Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка інтелектуального модуля для розв'язання задач оптимального проєктування конструкцій»

Виконав: студент 2-ого курсу групи 1АКІТ-22м
спеціальності 151 – Автоматизація та комп'ютерно-інтегровані технології
Вовковинський В. О.

Керівник: к.т.н., доцент каф. АІІТ
Іванов Ю. Ю.

« 10 » 12 2023 р.

Опонент: к.т.н., професор каф. КСУ
Биков М.М.

« 10 » 12 2023 р.

Допущено до захисту
зав. кафедри АІІТ
д.т.н., проф. Бісікало О. В.
« 11 » 12 2023 р.

Вінниця ВНТУ – 2023 рік

Вінницький національний технічний університет

Факультет інтелектуальних інформаційних технологій та автоматизації

Кафедра автоматизації та інтелектуальних інформаційних технологій

Рівень вищої освіти II-ий (магістерський)

Галузь знань – 15 – Автоматизація та приладобудування

Спеціальність – 151 – Автоматизація та комп'ютерно-інтегровані технології

Освітньо-професійна програма – Інтелектуальні комп'ютерні системи

ЗАТВЕРДЖУЮ

Завідувач кафедри АІІТ

д.т.н., проф. Бісікало О. В.

«___»_____ 2023 р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Вовковинському Вадиму Олександровичу

1. Тема роботи: Розробка інтелектуального модуля для розв'язання задач оптимального проєктування конструкцій.

   Керівник роботи: к.т.н., доцент каф. АІІТ Іванов Ю. Ю.

   Затверджені наказом ВНТУ від «18» 09 2023 року № 247.
2. Строк подання роботи студентом: до 19.12.2023 р.
3. Вихідні дані до роботи: розмірність простору пошуку; параметри алгоритму (розмір популяції, швидкість агента, гучність, інтенсивність, частота сигналів, кількість ітерацій); кількість запусків програми; набір задач проєктування.
4. Зміст текстової частини: вступ; аналіз предметної області роботи; розробка модифікованої математичної моделі ройового алгоритму кажанів; розробка інтелектуального програмного модуля та експериментальні дослідження; економічний розділ; висновки; список використаних джерел.
5. Перелік ілюстративного матеріалу: принцип роботи особини рою; принцип випадкового блукання; принцип роботи алгоритму кажанів; приклад кривої збіжності; схема програми; приклад роботи програми.

6. Консультанти розділів магістерської кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Іванов Ю. Ю., к.т.н., доцент каф. АІІТ	20.09.2023	04.12.2023
4	Козловський В. О., к.е.н., проф. каф. ЕПВМ	23.09.2023	26.11.2023

7. Дата видачі завдання: «20»____09____ 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області роботи	до 10.10.23	Виконано
2	Розробка модифікованої математичної моделі ройового алгоритму кажанів	до 25.10.23	Виконано
3	Розробка інтелектуального програмного модуля та експериментальні дослідження	до 10.11.23	Виконано
4	Економічний розділ	до 26.11.23	Виконано
5	Оформлення пояснювальної записки і графічного матеріалу	до 04.12.23	Виконано
6	Попередній захист роботи	до 05.12.23	Виконано
7	Остаточний захист роботи	до 19.12.23	Виконано

Студент _____ Вовковинський В. О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Іванов Ю. Ю.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 519.85 + 004.023

Вовковинський В. О. Розробка інтелектуального модуля для розв'язання задач оптимального проєктування конструкцій. Магістерська кваліфікаційна робота зі спеціальності 151 – Автоматизація та комп'ютерно-інтегровані технології, освітньо-професійна програма – Інтелектуальні комп'ютерні системи управління. Вінниця: ВНТУ, 2023. 111 с.

На укр. мові. Бібліогр.: 35 назв; рис.: 29; табл.: 17.

У роботі проаналізовано та модифіковано алгоритм ройового інтелекту, який інспіровано колективною поведінкою кажанів. Представлено його математичний апарат, а також наведено принципи роботи. Розроблено відповідне алгоритмічне та програмне забезпечення, яке дозволяє виконати експериментальні дослідження.

Ключові слова: задача оптимізації, екстремум, функція, метаевристичний алгоритм, кажан, ройовий інтелект.

ABSTRACT

Vovkovynskii V. O. Development of an Intelligent Module for Solving Tasks of Optimal Structures Design. Master's thesis in specialty 151 – Automation and computer-integrated technologies, educational and professional program – Intelligent computer control systems. Vinnitsa: VNTU, 2023. 111 p.

In Ukrainian language. Bibliography: 35 titles; fig.: 29; tabl.: 17.

In this work has been analyzed and modified an algorithm of swarm intelligence, which is inspired by the collective behaviour of bats. It mathematical apparatus has been presented, as well as the working principles. The algorithms and software, allows us to perform some experimental researches have been developed.

Keywords: optimization task, extremum, function, metaheuristic algorithm, bat, swarm intelligence.

ЗМІСТ

ВСТУП

Актуальність. У ході розробки великих технологічних, енергетичних, аерокосмічних, інформаційних та інших складних комплексних систем виникають питання, пов'язані з вибором оптимальної організації взаємодії елементів, режимів їх функціонування, як наслідок, з необхідністю розв'язання задач оптимізації. Такі задачі можуть містити великий обсяг даних, що практично виключає можливість застосування класичних методів пошуку екстремуму функцій зі складним рельєфом поверхонь та обумовлює високу обчислювальну складність пошуку раціональних рішень у багатовимірному просторі альтернатив. Саме тому велика увага розробників приділяється використанню недетерміністичних поліноміальних (метаевристичних) методів, які дозволяють знайти "хороші" розв'язки задачі за прийнятний проміжок часу роботи відповідного програмного забезпечення [1]. Серед таких методів оптимізації окремий клас формує ройовий інтелект, який заснований на використанні програмних агентів з елементами випадковості, що дозволяє задати колективну поведінку певної децентралізованої системи з самоорганізацією пошукових одиниць. Дана галузь обчислювального інтелекту активно розвивається, кількість відповідних досліджень у різних галузях науки та практики збільшується [2, 3].

У науковій літературі представлено досить багато алгоритмів ройового інтелекту, які мають різні характеристики, що актуалізує вивчення їх поведінки під час роботи зі складними практичними задачами. Наприклад, можна виділити праці К.-Ш. Янга [4, 5], Ш. Їлмаза [6], О. Хасанцебі [7],

В.В. Литвина [8] та інших. Вчені виділяють метод оптимізації, інспірований поведінкою кажанів як один із найбільш ефективних та зручних методів, який можна використовувати для розв'язання різноманітних задач оптимального проєктування [9]. Важливою науково-практичною задачею є дослідження роботи даної метаевристики на практичних задачах з урахуванням модифікацій.

Мета і задачі дослідження. Метою роботи є підвищення ефективності розв'язання задачі оптимізації функцій, включаючи задачі оптимального проєктування конструкцій, з використанням ройового алгоритму кажанів за рахунок використання нової моделі міграційного процесу особин зграї.

Для досягнення мети необхідно розв'язати наступні *задачі*:

– розглянути постановку задачі оптимізації функцій та проаналізувати алгоритми її розв'язання, включаючи метаевристики;

– модифікувати математичну модель ройового інтелекту на основі поведінки кажанів;

– розробити програмні засоби, виконати експерименти та оцінити ефективність роботи модифікації.

Об'єктом дослідження є оптимізація функцій, які виникають у ході задач проєктування.

Предметом дослідження є моделі, методи та інструментальні засоби для розв'язання задачі оптимізації заданих функцій.

Методи дослідження. У роботі використано поняття теорії оптимізації та обчислювального інтелекту під час розробки інтелектуального модуля для розв'язання задачі оптимізації функцій у ході проєктування конструкцій. Для

аналізу та перевірки достовірності отриманих теоретичних положень застосовано експериментальне дослідження.

Наукова новизна отриманих результатів: запропоновано модифікацію ройового алгоритму кажанів, особливістю якої є математична модель міграційного процесу зграї, що дозволяє підвищити точність розв'язання задачі оптимізації функцій, які виникають у ході розв'язання задач проєктування.

Практичне значення результатів роботи: розроблене математичне, алгоритмічне та програмне забезпечення дає можливість дослідити вплив параметрів алгоритму кажанів на ефективність розв'язання заданих задач.

Апробація результатів та публікації. За результатами представленої роботи опубліковано тези доповіді на всеукраїнській науково-практичній конференції "Молодь в науці: дослідження, проблеми, перспективи" (м. Вінниця, 2023) [10]. Основні результати можна використати на практиці, що підтверджено довідкою про впровадження розробки.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ РОБОТИ

1.1 Теоретична основа

Перш ніж розв'язувати певну задачу, людина намагається зважити наявну в неї інформацію та вибрати з неї ключові елементи. І тільки потім, коли стане більш менш ясно, на який результат розраховувати, вона приступає до розв'язання самої задачі. Іноді описаний процес називають "з'ясуванням задачі", фактично це заміна вихідної реальної задачі деякою її моделлю [11]. Різноманітність інформаційних аспектів у кожній такій задачі настільки велика, що буває складно з усього різноманіття інформації про явище або об'єкт вибрати найбільш суттєві особливості. У таких випадках необхідно зробити спрощувальне припущення, щоб виділити вихідні дані, визначити, що буде результатом і який зв'язок між вихідними даними та результатом. Усе це – припущення, вихідні дані, результати, зв'язки – називають моделлю задачі.

Ключову роль у моделі відіграє поняття функції. Функцією $y = f(x)$ називається правило, яке ставить у відповідність кожному значенню аргумента x єдине значення y. Функція є функцією від однієї змінної (однопараметричною), якщо аргумент x – скаляр, і функцією від багатьох змінних (багатопараметричною), якщо x – вектор, тобто $x = (x_1, x_2, ..., x_n)^T$.

Під час проєктування різних систем досить часто виникає необхідність досягнути мети не як-небудь, а деяким оптимальним способом, де під оптимальним розуміють такий спосіб, при якому досягається мінімальне або

максимальне значення деякого критерію якості [12].

Цільова функція $f(x)$ (функція втрат, критерій якості) – це функція, яка дозволяє розрахувати значення критерію оптимальності системи. Керовані змінні x (проєктні параметри) – параметри системи, які впливають на значення цільової функції, в процесі оптимізації можуть бути змінені. Обмеження (умови, область допустимих рішень) – множина допустимих значень керованих змінних.

Точка $x*$ функції $f(x)$ називається локальним мінімумом (максимумом) функції $f(x)$, якщо для всіх x, віддалених від точки $x*$ на відстань не більшу деякої малої величини ε, виконується умова

$$f(x^*) \le f(x) \ (f(x^*) \ge f(x)). \tag{1.1}$$

Точка $x**$ функції $f(x)$ називається глобальним (абсолютним) мінімумом (максимумом), якщо для будь-якого x виконується умова

$$f(x^{**}) \le f(x) \ (f(x^{**}) \ge f(x)). \tag{1.2}$$

Глобальний мінімум (максимум) є локальним, зворотнє виконується не завжди. Мінімум (максимум) $x*$ функції $f(x)$ називається сильним, якщо

$$f(x^*) < f(x) \ (f(x^*) > f(x)), \tag{1.3}$$

і слабким, якщо

$$f(x^*) \le f(x)\ (f(x^*) \ge f(x)). \qquad (1.4)$$

Точки мінімуму (максимуму) називають точками екстремуму або оптимуму (рисунок 1.1). Функція може не мати точок екстремуму, мати один локальний мінімум або максимум, мати кілька локальних мінімумів і максимумів тощо.

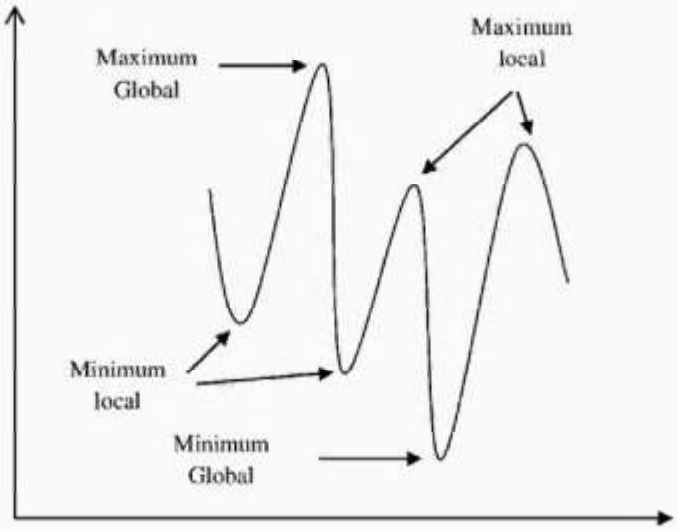

Рисунок 1.1 – Функція з локальним та глобальним екстремумом

Функція $f(x)$ називається монотонно зростаючою, якщо для будь-яких x_1 і x_2, таких що $x_1 < x_2$, виконується умова [13]

$$f(x_1) \le f(x_2), \qquad (1.5)$$

і монотонно спадаючою, якщо

$$f(x_1) \ge f(x_2). \qquad (1.6)$$

Градієнтом $\nabla f(x)$ неперервної диференційовної функції $f(x)$ в точці x називається вектор-стовпчик, елементами якого є частинні похідні першого порядку, обчислені в цій точці

$$\nabla f(x) = \begin{pmatrix} \dfrac{\partial f(x)}{\partial x_1} \\ \dfrac{\partial f(x)}{\partial x_2} \\ \vdots \\ \dfrac{\partial f(x)}{\partial x_n} \end{pmatrix} = \begin{pmatrix} \dfrac{\partial f(x)}{\partial x_1} & \dfrac{\partial f(x)}{\partial x_2} & \cdots & \dfrac{\partial f(x)}{\partial x_n} \end{pmatrix}^T. \qquad (1.7)$$

Градієнт функції направлений перпендикулярно до дотичної площини, проведеної в точці x, в бік найбільшого зростання функції в точці обчислення градієнта, а антиградієнт — навпаки. Нормою градієнта $\|\nabla f(x)\|$ функції $f(x)$ в точці x називають величину [14]

$$\|\nabla f(x)\| = \sqrt{\sum_{i=1}^{n} \left(\frac{\partial f(x)}{\partial x_i} \right)^2}. \qquad (1.8)$$

Поверхнею рівня функції $f(x)$ називають множину точок, в яких функція приймає однакові значення. Для функцій двох змінних поверхні рівня представляють собою лінії рівня.

Матрицею Гессе $H(x)$ неперервної диференційовної в точці x функції $f(x)$ називають матрицю частинних похідних другого порядку, обчислених в

цій точці [11]

$$H(x)=\begin{pmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_3 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_3 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_3 \partial x_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_n^2} \end{pmatrix}. \quad (1.9)$$

Матриця Гессе має розмір $n \times n$ і є симетричною відносно головної діагоналі. Для визначення характеру квадратичної форми використовують критерій Сільвестра (критерій перевірки достатніх умов екстремуму) [12]:

1) матриця Гессе $H(x)$ додатньо визначена тоді, коли всі її визначники (кутові мінори) додатні

$$\Delta_1 = h_{11}, \quad (1.10)$$

$$\Delta_2 = \begin{vmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{vmatrix}, \quad (1.11)$$

$$\Delta_n = \begin{vmatrix} h_{11} & \cdots & h_{1n} \\ \cdots & \cdots & \cdots \\ h_{n1} & \cdots & h_{nn} \end{vmatrix}; \quad (1.12)$$

2) матриця Гессе $H(x)$ від'ємно визначена тоді, коли всі непарні кутові визначники від'ємні, а парні – додатні;

3) при будь-якій іншій комбінації визначників квадратична форма буде

невизначеною.

Також потрібно перевірити необхідні умови екстремуму першого порядку [14]:

1) матриця Гессе $H(x)$ додатньо напіввизначена тоді, коли всі головні мінори її визначника невід'ємні;

2) матриця Гессе $H(x)$ від'ємно напіввизначена тоді, коли всі головні мінори парного порядку для її визначника невід'ємні, а непарного – недодатні.

Головними мінорами називаються визначники m-ого порядку ($m \le n$), які отримують із визначника матриці Гессе $H(x)$ викреслюванням будь-яких (n-m) стрічок та (n-m) стовпців з одними і тими ж номерами.

Інший спосіб перевірки достатніх та необхідних умов екстремуму другого порядку – використання власних значень λ_i матриці Гессе $H(x)$:

$$\begin{vmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} - \lambda & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2^2} - \lambda & \cdots & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_3 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_3 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_3 \partial x_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_n^2} - \lambda \end{vmatrix} = 0. \qquad (1.13)$$

Можна сформулювати такі правила:

1) матриця Гессе $H(x)$ додатньо визначена, якщо значення λ_i додатні;

2) матриця Гессе $H(x)$ від'ємно визначена, якщо значення λ_i від'ємні;

3) при іншій комбінації значень λ_i квадратична форма невизначена.

