 12; 5, 12]^D$,

$$f(\mathbf{x}) = 1 - \frac{1 + \cos\left(12\sqrt{\sum_{i=1}^D x_i^2}\right)}{0.5 \sum_{i=1}^D x_i^2 + 2}.$$

Таблиця 1. Значення найкращих знайдених рішень для $D = 5$

ID	GA	PSO	SQP	CHC	TBO	EJS
F1	2,9E-5	4,1E-5	3,1E-11	0,0	0,0	0,0
F2	2,1E-5	5,5E-5	1,6E-10	0,0	0,0	0,0
F3	7,2E-4	3,8E-4	4,6E-5	0,0	0,0	0,0
F4	2,5E-2	2,5E-2	3,0E-2	0,0	7,3E-2	1,9E-1
F5	1,6E-1	4,8E-2	6,7E-1	0,0	0,0	0,0

3.2. Тестування алгоритмів на наборі CEC2014**F1 – обернена високо обумовлена еліптична функція [6]**Мінімум $f = 100$, область: $[-100; 100]^D$,

$$F_1(\mathbf{x}) = f_1(\mathbf{M}(\mathbf{x} - \mathbf{o})) + F_1^*,$$

де

$$f_1(\mathbf{x}) = \sum_{i=1}^D (10^6)^{\frac{i-1}{D-1}} x_i^2.$$

F2 – обернена функція «Зігнута сигара»Мінімум $f = 200$, область: $[-100; 100]^D$,

$$F_2(\mathbf{x}) = f_2(\mathbf{M}(\mathbf{x} - \mathbf{o})) + F_2^*,$$

де

$$f_2(\mathbf{x}) = x_1^2 + 10^6 \sum_{i=2}^D x_i^2.$$

F3 – обернена функція «Диск»Мінімум $f = 300$, область: $[-100; 100]^D$,

$$F_3(\mathbf{x}) = f_3(\mathbf{M}(\mathbf{x} - \mathbf{o})) + F_3^*,$$

де

$$f_3(\mathbf{x}) = 10^6 x_1^2 + \sum_{i=2}^D x_i^2.$$

о та $\mathbf{M}$ – зсув та обернення глобального оптимуму відповідно.Таблиця 2. Значення найкращих знайдених рішень для F_1 – F_3 в CEC2014

ID	CHC	TBO	EJS
F1	100,0019	2678,1049	1885866,8417
F2	200,0000	200,0000	32476802,0885
F3	300,0000	300,0000	1594,5554

Висновки

У роботі було досліджено три алгоритми, алгоритму «Півень-Курка-Курча», алгоритм оптимізації тигрового жука, багатостратегічний покращений ройовий алгоритм пошуку медуз. Кожен з них тестувався на класичних оптимізаційних задачах та на конкурсному наборі CEC2014, було включено такі критерії як ротація та зсуви, що значно ускладнюють задачу. Практичні результати роботи полягають в оцінці придатності методів до практичних задач, здатності конкурувати одне з одним та загальновідомими алгоритмами.

Перелік використаних джерел

1. R. S., I. C. AI Algorithms and Swarm Intelligence: Exploring Nature-Inspired Solutions in AI. — 2025. — URL: <https://www.unaligned.io/p/ai-algorithms-and-swarm-intelligence>.
2. M. G. Z., J. Z. Cock-hen-chicken Optimizer: A Nature-inspired Algorithm for Real-world Engineering Optimization // Computational Biomedicine. — 2024.
3. A. S., A. A.-S. M., A. F. M. A New Tiger Beetle Algorithm for Cybersecurity, Medical Image Segmentation and Other Global Problems Optimization // Mesopotamian Journal of CyberSecurity. — 2024. — С. 17–46. — URL: <https://doi.org/10.58496/MJCS/2024/003>.
4. EJS: Multi-Strategy Enhanced Jellyfish Search Algorithm for Engineering Applications / H. G., W. J., H. A. G. Li M., A. M. // Mathematics. — 2023. — С. 851. — URL: <https://doi.org/10.3390/math11040851>.
5. V. P., G. S. A Collection of 30 Multidimensional Functions for Global Optimization Benchmarking. — 2022. — URL: <https://doi.org/10.3390/data7040046>.
6. J.-J. L., B.-Y. Q., N. S. P. Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization : тех. звіт. — 2013. — Technical Report 201311.