Наприклад, для функції $f(x) = x_1^2 + x_2^2$ обчислити градієнт, норму градієнта і матрицю Гессе в точках $x^1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, $x^2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, $x^3 = \begin{pmatrix} 0 \\ -2 \end{pmatrix}$, $x^4 = \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix}$.

Обчислення градієнта приведемо в такому вигляді

$$\nabla f(x) = \begin{pmatrix} 2x_1 \\ 2x_2 \end{pmatrix};$$

$$\nabla f(x^1) = \begin{pmatrix} 0 \\ 2 \end{pmatrix};$$

$$\nabla f(x^2) = \begin{pmatrix} 2 \\ 0 \end{pmatrix};$$

$$\nabla f(x^3) = \begin{pmatrix} 0 \\ -4 \end{pmatrix};$$

$$\nabla f(x^4) = \begin{pmatrix} \sqrt{2} \\ \sqrt{2} \end{pmatrix}.$$

Норма градієнта обчислюється наступним чином

$$\left\|\nabla f\left(x^1\right)\right\| = \sqrt{0^2 + 2^2} = 2,$$

$$\left\|\nabla f\left(x^2\right)\right\| = \sqrt{2^2 + 0^2} = 2,$$

$$\left\|\nabla f\left(x^3\right)\right\| = \sqrt{0^2 + (-4)^2} = 4,$$

$$\left\|\nabla f\left(x^2\right)\right\| = \sqrt{\left(\sqrt{2}\right)^2 + \left(\sqrt{2}\right)^2} = 2.$$

Матриця Гессе буде виглядати наступним чином

$$H(x^1) = H(x^2) = H(x^3) = H(x^4) = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}.$$

Записуємо квадратичну форму

$$\Delta x^T H(x) \Delta x = (\Delta x_1 \ \Delta x_2) \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} \begin{pmatrix} \Delta x_1 \\ \Delta x_2 \end{pmatrix} = 2(\Delta x_1^2 + \Delta x_2^2).$$

Таким чином, квадратична форма $\Delta x^T H(x) \Delta x > 0$ для будь-якого ненульового Δx, тобто вона додатньо визначена ($H(x) > 0$).

1.2 Постановка задачі оптимізації функціональних залежностей

Незважаючи на те, що прикладні задачі виникають у абсолютно різних галузях, вони мають загальну форму і класифікуються як задачі мінімізації дійснозначної функції $f(x)$ на певній множині Ω n-вимірного векторного аргументу $x = (x'_1, x'_2, ..., x'_n)^T$ [3].

Множина Ω задається обмеженнями на компоненти вектора x, які задовольняють систему з K рівнянь $h_k(x) = 0$ та J нерівностей $g_j(x) \geq 0$, а також обмежені зверху та знизу ($x_i^{'upper} \geq x'_i \geq x_i^{'lower}$). Тоді математичну постановку можна задати таким чином [10, 12]

$$\begin{cases} f(x) \to extremum\ (\max, \min); \\ h_k(x) = 0,\ k = 1,...,K; \\ g_j(x) \geq 0,\ j = 1,...,J; \\ x_i^{'upper} \geq x'_i \geq x_i^{'lower},\ i = 1,...,n. \end{cases} \tag{1.14}$$

Якщо множина Ω представляє весь n-вимірний простір, тобто обмеження відсутні $K = J = 0$, а $+\infty \geq x'_i \geq -\infty$ $(x_i^{'upper} = +\infty;\ x_i^{'lower} = -\infty)$, то оптимізаційна задача буде безумовною. Її можна представити у такому вигляді

$$f(x) \to extremum\ (\max, \min),\ x \in \Omega;\ \Omega = \{-\infty; +\infty\}. \tag{1.15}$$

Слід зазначити, що у ході розв'язання інженером задач проєктування та управління технічними системами проявляються їх властивості (таблиця 1.1), які на практиці зазвичай можуть викликати труднощі, що призводить до переосмислення задачі та підходів до її розв'язання [11–14].

Таблиця 1.1 – Опис ключових властивостей задач проєктування

№	Властивість	Характеристика
1	Нелінійність	Критерій оптимальності і обмеження задачі є нелінійними функціями в силу нелінійності фізичних законів, які визначають залежності в системі.
2	Висока розмірність простору	Наявна велика кількість керованих змінних n, кожна з яких може приймати m значень.

Продовження таблиці 1.1

№	Властивість	Характеристика
3	Мультимодальність	Велика кількість альтернатив, локальна зміна рішень не дає можливості покращити результат, тому процес може зупинитися в локальному оптимумі.
4	Недиференційовність	Багато актуальних задач не мають аналітичного виразу для функції критерію оптимальності, який можна було б досліджувати за допомогою методів диференціального числення.
5	Мультифакторність	Для досягнення максимальної ефективності роботи системи потрібно вирішувати задачу не тільки параметричної, але й структурної оптимізації, розглядати велику кількість різних керованих змінних.
6	Мультикритеріальність	Системи можуть виконувати декілька функцій, кожна з яких може мати свої критерії та обмеження.
7	Робота в режимі "реального часу"	Швидкість, з якою система реагує на зміни навколишнього середовища, важлива для безпеки і економічності.

У функції може бути багато локальних мінімумів (багатоекстремальна функція). Для знаходження абсолютного (глобального) мінімуму необхідно знайти всі локальні мінімуми, порівняти їх і вибрати найменше значення $f(x^*)$. Відомо, що кількість локальних оптимумів збільшується експоненціально збільшенню розмірності вектора рішення, якщо цільова функція $f(x)$ є мультимодальною функцією. Якщо функція $f(x)$ має в точці x^* мінімум, то для функції $-f(x)$ в цій же точці – максимум. Отже, для пошуку максимуму застосовують ті ж методи, що і для пошуку мінімуму.

1.3 Огляд методів оптимізації

Методи оптимізації – методи пошуку екстремуму функції за наявності обмежень або без них дуже широко використовуються на практиці. Це насамперед оптимальне проєктування (вибір найкращих номінальних технологічних режимів, елементів конструкцій, структури технологічних ланцюжків, умов економічної діяльності тощо), оптимальне управління побудовою моделей об'єктів управління (мінімізація нев'язок різної структури моделі та реального об'єкта) та багато інших аспектів розв'язання економічних та соціальних задач (управління запасами, трудовими ресурсами, транспортними потоками тощо) [1]. Розглянемо низку ключових методів оптимізації.

Метод грубої сили є найбільш простим підходом розв'язання оптимізаційних задач, шляхом безпосереднього перебору всіх можливих

рішень. Складність методу сильно залежить від розмірності простору пошуку. Наприклад, якщо потрібно знайти оптимальну комбінацію n змінних, кожна з яких може приймати m значень, то необхідно проаналізувати m^n комбінацій. Для простої, на перший погляд, задачі з $n = 30$ і $m = 15$ кількість можливих комбінацій буде складати 15^{30}. Тоді, щоб перебрати всі варіанти, для пристрою, який обробляє 10^{12} варіантів за секунду, потрібно $6{,}08 \cdot 10^{15}$ років. Приклад повного перебору для англійського алфавіту та цифр наведено на рисунку 1.2 [12].

Рисунок 1.2 – Приклад повного перебору для підбору паролю

Пошук з поверненням представлено американським математиком Лемером. Суть пошуку з поверненням: розвиток (розширення) раніше сформованого часткового розв'язку. Якщо це неможливо виконати, то алгоритм повертається до одного з минулих станів (розв'язків) та намагається розвивати його. Даний алгоритм має прямий зв'язок з методом гілок і меж, оскільки останній є його модифікацією [11].

Пошук з поверненням може бути реалізований стохастично, тобто процедура вибору нового рішення базується на ймовірнісному правилі. Однак виникає проблема налаштування параметрів вибору. Так, якщо максимальний

крок переходу буде великим, то алгоритм може "застрягнути" в локальному екстремумі і не вийти з нього. Якщо ж крок малий, то алгоритм буде працювати дуже довго.

Удосконалення алгоритмів повного перебору і бектрекінгу є відсікання підмножин, які завідомо не містять оптимальний результат (неперспективні варіанти). Такий метод запропонували Ленд і Дойг. Він був названий метод гілок та меж. Для його роботи необхідні дві процедури: рекурсивне бінарне розгалуження і знаходження оцінок. Процедура розгалуження полягає в розбитті певної підмножини на менші підмножини, що у результаті дозволяє отримати пошукове дерево. Приклад роботи методу оптимізації наведено на рисунку 1.3 [11].

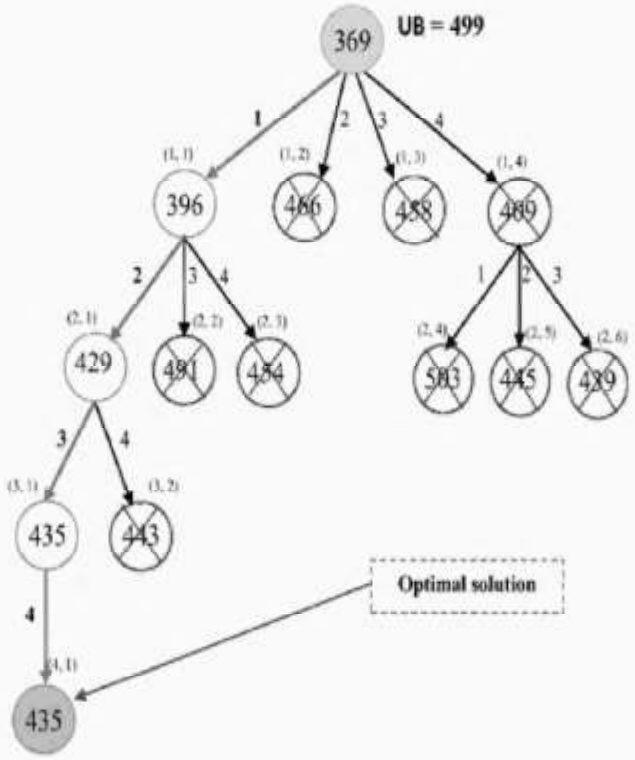

Рисунок 1.3 – Приклад роботи методу гілок та меж

Процедура знаходження оцінок полягає в пошуку верхніх і нижніх меж для вирішення задачі на певній підмножині рішень. В основі методу гілок і меж лежить наступна ідея: якщо нижня межа значень функції на підмножині A пошукового дерева більша, ніж верхня межа на будь-якій раніше розглянутій підмножині B, то A виключається з подальших обчислень, а гілка дерева відрізається (правило відсіву). У результаті можна отримати певне рішення, яке називають рекордом R. Для пошуку глобального оптимуму необхідно перевірити всі вузли, в яких межа менша за поточний рекорд. Час розв'язання задачі оптимізації у загальному випадку експоненціально зростає зі збільшенням розмірності задачі, що робить застосування даного методу обмеженим.

Однією з поширених на практиці задач є задача лінійного програмування, яка подається у вигляді: знайти екстремум функції

$$Z = c_1x_1 + c_2x_2 + ... + c_nx_n \to ext(\max, \min), \quad x_1 \geq 0, x_2 \geq 0, ..., x_n \geq 0 \quad (1.16)$$

за умов

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n \{\leq, \geq, =\} b_1; \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n \{\leq, \geq, =\} b_2; \\ ..... \\ a_{m1}x_1 + a_{m2}x_2 + ... + a_{mn}x_n \{\leq, \geq, =\} b_m. \end{cases} \quad (1.17)$$

Отже, потрібно знайти значення змінних $x_1, x_2, \ldots, x_n$, які задовольняють дані нерівності, тоді як цільова функція набуває екстремального

(максимального чи мінімального) значення. Дану задачу можна звести до стандартної форми, коли в системі обмежень b_i невід'ємні, а всі обмеження є рівностями. Якщо задача лінійного програмування має оптимальний план, то цільова функція набуває екстремуму в одній із вершин багатокутника розв'язків. Для розв'язання подібних задач існує симплекс-метод, ідея якого полягає у переборі вершин опуклого багатогранника в заданому багатовимірному просторі (рисунок 1.4) [14].

Рисунок 1.4 – Приклад роботи симплекс-методу

Алгоритм отримав широкого поширення завдяки своїй універсальності, тобто застосування до різноманітних задач, включаючи багатокритеріальні, і невисокий рівень вимог до обчислювальної потужності технічних засобів. При використанні симплекс-методу задачу необхідно привести до канонічної форми, тобто відповідним чином записати цільову функцію та систему обмежень. Далі сформувати симплекс-таблицю та на кожній ітерації замінювати одну вільну змінну на базисну, при цьому коефіцієнти у таблиці перераховуються відповідно до алгоритму їх перетворення.

Динамічне програмування ґрунтується на принципах Беллмана. Оптимальний розв'язок на кожному кроці визначається станом системи на початку цього кроку. Яким би не був стан системи в результаті певної кількості кроків, на найближчому кроці потрібно вибирати стан таким чином, щоб він в сукупності з оптимальним вибором на всіх наступних кроках призводив до оптимального виграшу на всіх кроках, що залишилися, включаючи даний [12].

Форма завдання, яке розв'язується методом динамічного програмування, не змінюється при зміні кількості кроків n. У цьому сенсі конкретний процес із заданим числом кроків виявляється як би зануреним в сімейство подібних йому процесів і може розглядатися з позиції більш широкого класу задач.

Для простоти можна вважати, що задані початковий s_0 і кінцевий s_N стани системи. Позначимо через $z_1(s_0,u_1)$ значення цільової функції на першому етапі при початковому стані системи s_0 і при управлінні u_1, через $z_2(s_1,u_2)$ – відповідне значення цільової функції на другому етапі, $z_i(s_{i-1},u_i)$ – на i-му етапі, $z_N(s_{N-1},u_N)$ – на N-му етапі. Потрібно знайти оптимальне управління u^*. Для розв'язання цієї задачі занурюємо її в сімейство подібних. Позначимо через $F_1(s_{N-1})$, $F_2(s_{N-2})$, …, $F_k(s_{N-k})$, $F_N(s_0)$ відповідно умовно-оптимальні значення цільової функції z на останньому етапі, двох останніх, на k останніх, на всіх N етапах. Починаємо з останнього етапу. Нехай s_{N-1} – можливий стан системи на початок N-го етапу. Тоді визначаємо, що

$$F_1(s_{N-1}) = z_N(s_{N-1}, u_N), \tag{1.18}$$

$$F_2(s_{N-2}) = z_{N-1}(s_{N-2}, u_{N-1}) + F_1(s_{N-1}), \tag{1.19}$$

$$F_3(s_{N-3}) = z_{N-2}(s_{N-3}, u_{N-2}) + F_2(s_{N-2}), \tag{1.20}$$

$$F_k(s_{N-k}) = z_{N-k+1}(s_{N-k}, u_{N-k+1}) + F_{k-1}(s_{N-k+1}), \tag{1.21}$$

$$F_N(s_0) = z_1(s_0, u_1) + F_{N-1}(s_1). \tag{1.22}$$

Вирази (1.18)–(1.22) називаються функціональними рівняннями Беллмана. Проглядається їх рекурентний характер, тобто для знаходження оптимального управління на N кроках потрібно знати умовно-оптимальне управління на попередніх N-1 етапах і т.д. Фактично динамічне програмування (рисунок 1.5) представляє собою математичний апарат, який дозволяє здійснювати оптимальне планування багатокрокових керованих процесів, розв'язуючи набір підзадач.

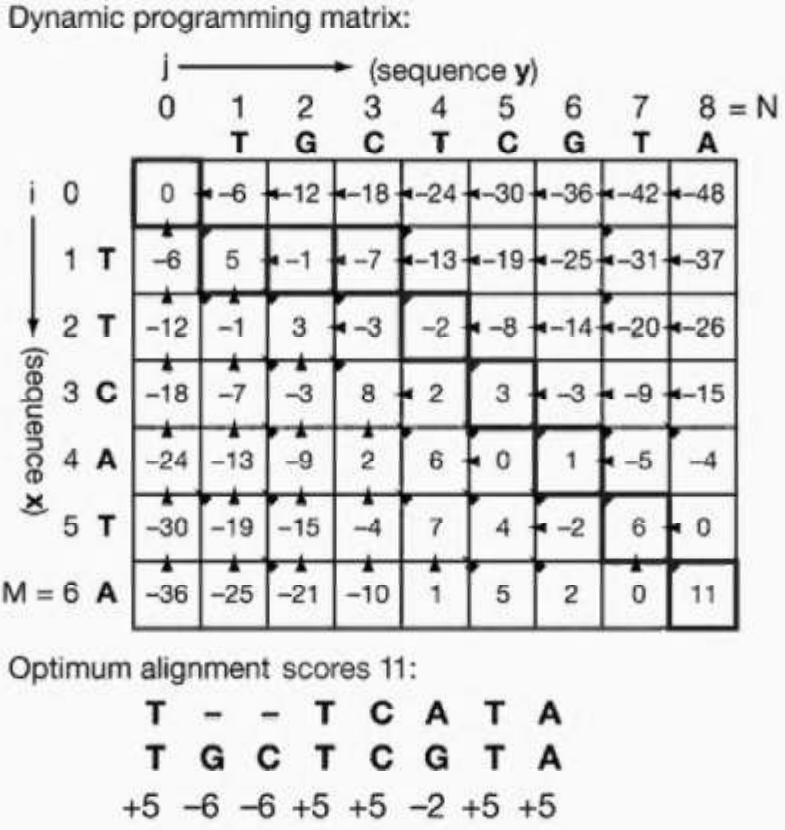

Рисунок 1.5 – Приклад роботи динамічного програмування

Жадібні алгоритми дуже схожі на динамічне програмування, але їх відмінність в тому, що вони працюють з однією локальною задачею, ніколи не повертаючись назад, завжди застосовуючи локально оптимальний вибір. Жадібні алгоритми можуть мати і стохастичні властивості. Для деяких із них, незалежно від початкових умов, можна довести асимптотичну збіжність до глобального оптимуму задачі. Приклад роботи жадібного алгоритму наведено на рисунку 1.6 [14].

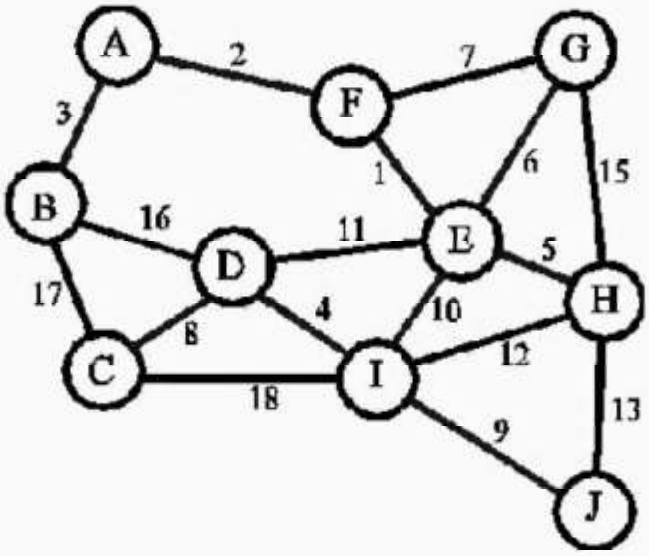

Рисунок 1.6 – Приклад роботи жадібного алгоритму

Але необхідно розуміти, що не існує загального підходу до формування жадібного алгоритму, тобто для кожної конкретної задачі його потрібно налаштовувати. Однак існує досить універсальний спосіб визначення доцільності використання подібного алгоритму для певної задачі. Необхідно з'ясувати чи є підмножини елементів даної задачі матроїдом. Якщо відповідь на це питання позитивна, то відповідно до теореми Радо-Едмондса до задачі можна застосувати жадібний алгоритм, який дозволить отримати оптимум.

Перевагою жадібних алгоритмів є висока швидкість роботи, оскільки на кожному кроці виконується тільки вибір найкращого варіанту без урахування результатів наступних кроків.

До алгоритмів, які базуються на елементах алгебраїчної геометрії відносять, наприклад, аналітичний метод пошуку екстремуму, який використовує необхідні та достатні умови екстремуму, а також квадратичну форму. Клас чисельних методів, які використовують градієнти цільової функції для пошуку її оптимуму, дозволяє сформувати ефективні ітеративні обчислювальні схеми, які є досить простими для реалізації і часто застосовуються для розв'язання ряду нелінійних задач. Але градієнтні алгоритми також мають істотний недолік: одного разу досягнувши локального екстремуму, вони вже неспроможні вибратися з нього. Яскравими представниками даного класу є методи градієнтного спуску, Ньютона, Левенберга-Марквардта, квазіньютонівські методи тощо (рисунок 1.7) [11–14].

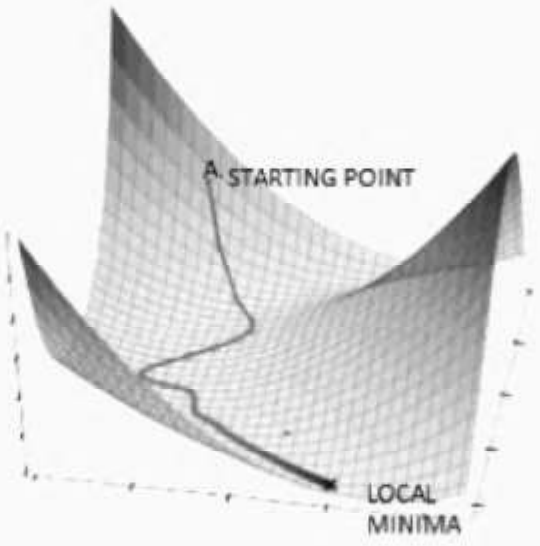

Рисунок 1.7 – Приклад роботи градієнтних алгоритмів

Розв'язання задачі мінімізації функції від багатьох змінних чисельними методами пов'язане зі знаходженням послідовності точок, які задовольняють умову

$$f(x^{k+1}) < f(x^k),\ k = 0, 1, ..., n. \tag{1.23}$$

При цьому пошук мінімуму x^* починається у довільно вибраній точці x^0, а наступні точки послідовності для ітерації $(k+1)$ знаходять за формулою

$$x^{(k+1)} = x^{(k)} + l^{(k)} d^{(k)}, \tag{1.24}$$

де $x^{(k)}$, $x^{(k+1)}$ – поточна та наступна точки;

$l^{(k)}$ – довжина кроку пошуку;

$d^{(k)}$ – напрямок переходу з точки $x^{(k)}$ в точку $x^{(k+1)}$.

У залежності від найвищого порядку частинних похідних функції $f(x)$, методи мінімізації функції поділяють на три групи (рисунок 1.8) [11–14].

Математик Л. Лагранж розробив потужний алгоритм оптимізації, який пізніше отримав його ім'я – множники Лагранжа. Даний алгоритм був узагальнений у роботі Г. Куна і А. Такера, в якій були сформульовані необхідні умови розв'язання задачі нелінійного програмування. Оскільки метод Лагранжа можна застосовувати для задач лінійного та нелінійного програмування, то він використовується для розв'язання різних задач економіки, теорії управління, енергетики та кодування даних [12].

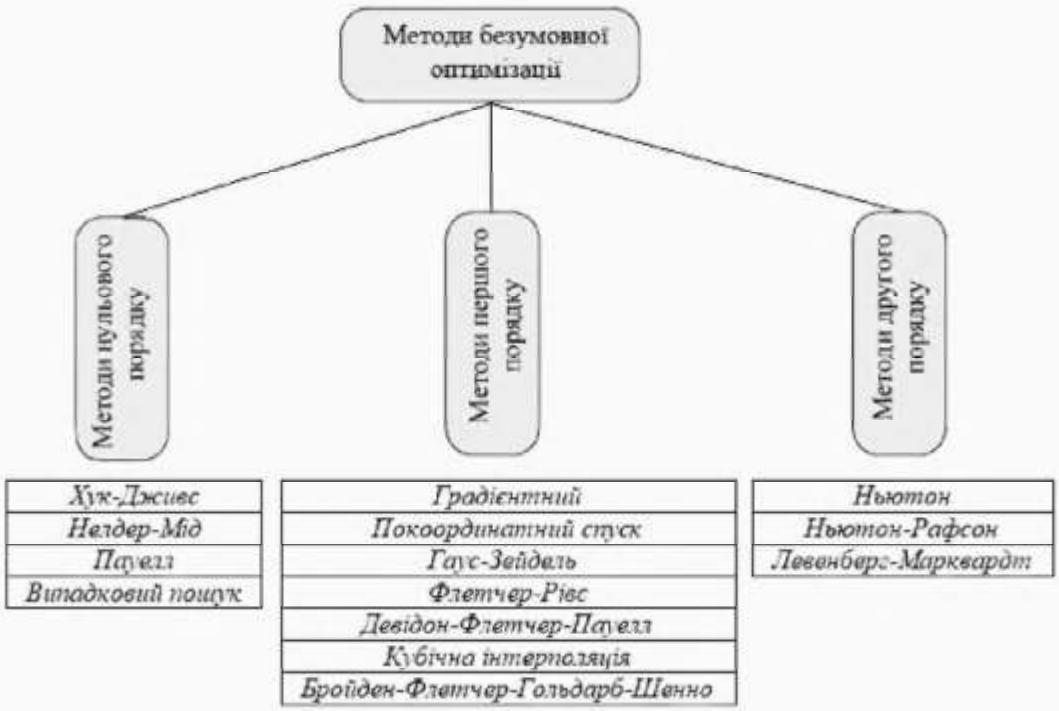

Рисунок 1.8 – Методи розв'язання задач безумовної оптимізації

Якщо необхідно розв'язати задачу оптимізації об'єкта керування у динаміці, то використовують класичне варіаційне числення. Нехай необхідно визначити оптимальне керування $u^*(t)$, яке забезпечує мінімум функціоналу:

$$I = \int_{t_0}^{t_k} F(y, y', u, u', t)dt\,. \tag{1.25}$$

При цьому математична модель об'єкта задана у формі рівняння:

$$y' = \varphi(y, u, t), \tag{1.26}$$

а граничні умови представлено у виразах

$$y(t_0) = y_0, \tag{1.27}$$

$$y(t_k) = y_k. \tag{1.28}$$

Тепер розв'язувати задачу на умовний екстремум можна методом невизначених множників Лагранжа. Для цього вводиться до розгляду новий функціонал [12]:

$$I_1 = \int_{t_0}^{t_k} F(y, y', u, u', t)dt = \int_{t_0}^{t_k} [F(y, y', u, u', t) + \lambda \cdot (y' - \varphi(y, u, t))]dt, \tag{1.29}$$

де λ – множник Лагранжа;

$F(y, y', u, u', \lambda, t)$ – функція Лагранжа;

$y' - \varphi(y, u, t) = 0$ – функція зв'язку.

За допомогою множників Лагранжа задача про умовний екстремум функціоналу (1.25) зводиться до задачі на безумовний екстремум функціоналу (1.29). Для функції Лагранжа складають рівняння Ейлера:

$$\begin{cases} \dfrac{dF}{dy} - \dfrac{d}{dt}\left(\dfrac{dF}{dy'}\right) = 0; \\ \dfrac{dF}{du} - \dfrac{d}{dt}\left(\dfrac{dF}{du'}\right) = 0; \\ \dfrac{dF}{d\lambda} - \dfrac{d}{dt}\left(\dfrac{dF}{d\lambda'}\right) = 0. \end{cases} \tag{1.30}$$

Отримані рівняння називають рівняннями Ейлера-Лагранжа. У результаті їхнього розв'язання з урахуванням математичної моделі об'єкта і

граничних умов можна отримати оптимальне керування $u^*(t)$ об'єктом у динаміці.

Алгоритми Монте-Карло або алгоритм стохастичного моделювання сформувався в рамках Манхеттенського проєкту, в якому фізики досліджували радіаційний захист, відстань, яку нейтрони проходять у речовині до зіткнення з атомним ядром, та кількість енергії, яка виділяється. Зважаючи на велику кількість необхідних даних, задача не могла бути розв'язана за допомогою аналітичних розрахунків. Дж. фон Нейман, Н.К. Метрополіс і С.М Улам в 1949 році запропонували розв'язати її на основі моделювання експерименту на комп'ютері з використанням "випадковості" [12]. Будучи засекреченою, їхня робота вимагала кодове ім'я. Тому така оригінальна назва запозичена у міста Монте-Карло в князівстві Монако, відомому казино, в яких використовується один із самих відомих генераторів випадкових чисел – рулетка (рисунок 1.9).

Рисунок 1.9 – Класична рулетка

Фактично, використовуючи випадковість, метод дозволяє моделювати експеримент, який насправді дуже складно або неможливо поставити. Для

деяких задач – це єдиний практично прийнятний підхід до розв'язання, наприклад, моделювання якості роботи систем зв'язку.

Провівши серію з N незалежних випробувань, одержують вибірку статистичних даних, яку обробляють і представляють у вигляді чисельних оцінок величин (характеристик системи), що цікавлять дослідника. Теоретичною основою методу Монте-Карло є граничні теореми теорії ймовірностей. Вони гарантують високу якість статистичних оцінок при дуже великій кількості випробувань $N \to \infty$, тому алгоритм суттєво спирається на можливості комп'ютера. Він застосовується для дослідження як детермінованих, так і стохастичних систем.

Найбільш простий стохастичний алгоритм оптимізації – це алгоритм сходження на вершину, який представляє собою випадкову генерацію альтернатив (без будь-якого врахування якості раніше знайдених рішень) з подальшим вибором серед них найкращої (рисунок 1.10) [2].

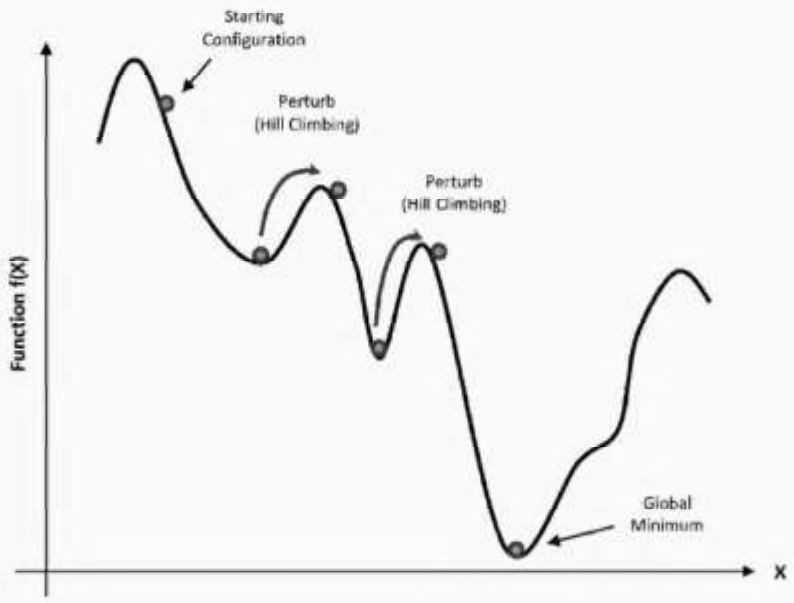

Рисунок 1.10 – Принцип роботи випадкового пошуку

Подібну процедуру можна легко розпаралелити, сформувавши "променевий пошук". Головним недоліком даного методу є відсутність пам'яті про попередні рішення [9].

Вчені С. Кіркпатрік, Д. Желатт, М. Веччі та В. Церні, використовуючи ідеї цієї праці, вперше застосовують метод на основі відпалу, який названо алгоритмом імітацією відпалу. Він заснований на використанні аналогій з процесом кристалізації речовини, в ході якого вона охолоджується і твердне, а швидкість руху молекул падає. Таким чином, поки штучна температура буде великою, алгоритм може робити великі кроки (аналогічні випадковим флуктуаціям в гарячій речовині) навіть у напрямках, які збільшують значення цільової функції. У багатьох випадках ця особливість дозволяє потрапити в окіл глобального мінімуму (природний стан речовини) [12].

Даний алгоритм має високу ефективність і швидкодію, відсутні вимоги до диференційовності цільової функції $f(x)$. Крім того, доведено збіжність до глобального екстремуму незалежно від топології задачі. Тому імітацію відпалу використовують для розв'язання різноманітних оптимізаційних задач. Недоліком є необхідність вдалого вибору евристичних параметрів.

В основі еволюційних алгоритмів лежить аналогія природного відбору (селекція, схрещування, мутація), але відбираються рішення оптимізаційної задачі, тобто виконується ітераційна процедура локального пошуку. Рішення представляється у вигляді вектора значень, який називається хромосомою або особиною, а їх набір на кожному кроці алгоритму називається популяцією. Кожна особина оцінюється на основі значення критерію оптимальності (фітнес-функції). Кращі результати (дочірні популяції або потомство)

відбираються до наступної ітерації і впливають на поточні рішення – схрещуються, мутують (рисунок 1.11) [3].

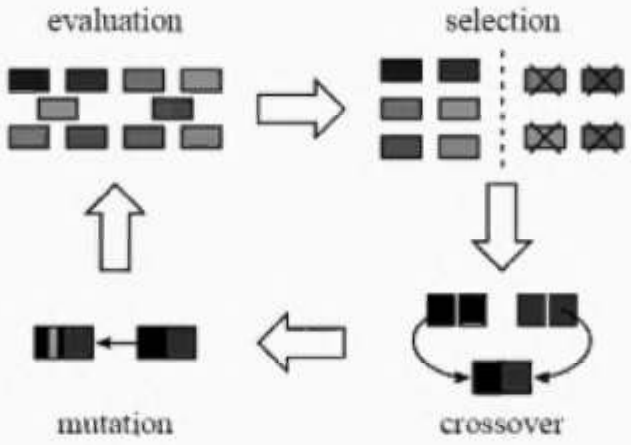

Рисунок 1.11 – Принципи роботи генетичного алгоритму

Принципові моменти механізму еволюції можуть бути формалізовані по-різному. Внаслідок цього з'являється велика кількість еволюційних алгоритмів. Перевагою еволюційних алгоритмів є простота застосування, а недоліками – невисока ефективність для задач великої розмірності і складної структури. Критики справедливо вказують на низьку швидкість їх роботи, оскільки еволюція і в природі протікає повільно.

У праці Х. Бені і В. Цзіна вперше фігурує термін "ройовий інтелект". Формально, рій може бути визначений як група мобільних програмних-агентів, які взаємодіють один з одним (прямо або опосередковано), впливаючи на своє оточення та навколишнє середовище. Така колективна узгодженість дій агентів призводить до розподілених колективних стратегій оптимізації. Отже, вчені звернули увагу на моделі колективної поведінки в природі і

змогли застосувати їх для створення оптимізаційних алгоритмів, які зараз формують область обчислювального інтелекту (рисунок 1.12) [1, 3].

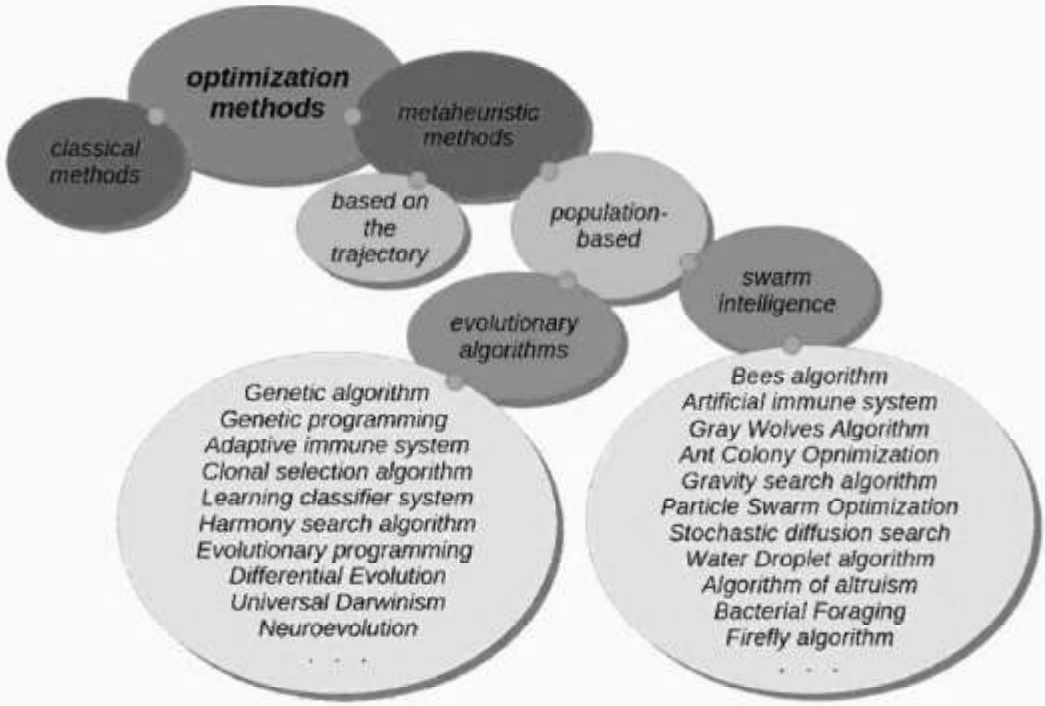

Рисунок 1.12 – Розвиток методів оптимізації та ройових алгоритмів

Потужним алгоритмом ройового інтелекту є алгоритм наслідування поведінки мурашиної колонії, розроблений бельгійським вченим М. Доріго для розв'язання комбінаторних задач, наприклад, задачі комівояжера. Він моделює дії колонії мурах у ході пошуку оптимального маршруту від мурашника до їжі (рисунок 1.13) [3]. Концепція алгоритму активно вдосконалюється як для дискретних, так і для неперервних задач оптимізації.

Однією з перших моделей, яка реалізувала колективну поведінку, стала модель переміщення птахів у зграї, запропонована К. Рейнольдсом. Незважаючи на її простоту, вона давала правдоподібну візуалізацію колективної поведінки зграї. На базі даної моделі Дж. Кеннеді і Р. Еберхарт

розробили алгоритм оптимізації неперервних нелінійних функцій, який назвали алгоритмом рою часток [3].

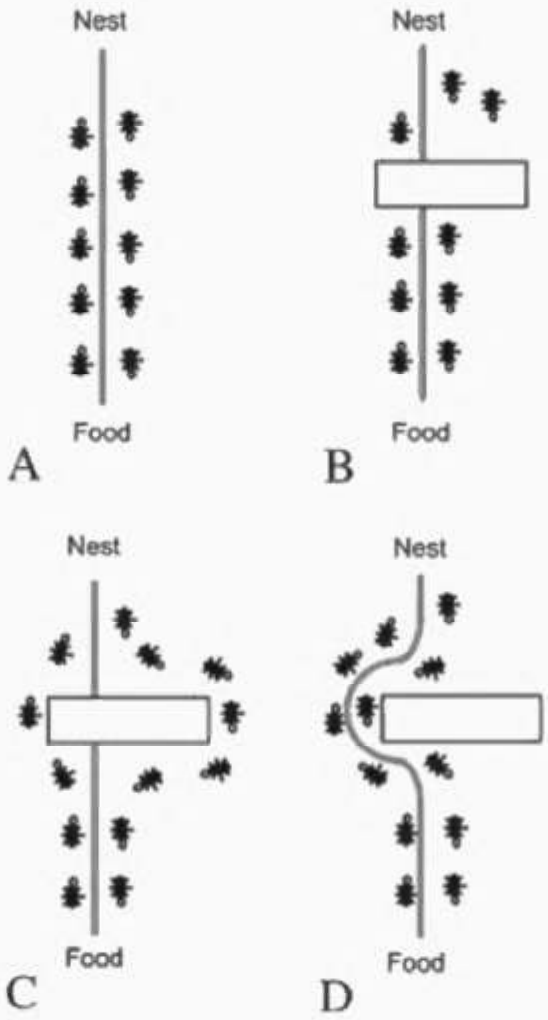

Рисунок 1.13 – Приклад роботи мурашиного алгоритму

Основу алгоритму становить факт, що частки при формуванні рою рухаються до "центру тяжіння" (оптимального рішення), а простір пошуку заповнюється популяцією часток, кожна з яких у конкретний момент часу має низку параметрів. У часток є своя пам'ять, вони можуть обмінюватися інформацією між собою. Для кожного положення частки обчислюється відповідне значення цільової функції, на основі якого за певними правилами,

обчислюються нові координати і швидкість даної частки. Описаний алгоритм має безліч модифікацій (рисунок 1.14).

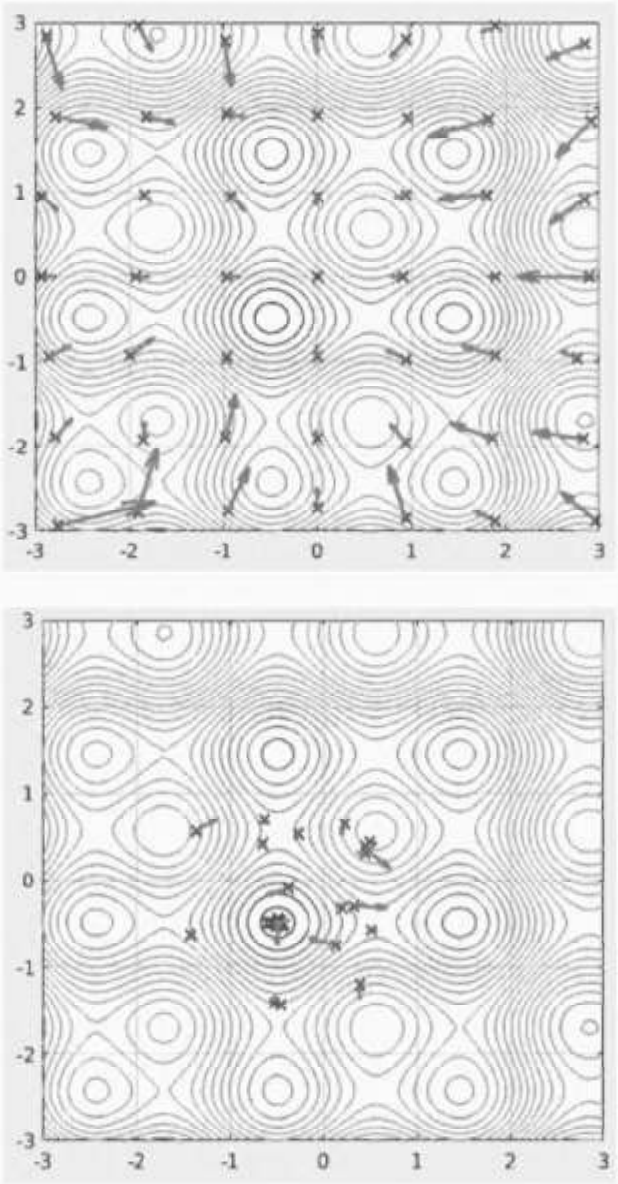

Рисунок 1.14 – Приклад роботи алгоритму рою часток

Також можна виділити алгоритми, які імітують боротьбу імунної системи організму з чужорідними тілами, базуючись на принципах імунології. Суть імунітету живого організму полягає в тому, що він захищається і

знешкоджує чужорідні тіла (антигени), які потрапляють в організм, за допомогою спеціальних речовин (антитіл). Якщо імунні клітини виробили антитіла, які змогли розпізнати антиген, то інформація про ці антитіла зберігається в клітинах пам'яті, що допомагає імунній системі накопичувати досвід та вдосконалюватися, набуваючи нових властивостей. Наступного разу, коли в організм потрапить такий же або схожий антиген, імунітет організму повинен працювати швидше і ефективніше. Особливістю роботи таких алгоритмів оптимізації є ретельне дослідження множини допустимих рішень, що може потребувати багато часу на налаштування алгоритму та пошук екстремуму [3].

Одним з відносно нових алгоритмів ройового пошуку є алгоритм, інспірований поведінкою кажанів, запропонований китайським вченим в галузі оптимізації К.-Ш. Янгом (рисунок 1.15) [4, 5].

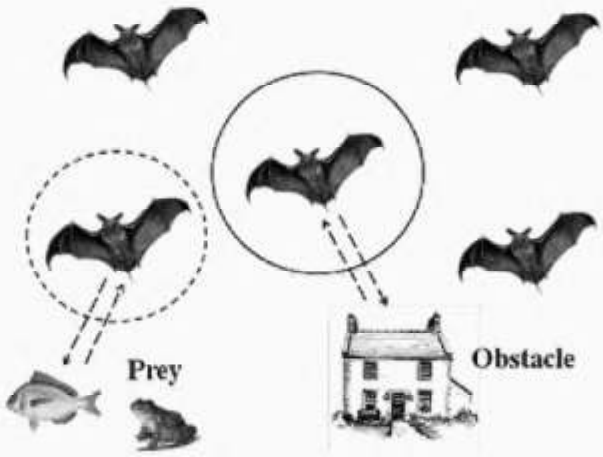

Рисунок 1.15 – Принцип роботи алгоритму кажанів

Всі кажани використовують ехолокацію (певний тип ультразвуку), завдяки якому вони можуть літати і полювати. Даний алгоритм потенційно

більш потужний, ніж алгоритм рою часток, генетичний алгоритм, а також гармонійний пошук. Крім того, гармонійний пошук і алгоритм рою часток представляють собою особливі випадки алгоритму кажанів при певних спрощеннях математичної моделі.

Серед інших ройових алгоритмів можна виділити стохастичний дифузний пошук (stochastic diffusion search, 1989), диференціальну еволюцію (differential evolution, 1997), алгоритм еволюції розуму (mind evolutionary computation, 1998), бактеріальну оптимізацію (bacterial optimisation, 2002), електромагнітний алгоритм (electromagnetism-like algorithm, 2003), алгоритм бджолиного рою (bees algorithm, 2005), алгоритм світлячків (firefly algorithm, 2007), пошук косяком риб (fish school search, 2008), гармонійний пошук (harmony search, 2008), алгоритм зозулі (cuckoo search, 2009), алгоритм гравітаційного пошуку (gravitational search, 2009), алгоритм основі поведінки сірих вовків (grey wolf opimizer, 2014), алгоритм на основі формування гратки кристалічної структури (crystal structure algorithm, 2021) та інші [2].

Перспективним є дослідження гібридних алгоритмів, які комбінують детерміновані та стохастичні підходи для пошуку глобального екстремуму. Гібридні ройові алгоритми можуть включати в себе елементи інших методів оптимізації, машинного навчання або еволюційних алгоритмів для покращення збіжності, підвищення адаптивності алгоритму до змінних умов середовища, кращих результатів.

1.4 Висновки

У даному розділі представлено основі особливості розв'язання задач оптимізації, а також представлено огляд методів оптимізації. Якщо задачу не можна розв'язати, використовуючи детермінований підхід, ройові алгоритми часто дозволяють знайти оптимальні або близькі до них рішення. Актуальною є задача дослідження та розробки ефективних алгоритмів ройового інтелекту. Отже, у наступному розділі необхідно розробити математичну модель одного з розглянутих алгоритмів, який інспірований поведінкою кажанів.

2 РОЗРОБКА МОДИФІКОВАНОЇ МАТЕМАТИЧНОЇ МОДЕЛІ РОЙОВОГО АЛГОРИТМУ КАЖАНІВ

2.1 Дослідження об’єкта автоматизації

Мета автоматизації – полегшення праці, об'єктом автоматизації є діяльність: процес або функція. У роботі розглядається автоматизована інформаційна система, яка орієнтована на обробку інформації. Подібна система для автоматизації інформаційних процесів може застосовуватись у різних галузях діяльності людини, таких як управління, проєктування, виробництво. Проєктування системи включає в себе розробку структури та алгоритмів обробки інформації, а також загальний алгоритм її функціонування. Технічні засоби, які використовуються для автоматизації управління процесами, здатні збирати, обробляти та трансформувати інформацію про стан процесу, передавати інформацію каналами зв'язку, формувати команди управління, використовувати та подавати командну інформацію для впливу на процес та зв'язок з оператором [15].

Необхідно дослідити процес обробки даних, організацію роботи з ними, дати рекомендації до вибору відповідних параметрів тощо. Загальна схема ройового алгоритму наведена у таблиці 2.1. Спочатку в області пошуку формуємо певну кількість початкових наближень до шуканого рішення задачі, тобто ініціалізуємо популяцію агентів. У ході цього процесу можна застосувати детерміновані та стохастичні методи. Зазвичай агентів

розподіляють випадковим чином рівномірно по всій області пошуку екстремуму [10].

Таблиця 2.1 – Схема ройового алгоритму

Крок	Опис
1	Ініціалізація популяції
2	Міграція агентів популяції
3	Завершення пошуку

За допомогою міграційних операторів переміщуємо агентів (випадкове блукання) таким чином, щоб наблизитися до глобального екстремуму цільової функції – функції пристосованості, придатності, корисності, яка оцінює "якість" популяції. Прикладами випадкових блукань можуть послужити траєкторія руху молекули в рідині або газі (броунівський рух), в природі у тварин для виживання (стратегія пошуку їжі), коливання цін акцій на фондовому ринку, фінансовий стан гравця. Всі описані випадки можуть бути апроксимовані моделями випадкового блукання, навіть незважаючи на те, що вони можуть не бути повністю випадковими в реальному житті. Випадкове блукання пояснює поведінку багатьох процесів і представляє собою фундаментальну модель для зареєстрованої стохастичної активності.

У стохастичних алгоритмах оптимізації на етапі міграції випадкові блукання представляють велику цінність, оскільки дозволяють досліджувати простір рішень. Для генерації кроку дослідження зазвичай використовують

кілька законів розподілу випадкових чисел. Приклад випадкового блукання показано на рисунку 2.1 [16].

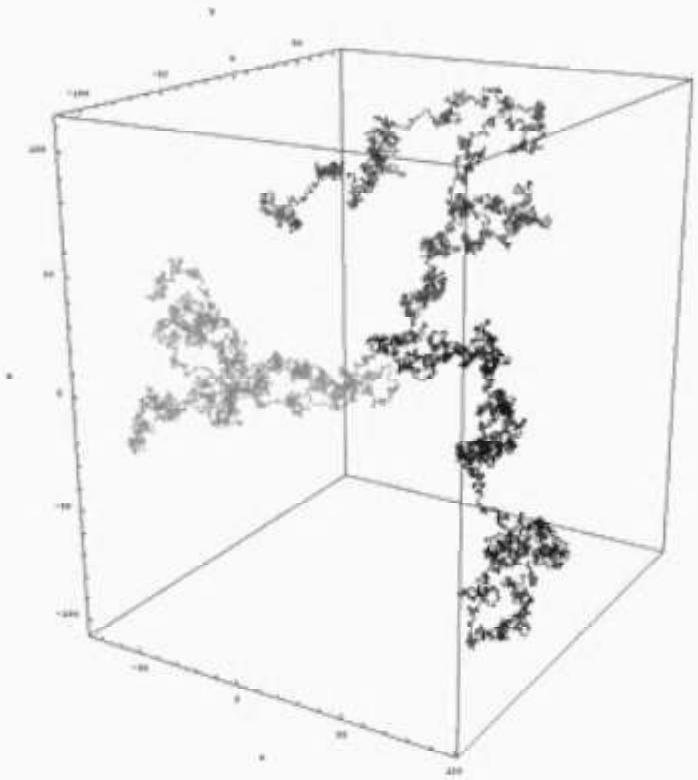

Рисунок 2.1 – Приклад випадкового блукання

В якості базової послідовності випадкових чисел для імітації випадкових факторів різної природи необхідно вибирати таку послідовність, яка може бути отримана з найменшими витратами машинного часу і, крім того, забезпечує простоту і зручність подальших перетворень. Практика показує, що найкраще даним вимогам задовольняє послідовність випадкових чисел з рівномірним розподілом на інтервалі:

$$f(x \mid a,b) = \frac{1}{b-a},\ a \le x \le b\,. \qquad (2.1)$$

У Python для генерації випадкового числа, розподіленого рівномірно на відрізку $[a, b] = [0, 1]$, використовується функція random.uniform(a, b). Функція щільності ймовірності рівномірного розподілу наведена на рисунку 2.2 [17].

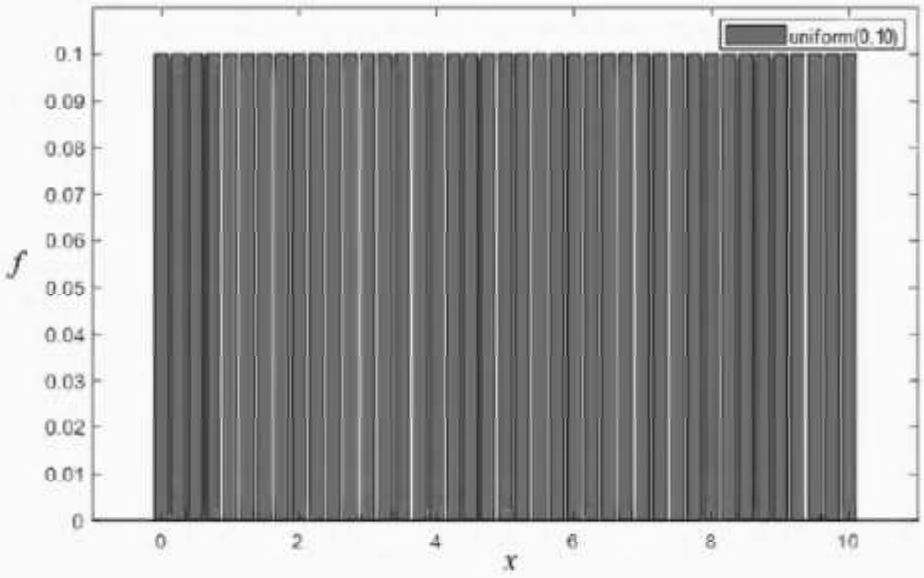

Рисунок 2.2 – Приклад функції щільності рівномірного розподілу

Ще одним популярним випадковим блуканням є гаусівське блукання, яке використовує нормальний розподіл з математичним очікуванням μ і середнім квадратичним відхиленням σ:

$$f(x \mid \mu,\sigma) = \frac{1}{\sigma \cdot \sqrt{2 \cdot \pi}} \cdot \exp\left(- \frac{(x-\mu)^2}{2 \cdot \sigma^2} \right), -\infty < \mu < \infty, \sigma > 0. \quad (2.2)$$

У Python використовується метод Зіккурат, який представляє собою приклад вибірки з відхиленням. Він випадковим чином генерує точку, незначно відхилену від потрібного розподілу, а потім перевіряє чи потрапила вона точно всередину такого розподілу. Якщо ні, то алгоритм повторюється

знову. Для генерації випадкового числа, розподіленого нормально $N(\mu,\sigma)$, використовується функція random.normalvariate(μ,σ). Функція щільності ймовірності нормального розподілу наведена на рисунку 2.3 [17].

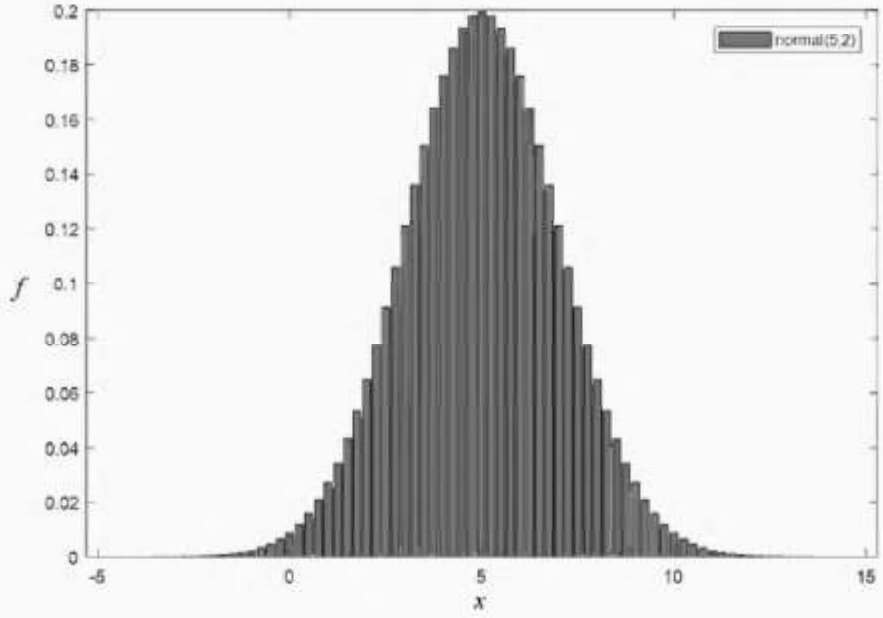

Рисунок 2.3 – Приклад функції щільності нормального розподілу

Після отримання стандартної нормальної величини $N(0,1)$ можна перейти до величини $N(\mu,\sigma)$, розподіленої нормально з математичним очікуванням μ і середнім квадратичним відхиленням σ, за формулою:

$$N(\mu,\sigma) = \mu + \sigma \cdot N(0,1). \tag{2.3}$$

Ще одним видом випадкових блукань є польоти Леві, які характеризуються серією стрибків, обумовлених функцією щільності ймовірності з важкими хвостами, за рахунок яких ймовірність значних відхилень від середнього більша, ніж у нормального розподілу. Розподіл із

важким хвостом – це розподіл, хвіст якого не можна "відрізати", тобто не можна знехтувати великими, але рідкісними подіями.

У польотах Леві розмір кроку контролюється розподілом імовірностей саме з таким важким хвостом, зазвичай відомим як розподіл Леві, у якому математичного очікування та дисперсії не існує. Польоти Леві мають відношення до фракталів, оскільки в них фрагменти є подібністю до цілого. Якщо накреслити траєкторію польоту Леві, то вийде велика фігура, яка складається з менших, що формою нагадують велику.

Закон розподілу Леві з параметрами β, μ можна навести у вигляді [16]

$$f(x \mid \beta, \mu) = \sqrt{\frac{\beta}{2 \cdot \pi \cdot (x-\mu)^3}} \cdot \exp\left(-\frac{\beta}{2 \cdot (x-\mu)}\right), \mu < x < \infty. \qquad (2.4)$$

Функція щільності ймовірності наведена на рисунку 2.4.

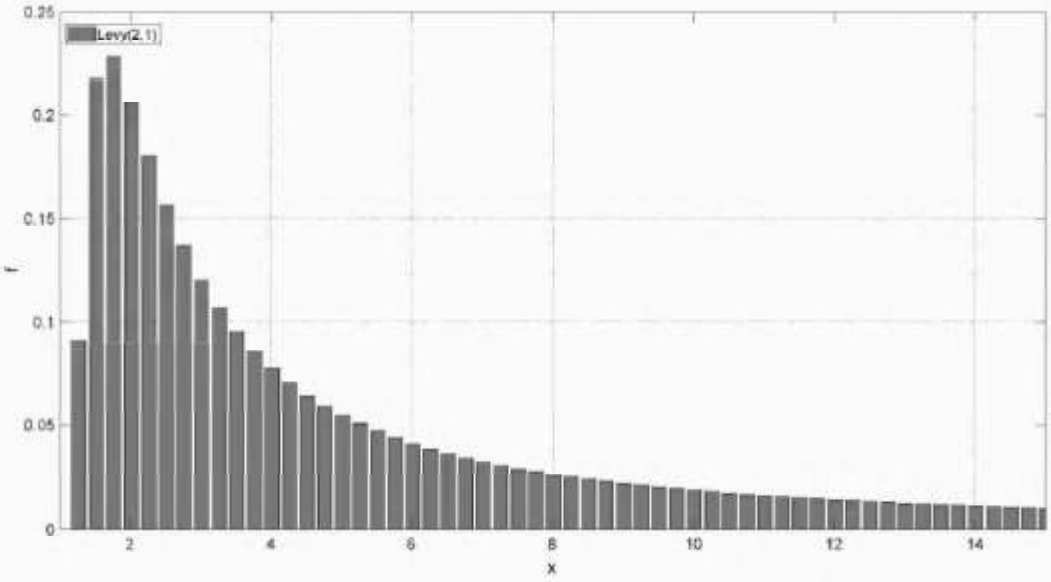

Рисунок 2.4 – Приклад функції щільності розподілу Леві

Останнім кроком ройового алгоритму є перевірка виконання умови закінчення ітераційного процесу (час роботи; кількість ітерацій або поколінь; стагнація алгоритму – найкраще отримане значення не змінюється впродовж декількох поколінь тощо) і, якщо вони виконані, припиняємо обчислення, вважаючи найкраще із знайдених положень агентів популяції наближеним розв'язком задачі.

Оскільки ітераційний процес не може тривати нескінченно, потрібно вибрати відповідну умову його закінчення – критерій збіжності ітерацій, при виконанні якого, останнє знайдене ітераційне наближення буде прийняте за шукане рішення [3, 10].

Розглянемо кілька критеріїв закінчення ітераційного процесу. Контролювати збіжність можна одночасно всіма критеріями або вибірково деякими з них. Зазвичай використовують такі критерії зупинки пошуку:

а) знайдено відомий глобальний оптимум $f(X^{best})$, де $X^{best} = (x_1, x_2, ..., x_n)^T$;

б) досягнуто граничних значень

1) ітерацій

$$i > limit; \tag{2.5}$$

2) часу

$$t > t\_limit; \tag{2.6}$$

3) звернень до цільової функції

$$f\_call > f\_limit. \quad (2.7)$$

в) стагнація пошуку:

1) найкраще рішення не змінюється досить довго;

2) всі агенти знаходяться на дуже близькій відстані *eps* від найкращого рішення;

3) середня якість популяції не змінюється досить довго.

Слід зазначити, що ні один метод або клас методів не відрізняється високою ефективністю при вирішенні оптимізаційних задач різних типів. Це представлено у "теоремі про відсутність безкоштовних сніданків" для оптимізації. Зокрема, можливі випадки, коли відбувається переповнення пам'яті ЕОМ; в інших ситуаціях обчислення значень цільової функції вимагає надмірних витрат часу; в деяких завданнях потрібно отримати дуже точне рішення; інколи параметри методу підібрані невдало тощо [2].

2.2 Загальний алгоритм кажанів

Робота алгоритму базується на наступній моделі колективної поведінки кажанів [4, 5, 18–21]:

1. Всі кажани використовують ехолокацію, щоб визначати відстань, а також розрізняти їжу / здобич і перешкоди.

2. Поточне положення кожного кажана позначимо $X_i = (x_{i1},...,x_{in})^T$, з нього вони переміщаються випадковим чином зі швидкістю v_i. У процесі руху

кажани випускають звукові сигнали, які мають частоту ω_i і гучність A_i. При цьому вони можуть змінювати як частоту, так і інтенсивність імпульсів $r_i \in [0,1]$, в залежності від близькості до мети.

3. Гучність звукового сигналу змінюється від більшого початкового $A_{\max} = A_0$ до меншого заданого значення $A_{\min}$, тобто вважаємо, що гучність знаходиться у фіксованому діапазоні $[A_{\max}, A_{\min}]$.

4. Кожний кажан представляє розв'язок задачі (його координати у просторі пошуку та якість джерела їжі).

Розглядаємо задачу мінімізації функції. Алгоритм, інспірований поведінкою кажанів, включає в себе наступні кроки [4, 5]:

1. Ініціалізація параметрів:

– *LB*, *UB* – нижня і верхня межі області допустимих значень *D* розмірності *n*, *UB* > *LB*;

– популяція кажанів $S = (s_i, i \in [1,|S|])$;

– $X_i = (x_{i1},...,x_{in})^T$ – положення *i*-ого кажана;

– v_i – швидкість кажана;

– $A_i = [A_{\max}, A_{\min}]$ – гучність звукового сигналу;

– $[A_{\max}, A_{\min}]$ – мінімальне та максимальне значення гучності звукового сигналу;

– $r_i^{(0)} = [0, 1]$ – інтенсивність звукового сигналу;

– $\omega_i = [\omega_{\min}, \omega_{\max}]$ – частота звукового сигналу;

– $\omega_{\min}, \omega_{\max}$ – мінімальне та максимальне значення частоти звукового сигналу;

– *limit* – максимальна кількість ітерацій алгоритму.

Якщо при генерації нового рішення його *j*-та компонента виходить за межі області *D*, то проводиться декілька спроб згенерувати її заново. Якщо після закінчення *k* спроб нове рішення постійно знаходиться за межами множини *D*, то дана координата замінюється на відповідне граничне значення.

$$x_{ij}^{new} = \begin{cases} LB_j, \text{ якщо } x_{ij} < LB_j; \\ UB_j, \text{ якщо } x_{ij} > UB_j, \end{cases} \tag{2.8}$$

де LB_j, UB_j – нижня та верхняя межі множини допустимих рішень по *j*-ій координаті, $UB_j > LB_j$.

Подібну перевірку можна реалізувати програмно без циклу в один рядок:

$$x_{ij}^{new} = \max(\min(x_{ij}, UB_j), LB_j). \tag{2.9}$$

2. Генеруємо та оцінюємо якість агентів популяції $f(X_i) = f_i$, $i \in [1,|S|]$, $X_i = (x_{i1},...,x_{in})^T$. Визначаємо на цій основі глобально найкращого агента s^{best}.

3. Виконуємо міграційну процедуру для *i*-ого кажана::

3.1. Знаходимо частоту звукового сигналу:

$$\omega_i = \omega_{\min} + rand \cdot (\omega_{\max} - \omega_{\min}), \tag{2.10}$$

де *rand* – незалежне дійсне випадкове число, рівномірно розподілене на інтервалі [0, 1].

3.2. Обчислюємо швидкість:

$$v_i^{(t+1)} = v_i^{(t)} + \omega_i \cdot \left(X_i^{(t)} - X^{best} \right), \tag{2.11}$$

де X^{best} – положення кажана, в якому знайдено найкраще глобальне рішення.

3.3. Визначаємо нове положення:

$$X_i^{(t+1)} = X_i^{(t)} + v_i^{(t+1)}. \tag{2.12}$$

4. Якщо для *i*-ого кажана $rand > r_i$, то виконати локальний пошук в околі поточного стану кажана з використанням випадкового блукання:

$$X_i^{new} = \begin{cases} X_i^{old} + U(a,b) \otimes A_{mean}^{(t)}, & якщо\ rand > r_i; \\ X_i^{old}, & інакше, \end{cases} \tag{2.13}$$

де $U(a, b)$ – $(n \times 1)$-вектор незалежних дійсних випадкових чисел, рівномірно розподілених на інтервалі $[a, b]$;

$a = -1$; $b = 1$ – дійсні числа, $b > a$;

rand – незалежне дійсне випадкове число, рівномірно розподілене на інтервалі [0, 1];

$\otimes$ – поелементне множення;

$A_{mean}^{(t)}$ – середнє значення гучності всіх звукових сигналів на ітерації *t*:

$$A_{mean}^{(t)} = \frac{1}{|S|} \cdot \sum_{i=1}^{|S|} A_i \,. \tag{2.14}$$

5. Якщо $rand < A_i$ і $f\left(X_i^{new}\right) \le f\left(X_i^{best}\right)$, то прийняти нове положення:

$$X_i^{(t+1)} = X_i^{best} = \begin{cases} X_i^{new}, \text{ якщо } rand < A_i \text{ and } f\left(X_i^{new}\right) \le f\left(X_i^{best}\right); \\ X_i^{old}, \text{ інакше,} \end{cases} \quad (2.15)$$

де $f\left(X_i^{best}\right)$ – якість найкращої позиції i-ого кажана.

Якщо умова виконана, то зменшити A_i і збільшити r_i з використанням еволюції параметрів:

$$A_i^{(t+1)} = \alpha \cdot A_i^{(t)}, \quad (2.16)$$

$$r_i^{(t+1)} = r_i^{(0)} \cdot \left(1 - \exp(-\gamma \cdot t)\right), \quad (2.17)$$

де α, γ – задані константи, рекомендовані значення яких дорівнюють 0,9.

Тобто зі збільшенням кількості ітерацій гучність A_i звукових сигналів буде зменшуватися, а їх інтенсивність r_i – збільшуватися, моделюючи цим наближення кажана до мети $A_i^{(t)} \to 0, r_i^{(t)} \to r_i^{(0)}$ при $t \to \infty$.

6. Вибрати глобально кращого агента s^{best} аналогічно до кроку 2. Тобто якщо якість знайденого рішення $f(X_i)$ не більша збереженого глобального розв'язку $f(X^{best})$, то глобально кращим стає положення кажана зі значенням функції $f(X_i)$:

$$f(X^{best}) = \begin{cases} f(X_i), \quad \text{якщо } f(X_i) \le f(X^{best}); \\ f(X^{best}), \text{ інакше.} \end{cases} \quad (2.18)$$

7. Перевірити критерій зупинки пошуку. Наприклад, якщо $t > limit$, то перейти на крок 8; інакше задати $t = t + 1$ і перейти на крок 3.

8. Вивести глобально найкраще рішення $f^{best}(X^{best})$ для агента s^{best}.

Рисунок 2.5 ілюструє принцип роботи алгоритму кажанів.

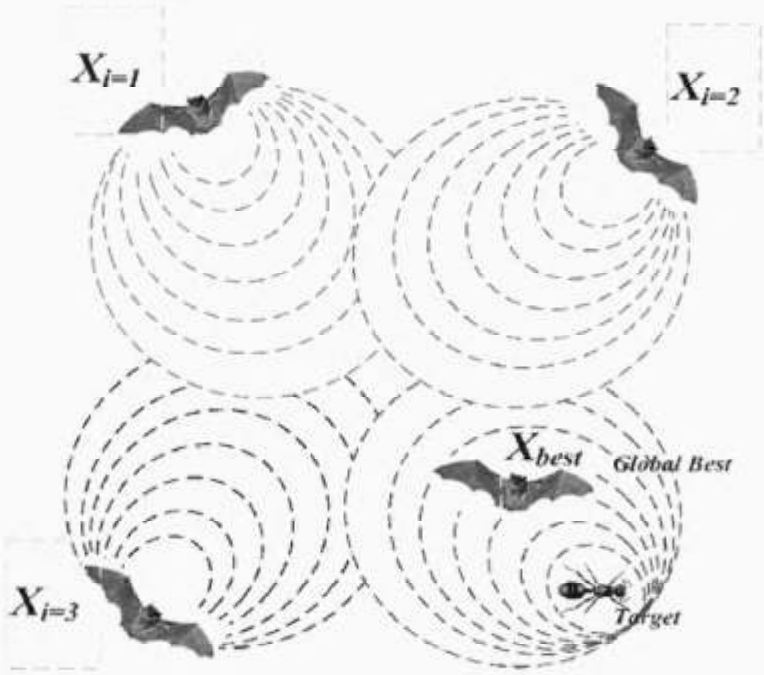

Рисунок 2.5 – Приклад роботи алгоритму

Алгоритм кажанів можна успішно використовувати для розв'язання складних комплексних задач оптимізації, завдяки простоті, гнучкості та ефективності. Перевагами є:

- простота обчислень;
- швидкість роботи;
- збіжність до оптимуму;
- можливість розпаралелювання,

а з недоліків можна виділити:

- складність аналізу;
- невідомий час збіжності.

Отже, дослідник повинен налаштовувати даний метод до певної конкретної задачі. Пошук рішення завжди залишається мистецтвом, якому можна навчитися лише шляхом проб і помилок, застосовуючи різні методи для вирішення конкретних задач.

2.3 Запропонована модифікація

Запропоновано використання моделі переміщення зграї, яка базується на принципі "виживання найкращих" [3, 22, 23]. Нехай задано цільову функцію $f(X) = f(X_1,...,X_n)$, яку потрібно оптимізувати (знайти мінімум). Відповідно на будь-якій ітерації i існує n проєктних змінних ($j = \overline{1,n}$), які обмежені нижньою і верхньою межами області допустимих значень D, при цьому розмір популяції алгоритму представляє кількість варіантів розв'язків. Найкращий агент популяції *best* отримує найкраще поточне значення $f(X)$, а найгірший агент *worst* визначає найгірше поточне значення $f(X)$.

Тоді для пошуку значень проєктних змінних X для i-ого кажана та j-ої координати на ітерації t можна застосувати наступну міграційну формулу:

$$X_{i,j}^{(t+1)} = X_{i,j}^{(t)} + \left(rand_i \cdot v_{i,j}^{(t)}\right) + \left(B_{i,j}^{(t)} + W_{i,j}^{(t)}\right), j = \overline{1,n}, t = \overline{1,\text{limit}}, \qquad (2.19)$$

де $rand_i$ – незалежне дійсне випадкове число, рівномірно розподілене на інтервалі [0, 1];

$v_{i,j}^{(t)}$ – швидкість кажана;

$\omega_{1,i}, \omega_{2,i}$ – частоти сигналів;

$X_{i,j}^{(t)}$ – поточна позиція кажана;

$B_{i,j}^{(t)}$, $W_{i,j}^{(t)}$ – параметри, які враховують наближення до найкращого та віддалення від найгіршого розв'язку.

Пошуковий процес алгоритму дає можливість рішенню-кандидату наблизитися до найкращої позиції

$$B_{i,j}^{(t)} = \omega_{1,i} \cdot (X_j^{(t),best} - | X_{i,j}^{(t)} |), \tag{2.20}$$

та віддалятися від найгіршої позиції

$$W_{2,i}^{(t)} = -\omega_{2,i} \cdot (X_j^{(t),worst} - | X_{i,j}^{(t)} |), \tag{2.21}$$

де $X_j^{(t),best}, X_j^{(t),worst}$ – найкраща та найгірша позиції кажанів у зграї.

Це дозволяє більш успішним рішенням у популяції "підштовхувати" менш успішні у оптимальному напрямку, виконувати ретельний аналіз області пошуку та уникати застрягання у локальному оптимумі. Щоб продемонструвати роботу формули задаємо цільову функцію

$$f(x_1, x_2) = x_1^2 + x_2^2 \to \min, -100 \le x_1, x_2 \le 100, f_{\min}(0,0) = 0$$

та n = 5 кажанів у рої. Початкові рішення-кандидати генеруються випадковим чином у межах заданої області пошуку (таблиця 2.2).

Таблиця 2.2 – Стартова популяція

№	X_1	X_2	$f(X)$	Статус
1	-5	18	349	
2	14	63	4165	
3	70	-6	4936	*worst*
4	-8	7	113	*best*
5	12	-18	468	

1. Результати обчислень за міграційною формулою занесені до таблиці 2.3. Для прикладу знайдемо нове рішення для агента 1:

$$\begin{aligned} X_1^1(1) &= X_1^{(0)}(1) + 0{,}58 \cdot (X_1^{(0)}(best) - |X_1^{(0)}(1)|) - 0{,}81 \cdot (X_1^{(0)}(worst) - |X_1^{(0)}(1)|) = \\ &= X_1^{(0)}(1) + 0{,}58 \cdot (X_1^{(0)}(4) - |x_1^{(0)}(1)|) - 0{,}81 \cdot (X_1^{(0)}(3) - |X_1^{(0)}(1)|) = \\ &= -5 + 0{,}58 \cdot (-8 - |-5|) - 0{,}81 \cdot (70 - |-5|) = -65{,}19; \end{aligned}$$

$$\begin{aligned} X_2^1(1) &= X_2^{(0)}(1) + 0{,}92 \cdot (X_2^{(0)}(best) - |X_2^{(0)}(1)|) - 0{,}49 \cdot (X_2^{(0)}(worst) - |X_2^{(0)}(1)|) = \\ &= X_2^{(0)}(1) + 0{,}92 \cdot (X_2^{(0)}(4) - |X_2^{(0)}(1)|) - 0{,}49 \cdot (X_2^{(0)}(3) - |X_2^{(0)}(1)|) = \\ &= 18 + 0{,}92 \cdot (7 - |18|) - 0{,}49 \cdot (-6 - |18|) = 19{,}64. \end{aligned}$$

Порівняємо результати обчислень, використавши формулу (2.18), результати занесені у таблицю 2.3.

Таблиця 2.3 – Результат проміжних обчислень на першій ітерації

№	X_1	X_2	$f(X)$
1	-65,19	19,64	4635,466
2	-44,12	45,29	3997,759
3	24,76	0,8	613,698
4	-67,5	13,37	4735,007
5	-70,58	-16,36	5249,186

Таблиця 2.4 – Результат першої ітерації

№	X_1	X_2	$f(X)$	Статус
1	-5	18	349	
2	-44,12	45,29	3997,759	*worst*
3	24,76	0,8	613,698	
4	-8	7	113	*best*
5	12	-18	468	

2. Результати обчислень за формулою (2.19) занесені до таблиці 2.5, а результати другої ітерації наведено у таблиці 2.6. Для прикладу знайдемо нове рішення для агента 1:

$$X_1^2(1) = X_1^{(1)}(1) + 0{,}27 \cdot (X_1^{(1)}(best) - |X_1^{(1)}(1)|) - 0{,}23 \cdot (X_1^{(1)}(worst) - |X_1^{(1)}(1)|) =$$
$$= X_1^{(1)}(1) + 0{,}27 \cdot (X_1^{(1)}(4) - |X_1^{(1)}(1)|) - 0{,}23 \cdot (X_1^{(1)}(2) - |X_1^{(1)}(1)|) =$$
$$= -5 + 0{,}27 \cdot (-8 - |-5|) - 0{,}23 \cdot (-44{,}12 - |-5|) = 2{,}79;$$

$$X_2^2(1) = X_2^{(1)}(1) + 0{,}38 \cdot (X_2^{(1)}(best) - |X_2^{(1)}(1)|) - 0{,}51 \cdot (X_2^{(1)}(worst) - |X_2^{(1)}(1)|) =$$
$$= X_2^{(1)}(1) + 0{,}38 \cdot (X_2^{(1)}(4) - |X_2^{(1)}(1)|) - 0{,}51 \cdot (X_2^{(1)}(2) - |X_2^{(1)}(1)|) =$$
$$= 18 + 0{,}38 \cdot (7 - |18|) - 0{,}51 \cdot (45{,}29 - |18|) = -0{,}01.$$

Таблиця 2.5 – Результат проміжних обчислень на другій ітерації

№	X_1	X_2	$f(X)$
1	2,79	-0,01	7,78
2	-37,897	30,74	2381,13
3	31,76	-19,53	1390,11
4	-0,33	-12,53	157,11
5	-4,49	-36,098	1323,25

Таблиця 2.6 – Результат другої ітерації

№	X_1	X_2	$f(X)$	Статус
1	2,79	-0,01	7,78	*best*
2	-37,897	30,74	2381,13	*worst*
3	24,76	0,8	613,698	
4	-8	7	113	
5	12	-18	468	

Отже, можна помітити, що найкраще значення цільової функції $f(X)$ зменшується зі 113 до 7,78 всього за дві ітерації. Якщо дещо збільшити їх кількість, то можна отримати оптимальне значення $f(X)$. Отже, представлені модифікації дозволять провести ефективніший пошук глобального мінімуму цільової функції.

2.4 Висновки

У даному розділі описано стратегію пошуку екстремуму алгоритмами ройового інтелекту, розглянуто базову та наведено модифіковану математичну модель для розв'язання оптимізаційних задач алгоритмом кажанів. У наступному розділі буде проведено експериментальні дослідження ефективності роботи модифікації.

3 РОЗРОБКА ІНТЕЛЕКТУАЛЬНОГО ПРОГРАМНОГО МОДУЛЯ ТА ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

3.1 Вибір мови програмування

Python представляє популярну високорівневу мову програмування, яка призначена для створення додатків різних типів. Вперше мова Python була анонсована 1991 року голландським розробником Г. ван Россумом. З того часу вона пройшла великий шлях розвитку. У 2000 році було видано версію 2.0, а в 2008 році – версію 3.0 [17]. Python є однією з найпопулярніших мов програмування, які використовуються для розв'язання задач оптимізації. Існує кілька причин, чому Python може бути хорошим вибором для розв'язання таких задач:

1. Простота та зручність використання, що робить дану мову привабливою для широкого кола користувачів. Python має досить простий та зрозумілий синтаксис, що дозволяє розробникам швидко та ефективно писати програмний код.

2. Розширені можливості бібліотек та фреймворків. Наприклад, бібліотека NumPy надає ефективні інструменти для роботи з масивами даних, а бібліотека SciPy містить безліч оптимізаційних алгоритмів та функцій.

3. Хоча Python не є найшвидшою мовою програмування, його продуктивність може бути покращена за допомогою оптимізації коду та використання бібліотек, написаних на C або Fortran.

4. Python має велику та активну спільноту користувачів, які розробляють нові інструменти та бібліотеки для розв'язання різних задач.

5. Python може бути використаний у поєднанні з іншими мовами програмування, такими як C або C++, що може бути корисним при роботі з великими обсягами даних або обчисленнями, що вимагають високої продуктивності.

Python також підтримує функціональне програмування, що може допомогти в розв'язанні складних оптимізаційних задач. Використання Python для розв'язання задач оптимізації може забезпечити більш високий рівень абстракції та модульності, що дозволяє зосередитися на задачі оптимізації, а не на технічних деталях реалізації алгоритмів. Рейтинг мов програмування на кінець минулого року наведено на рисунку 3.1.

Technology	Percentage	Technology	Percentage
Java	22.42%	Golang	1.24%
Python	21.01%	VBA	1.20%
Javascript	14.27%	Powershell	1.09%
C/C++	9.66%	Assembly Language	0.20%
SQL	5.49%	R language	0.25%
PHP	5.16%	Dart language	0.40%
MatLab	4.70%	Ruby language	0.08%
Node js	4.44%	Scala language	0.05%
Typescript	2.61%	Groovy language	0.04%
C#	2.10%	Perl Programming	0.02%
Swift Programming	2.05%	Bash language	0.02%
Kotlin	1.50%	Grand Total	100.00%

Рисунок 3.1 – Рейтинг мов програмування

Отже, Python може бути хорошим вибором для розв'язання задач оптимізації, зокрема, завдяки своїм простоті та зручності використання, наявності широкого спектру бібліотек та фреймворків, великій та активній спільноті користувачів, високій продуктивності та можливості комбінування з іншими мовами програмування.

3.2 Опис програми

Алгоритм на основі поведінки кажанів реалізовано у середовищі Python Anaconda (пакет Spyder). Концептуально інтелектуальний модуль представляє набір підпрограм (рисунок 3.2).

Рисунок 3.2 – Структура програмного забезпечення

Робота з програмою починається з вибору необхідних даних на інтерфейсі. Головне вікно програми складається з пунктів вибору тестового прикладу (Select function); значень параметрів алгоритму (розмір популяції, гучність звукового сигналу, інтенсивність звукового сигналу; мінімальне та максимальне значення частоти звукового сигналу); кількості ітерацій (Max Cycle); розмірності (Param Num D) та меж (Lower Bound, Upper Bound) простору пошуку D; кількості тестових запусків (Runtime); кнопок управління; панелей виведення результатів та поверхні функції (рисунок 3.3).

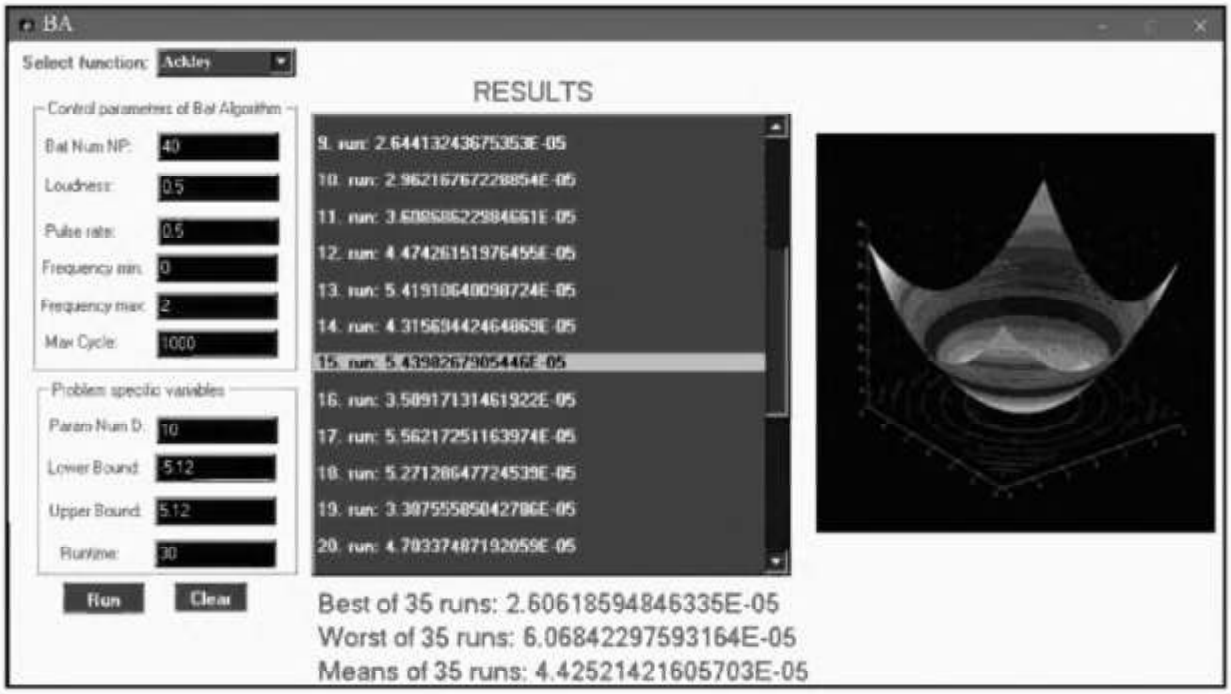

Рисунок 3.3 – Головне вікно користувача програми

При натисканні на кнопку "Run" програма проводить розрахунок і зберігає кінцеві результати: вектор аргументів X^* і значення $f(X^*)$, а також буде показано кінцеву популяцію кажанів та виділено найкращу особину рою. Результати виконання кожної ітерації заносяться в таблицю, яку можна

викликати натисканням кнопки " Details". Для "очищення" робочого поля слід натиснути на кнопку "Clear".

Із програм аналогів можна виділити солвери CPLEX Optimization Studio та GEKKO які мають потужну математичну основу та показали передові результати у розв'язанні низки практичних задач [2]. Розробка включає переваги сучасних рішень (простота, гнучкість, ефективність). Недоліки: необхідність доповнення функціоналу, відсутність коштів для формування команди розробників та проведення досліджень. У таблиці 3.1 наведені основні технічні показники аналога і нового програмного продукту.

Таблиця 3.1 – Основні технічні показники програмного продукту

Показники	Аналог	Розробка	Перевага
Функціональність	немає можливостей корекції коду	можна виконувати гібридизацію методів	перевага у розробки
Надійність	забезпечено необхідний рівень працездатності	забезпечено необхідний рівень працездатності	подібні
Простота	зручний графічний інтерфейс	графічний інтерфейс + консоль	перевага у аналога
Економія ресурсів	швидкодія, можливість роботи з графікою під час розрахунків	швидкодія, можливість задавати критерій зупинки	подібні
Сумісність	пропрієтарне ПЗ	вільне розповсюдження	перевага у розробки

Документація до програмного продукту може бути розроблена інженерами-програмістами та менеджерами за короткий термін. У даній розробці можуть бути зацікавлені компанії, які займаються науковою діяльністю, аналізом даних, проєктуванням систем та мереж тощо. Потенційним замовником продукту є IT компанії. У подальшому планується виконати оптимізацію роботи програми.

3.3 Результати експериментів

Для проведення обчислювальних експериментів вибрали дві тестові функцій [24]:

1. Функція Еклі (рисунок 3.4). Функція зі складним рельєфом. Вважається складною для оптимізації, оскільки існує множина локальних мінімумів та один глобальний мінімум $f(X_i^*) = f(0,\ldots,0) = 0$.

$$f(x_1,\ldots,x_n) = 20 + e - 20 \cdot \exp\left(\frac{1}{5} \cdot \sqrt{\frac{1}{n}\sum_{j=1}^{n} x_j^2}\right) - \\ - \exp\left(\frac{1}{n} \cdot \sum_{j=1}^{n} \cos(2\pi x_j)\right), x_i \in [-20;20]. \tag{3.1}$$

2. Функція Растрігіна (рисунок 3.5). Функція зі складним рельєфом. Вона грунтується на функції "еліптичний параболоїд" з додаванням косинусної модуляції для створення багатьох локальних мінімумів, розміщених регулярно. Глобальний мінімум функції $f(X_i^*) = f(0,\ldots,0) = 0$.

$$f(x_1,...,x_n) = 10 \cdot n + \sum_{j=1}^{n}(x_j^2 - 10\cos(2\pi x_j)),\ x_j \in [-5; 5]. \qquad (3.2)$$

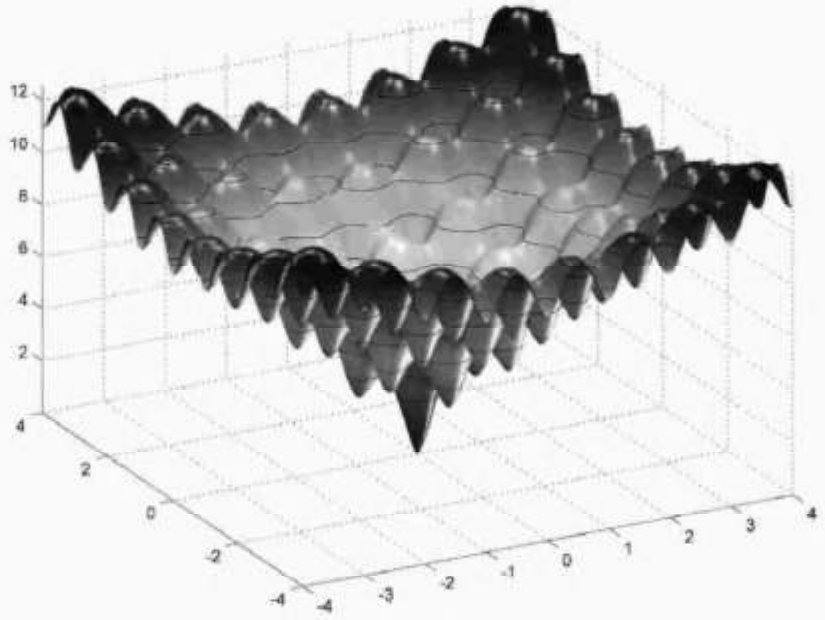

Рисунок 3.4 – Графічне відображення функції Еклі

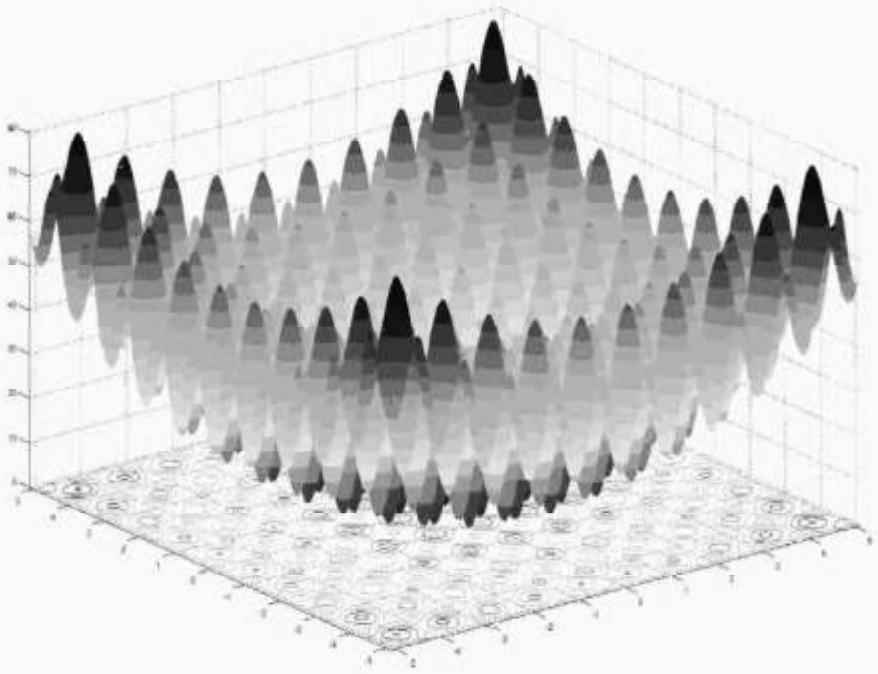

Рисунок 3.5 – Графічне відображення функції Растрігіна

Необхідно проаналізувати вплив зміни параметрів на ефективність розв'язання задачі оптимізації. Досліджено залежність ефективності роботи модифікованого алгоритму від значень гучності $A_i^{(0)}$ і інтенсивності $r_i^{(0)}$ звукового сигналу. У першій серії експериментів прийняли $r_i^{(0)} = 0,5$, а значення гучності змінювали в діапазоні від 0,1 до 1. У наступних експериментах вважали $A_i^{(0)} = 0,5$, а інтенсивність сигналу також змінювали від 0,1 до 1. Рухалися з кроком 0,1 при розмірності простору пошуку $n = 10$. У результаті експериментів визначено, що найкращі значення будуть, коли $A_i^{(0)} = 0,1...0,3$; $r_i^{(0)} = 0,4...0,5$.

Далі необхідно проаналізувати ефективність алгоритму у ході розв'язання задач оптимізації. Важливо, щоб час роботи алгоритма був поліноміальним. Вибрано такий набір параметрів алгоритму:

- $limit = 3500$ ітерацій;
- $|S| = 35$ кажанів;
- частота $\omega = [\omega_{\min}, \omega_{\max}] = [0, 2]$;
- гучність сигналу $A_0 = 0,35$;
- інтенсивність сигналу $r_0 = 0,4$;
- константи $\alpha = \gamma = 0,9$;
- $p = 35$ запусків програми.

Визначалися найгірше f^{worst}, середнє f^{mean} та найкраще f^{best} значення екстремуму за p запусків алгоритму. Отримані результати наведені в таблиці 3.2 та 3.3 для базового та модифікованого алгоритмів.

Таблиця 3.2 – Результати обчислень для першої тестової функції

Розмірність простору	Алгоритм	f^{best}	f^{mean}	f^{worst}
5	Базовий	0,5644	2,2463	3,9942
	Модифікований	$2{,}346 \cdot 10^{-8}$	$4{,}532 \cdot 10^{-7}$	$3{,}311 \cdot 10^{-6}$
10	Базовий	2,9763	3,6585	4,1231
	Модифікований	$2{,}715 \cdot 10^{-6}$	$1{,}104 \cdot 10^{-4}$	$3{,}83 \cdot 10^{-2}$
20	Базовий	3,8556	4,4252	5,3262
	Модифікований	$1{,}335 \cdot 10^{-5}$	$5{,}133 \cdot 10^{-2}$	0,8710

Таблиця 3.3 – Результати обчислень для другої тестової функції

Розмірність простору	Алгоритм	f^{best}	f^{mean}	f^{worst}
5	Базовий	0,384	3,457	7,245
	Модифікований	$3{,}532 \cdot 10^{-2}$	1,364	3,536
10	Базовий	13,129	25,782	33,531
	Модифікований	0,653	3,766	5,159
20	Базовий	53,257	97,271	145,673
	Модифікований	11,970	14,632	25,953

Можна помітити, що модифікований алгоритм перевершує базовий у всіх виконаних дослідах. Приклад кривої збіжності алгоритму для функції 1 наведено на рисунку 3.6 для базового та модифікованого алгоритмів для розмірності простору пошуку $n = 10$ та параметрів $A_i^{(0)} = 0{,}35;\ r_i^{(0)} = 0{,}4$.

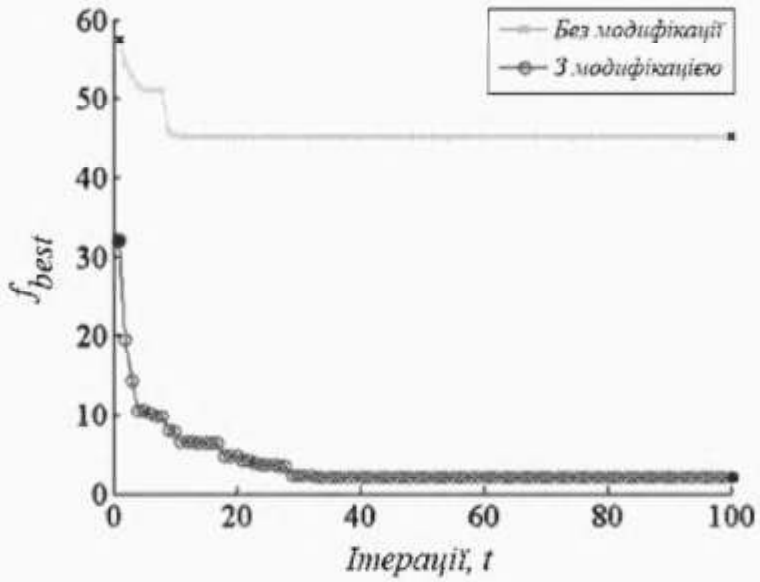

Рисунок 3.6 – Приклад кривої збіжності

Розглянемо інженерну задачу проєктування параметрів зубчастої передачі (рисунок 3.7). Передавальне відношення визначається як відношення кутової швидкості вихідного валу до швидкості вхідного валу. Даний коефіцієнт має бути максимально близьким до $GR = 1/6{,}931$. Кількість зубців зубчастих коліс $X = (x_1, x_2, x_3, x_4)^T$ у кожній передачі має бути цілим числом в межах від 12 до 60 [25–28].

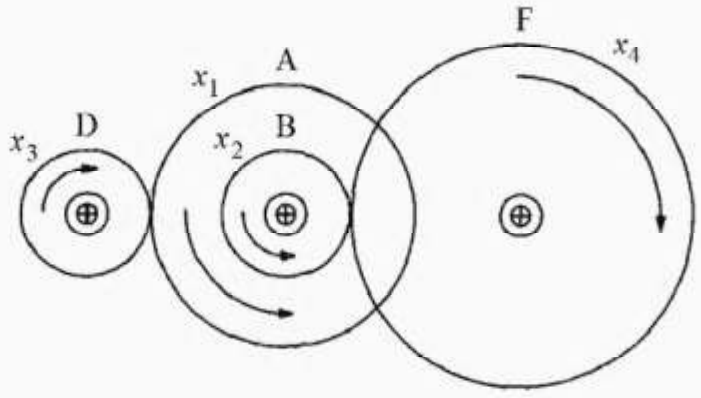

Рисунок 3.7 – Схема зубчастої передачі

Математично дану задачу можна описати так:

$$f = \left(\frac{1}{6{,}931} - \frac{x_1 \cdot x_2}{x_3 \cdot x_4} \right)^2, \tag{3.3}$$

$$f_{\min} = 2.700857 \cdot 10^{-12},\ x_i - цілі\ числа, 12 \le x_i \le 60,\ i = \overline{1,4}.$$

Порівняння результатів роботи різних авторів показано в таблиці 3.4.

Таблиця 3.4 – Порівняння алгоритмів оптимізації (задача 1)

Автори	Розробка	f^{best}
К. Деб, М. Гоял [27]	Комбінований генетичний алгоритм	$1{,}441753 \cdot 10^{-9}$
Б. Каннан, С. Крамер [28]	Метод невизначених множників Лагранжа	$2{,}124644 \cdot 10^{-8}$
Загальна стратегія	Повний перебір	$2{,}700857 \cdot 10^{-12}$
Дана робота	Модифікований алгоритм кажанів	$2{,}700857 \cdot 10^{-12}$

Запропонований алгоритм потрапляє у глобальний оптимум, перевершує низку інших, стабільний, не потребує громіздких обчислень, але необхідно вдало вибрати його параметри.

Розглянемо інженерну задачу оптимізації конструкції циліндричної посудини високого тиску [1, 25, 26, 29–33]. Цільова функція $f(X) = f(x_1, x_2, x_3, x_4)$ визначає вартість витрат на матеріали.

Задача нелінійної оптимізації формулюється наступним чином:

$$f(X) = 0,6224 \cdot x_1 \cdot x_3 \cdot x_4 + 1,7781 \cdot x_2 \cdot x_3^2 + \\ + 3,1661 \cdot x_1^2 \cdot x_4 + 19,84 \cdot x_1^2 \cdot x_3 \to \min, \tag{3.4}$$

з системою умов-нерівностей:

$$\begin{cases} g_1(X) = -x_1 + 0,0193 \cdot x_3 \le 0; \\ g_2(X) = -x_2 + 0,00954 \cdot x_3 \le 0; \\ g_3(X) = -\pi \cdot x_3^2 \cdot x_4 - \frac{4}{3} \cdot \pi \cdot x_3^3 + 1296000 \le 0; \\ g_4(X) = x_4 - 240 \le 0; \\ 1 \times 0,0625 \le x_1, x_2 \le 99 \times 0,0625; \\ 10 \le x_3, x_4 \le 200. \end{cases} \tag{3.5}$$

Дизайн конструкції посудини, зображеної на рисунку 3.8, визначається наступними параметрами: товщина стінки циліндра $T_S = x_1$, товщина сферичної голівки $T_h = x_2$, внутрішній радіус циліндричної оболонки $R = x_3$, довжина циліндричної частини $L = x_4$. Слід зазначити, що змінні x_1 і x_2 є дискретними величинами, які повинні бути кратними товщині листа сталі (0,0625).

Для знаходження безумовного екстремуму задачі умовної оптимізації необхідно перетворити задану функцію. Для цього застосовано метод штрафів, ідея якого полягає в отриманні нової цільової функції $F(X)$, $X = (x_1, ..., x_n)^T$ для області пошуку допустимих рішень D розмірності n за

рахунок введення у цільову функцію $f(X)$ вибраної спеціальним чином функції штрафу $P(X)$.

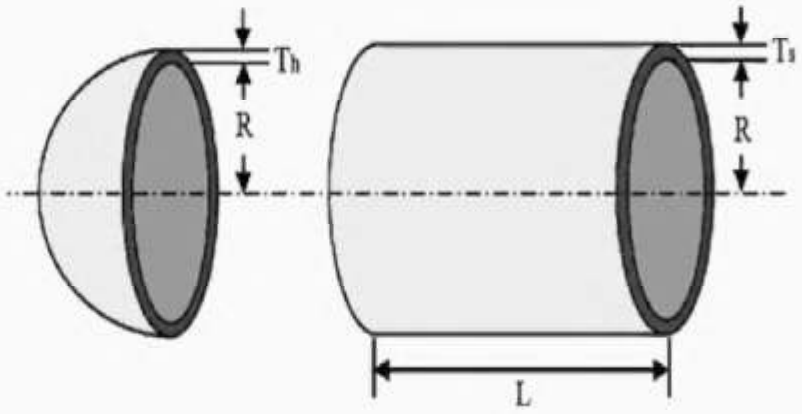

Рисунок 3.8 – Конструкція посудини високого тиску

Для розв'язання представленої задачі використовуємо метод штрафів в адитивній формі [12]:

$$F(X) = f(X) + P(X) = f(X) + \sum_{k=1}^{4} \mu_k \cdot g_k^2(X) \cdot h_k \to \min, \qquad (3.6)$$

де μ_k $(k = \overline{1,4})$ – константи штрафу;

$$h_k = \begin{cases} 1, \text{ if } g_k(X) > 0; \\ 0, \text{ if } g_k(X) \le 0. \end{cases} \qquad (3.7)$$

Пошуком оптимального розв'язку даної задачі займалися багато дослідників, застосовуючи різноманітні методи. Порівняння результатів показано в таблиці 3.5.

Таблиця 3.5 – Порівняння алгоритмів оптимізації (задача 2)

Автори	Розробка	X	f^{best}
А. Кавех, С. Талатахарі (2010) [28]	Модифікований неперервний алгоритм мурашиної колонії	0.8125 0.4375 42.0983 176.6377	6059.7258
Б. Аккей, Д. Карабога (2012) [30]	Алгоритм штучної бджолиної колонії	0.8125 0.4375 42.0984 176.6365	6059.7143
С. Мірджалілі, С.М. Мірджалілі, А. Льюїс (2014) [31]	Оптимізатор на основі поведінки сірих вовків	0.8125 0.4345 42.0891 176.7587	6051.5639
Я. Белкоурчія, Л. Азрар, Е.-С. Зеріаб (2019) [32]	Гібридний алгоритм (генетичний алгоритм + рій часток + послідовне квадратичне програмування)	0.8809 0.4337 45.6342 137.2499	5798.7989
С. Талатахарі, М. Азізі, М. Толоуей та інші (2021) [33]	Алгоритм на основі формування гратки кристалічної структури	0.7781 0.3846 40.3196 200.0000	5885.3313
Дана робота	Модифікований алгоритм кажанів	0.7782 0.3830 40.3207 199.9842	5880.7115

Модифікований алгоритм кажанів отримав наступне рішення: $f(0.7782, 0.3830, 40.3207, 199.9842) = 5880.7115$. Слід зазначити, що на сьогоднішній день відомі результати з кращим рішенням, наприклад, знайдені спеціалізованим гібридним алгоритмом, який застосовує генетичний алгоритм, рій часток та послідовне квадратичне програмування в загальній пошуковій стратегії оптимізації.

У ході обчислювальних експериментів було виявлено вплив основних параметрів алгоритму, конфігурація яких дозволяє легко підлаштувати алгоритм для вирішення різних оптимізаційних задач. Результати експериментів показали, що модифікований алгоритм кажанів більш ефективний для розв'язання задач глобальної оптимізації у порівнянні з базовим варіантом. Він характеризується більш високою швидкістю збіжності і точністю знаходження оптимального значення. Недоліками є більший час виконання ітерації та складність підбору параметрів. Оптимальні значення параметрів гучності й інтенсивності сигналу повинні вибиратися під кожну конкретну задачу. Подальші дослідження необхідно направити на аналіз модифікацій алгоритму кажанів та розробку алгоритму метаоптимізації, його адаптацію до різноманітних задач оптимізації.

Отже, представлений у роботі алгоритм можна застосувати для розв'язання різноманітних комплексних оптимізаційних задач у різних галузях науки та техніки.

3.4 Висновки

У даному розділі описано принципи роботи розробленого програмного забезпечення. Програма дозволяє керувати параметрами модифікованого методу й аналізувати ефективність його роботи. Визначено, що модифікований метод кажанів стабільний і збігається до оптимуму, якщо вдало вибрати ключові параметри. Метод показав свою конкурентоспроможність і можливість до подальшого розвитку.

4 ЕКОНОМІЧНИЙ РОЗДІЛ

4.1 Технологічний аудит модифікованого алгоритму кажанів

Для встановлення комерційного потенціалу розробленого нами інтелектуального модуля для розв'язання задач оптимального проєктування конструкцій було запрошено 3-ох відомих експертів – к.т.н., професора Папінова В. М., к.т.н., доцента Кривогубченко С. Г. та і к.т.н., доцента Овчинникова К. В. Визначення комерційного потенціалу розробленого інтелектуального модуля для розв'язання задач оптимального проєктування конструкцій було здійснено за критеріями, наведеними в таблиці 4.1.

Таблиця 4.1 – Рекомендовані критерії оцінювання рівня комерційного потенціалу будь-якої розробки і їх бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів

Продовження таблиці 4.1

	0	1	2	3	4
			Ринкові перспективи		
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
			Практична здійсненність		
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій > 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій > 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років

Продовження таблиці 4.1

	0	1	2	3	4
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Для встановлення комерційного потенціалу розробленого нами інтелектуального модуля скористуємося рекомендаціями, які наведено в таблиці 4.2 [34, 35]. Запрошені експерти оцінили розроблений нами інтелектуальний модуль для розв'язання задач оптимального проєктування конструкцій у таблиці 4.3.

Таблиця 4.2 – Рівні комерційного потенціалу будь-якої наукової розробки

Середньоарифметична сума балів $\overline{СБ}$, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 – 10	Низький
11 – 20	Нижче середнього
21 – 30	Середній
31 – 40	Вище середнього
41 – 48	Високий

Оскільки середньоарифметична сума балів, що їх виставили експерти, складає 44,67 бали, то це означає, що розроблений нами інтелектуальний модуль має рівень комерційного потенціалу, який вважається "високим". Це

пояснюється тим, що запропонована нами модифікація ройового алгоритму кажанів, особливістю якої є математична модель міграційного процесу зграї, дозволяє суттєво підвищити точність розв'язання задачі оптимізації функцій.

Таблиця 4.3 – Результати технологічного аудиту розробленого інтелектуального модуля (за шкалою оцінювання 0-1-2-3-4)

Критерії	Прізвище, ініціали експертів		
	Папінов В. М.	Кривогубченко С. Г.	Овчинников К. В.
	Бали, що їх виставили експерти:		
1	4	4	3
2	4	4	4
3	3	4	4
4	4	4	4
5	3	3	4
6	4	4	3
7	3	4	4
8	4	3	3
9	4	4	3
10	4	4	4
11	4	4	3
12	4	4	4
Сума балів	$СБ_1 = 45$	$СБ_2 = 46$	$СБ_3 = 43$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{3} = \frac{45+46+43}{3} = \frac{134}{3} = 44,67$		

4.2 Розрахунок витрат на розробку та проведення досліджень

При розробленні інтелектуального модуля нами було витрачено:

1) Основна заробітна плата $З_о$ розробників, яка визначається за формулою (таблиця 4.4):

$$З_о = \frac{М}{Т_р} \cdot t \text{ (грн.)}, \quad (4.1)$$

де М – місячний посадовий оклад розробника, М = (6700…22600) грн./місяць; $Т_р$ – кількість робочих днів в місяці, $Т_р$ = 24 дні; t – кількість днів роботи розробників.

Таблиця 5.4 – Основна заробітна плата розробників (виконавців)

Найменування посади виконавця	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Кількість днів роботи	Витрати на оплату праці, грн.
1. Науковий керівник магістерської роботи	20400	850	20 годин	≈ 2833,33
2. Магістрант-студент-виконавець	6700	279,17	81	≈ 22612,77
3. Консультант з економічної частини	18200	758,33	1,5 години	≈ 189,58 (при 6-годинному робочому дні)
Загалом				$З_о$ = 25635,68 грн ≈ 25636 грн

Б) Додаткова заробітна плата $З_д$ розробників (виконавців) розраховується як (10…12)% від величини їх основної заробітної плати, тобто:

$$З_д = \alpha \cdot З_о = (0{,}1...0{,}12) \cdot З_о. \quad (4.2)$$

Приймемо, що $\alpha = 0{,}114$. Тоді для нашого випадку отримаємо:

$$Зд = 0{,}111 \times 25636 = 2845{,}60 \approx 2846 \text{ грн.}$$

В) Нарахування на заробітну плату $НЗП_{зп}$ розробників (дослідників) розраховуються за формулою:

$$НЗП_{зп} = (З_о + З_д) \cdot \frac{\beta}{100}, \quad (4.3)$$

де β – ставка обов'язкового єдиного внеску на державне соціальне страхування у %, $\beta = 22\%$.

Тоді:

$$НЗН_{зп} = (25636 + 2846) \times 0{,}22 = 6266{,}04 \approx 6266 \text{ грн.}$$

Г) Амортизація основних засобів А, які використовувались під час виконання цієї роботи (таблиця 4.5):

$$А = \frac{Ц \cdot Н_а}{100} \cdot \frac{Т}{12} \text{ (грн.)}, \quad (4.4)$$

де Ц – загальна балансова вартість основних засобів, грн.; $Н_а$ – річна норма амортизаційних відрахувань, $Н_а$ = (2,5...25)%; Т – термін використання основних засобів, місяці.

Таблиця 4.5 – Розрахунок амортизаційних відрахувань

Найменування обладнання, приміщень тощо	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань, грн.
1. Комп'ютерна техніка, обладнання тощо	72700	25	3,3 (при 70% використанні)	3498,68 ≈ 3499
2. Приміщення університету, кафедри	43100	2,5	3,3 (при 50% використанні)	148,15 ≈ 149
Всього				А = 3648 грн.

Д) Витрати на матеріали М розраховуються за формулою:

$$М = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i - \sum_{1}^{n} В_i \cdot Ц_в \text{ (грн.)}, \quad (4.5)$$

де $Н_i$ – витрати матеріалу *i*-го найменування, кг; $Ц_i$ – вартість матеріалу *i*-го найменування; $К_i$ – коефіцієнт транспортних витрат, $К_i = (1{,}1 \ldots 1{,}15)$; $В_i$ – маса відходів матеріалу *i*-го найменування; $Ц_в$ – ціна відходів матеріалу *i*-го найменування; n – кількість видів матеріалів.

Е) Витрати на комплектуючі К розраховуються за формулою:

$$К = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i \text{ (грн.)}, \quad (4.6)$$

де $Н_i$ – кількість комплектуючих *i*-го виду, шт.; $Ц_i$ – ціна комплектуючих *i*-го виду; $К_i$ – коефіцієнт транспортних витрат, $К_i = (1{,}1 \ldots 1{,}15)$; n – кількість видів комплектуючих.

Під час виконання роботи загальні витрати на матеріали та комплектуючі склали приблизно 2650 грн.

Ж) Витрати на силову електроенергію $В_е$ розраховуються за формулою:

$$В_е = \frac{В \cdot П \cdot Ф \cdot К_п}{К_д}, \quad (4.7)$$

де В – вартість 1 кВт-год. електроенергії, в 2023 р. В ≈ 4,5 грн/кВт; П – установлена потужність обладнання, П = 1,02 кВт; Ф – фактична кількість годин роботи обладнання, Ф = 312 годин; $К_п$ – коефіцієнт використання потужності, $К_п < 1 = 0,82$; $К_д$ – коефіцієнт корисної дії, $К_д = 0,73$.

Тоді витрати на силову електроенергію будуть дорівнювати:

$$В_е = \frac{В \cdot П \cdot Ф \cdot К_п}{К_д} = \frac{4,5 \cdot 1,02 \cdot 312 \cdot 0,82}{0,73} = 1608,63 \approx 1609 \text{ грн.}$$

З) Інші витрати $В_{інш}$ можна прийняти як (50…300)% від основної заробітної плати розробників, тобто:

$$В_{інш} = (0,5....3) \times З_о. \quad (4.8)$$

Для нашого випадку отримаємо:

$$В_{інш} = 1,6 \times 25636 = 41017,6 \approx 41018 \text{ грн.}$$

К) Сума всіх попередніх статей витрат складає витрати на виконання роботи безпосередньо розробником-магістрантом – В:

$$В = 25636 + 2846 + 6266 + 3648 + 2650 + 1609 + 41018 = 83673 \text{ грн.}$$

Л) Загальні витрати на розроблення інтелектуального модуля $В_{заг}$ розраховуються за формулою:

$$В_{заг} = \frac{В}{\beta}, \quad (4.9)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання цієї роботи.

Оскільки робота знаходиться у ході доопрацювання документації та рефакторингу програмного коду, то можна прийняти, що, $\beta \approx 0{,}65$ [34, 35]. Тоді:

$$B_{заг} = \frac{92687}{0{,}65} = 128727{,}69 \text{ грн. або приблизно 129 тисяч грн.}$$

Тобто прогнозовані загальні витрати на розроблення інтелектуального модуля для розв'язання задач оптимального проєктування конструкцій становлять приблизно 129 тисяч грн.

4.3 Прогнозування комерційного ефекту від можливої комерціалізації розробки

Економічний ефект від впровадження та можливої комерціалізації розробленого інтелектуального модуля пояснюється його значно кращими функціональними можливостями та такими ключовими перевагами, як простота, гнучкість, ефективність, можливість використання в сфері освіти.

Тому нашу розробку можна реалізовувати на ринку дещо дорожче, ніж аналогічні (але гірші) за функціями подібні розробки. Якщо подібні інтелектуальні моделі у 2022-2023 роках коштували на ринку приблизно 150 тисяч грн, то нашу розробку можна буде реалізовувати приблизно за 180 тисяч грн або на 30 тисяч грн дорожче.

Аналіз місткості ринку показує, що на сьогодні в Україні попит на подібний програмний продукт, може бути великим. Цей попит формує низка компаній, які займаються науковою діяльністю, інтелектуальною обробкою контенту, системами автоматичного управління, проектуванням зразків техніки тощо. Розробка також актуальна в сфері освіти.

Тому можна очікувати стрімке зростання попиту на нашу розробку принаймні впродовж 3-ох років після її впровадження (до появи нових, більш ефективних розробок). Тобто, якщо наша розробка буде впроваджена з 1 січня 2024 року, то її результати будуть виявлятися впродовж 2024-го, 2025-го та 2026-го років. Прогноз зростання попиту на нашу розробку складає по роках: 2023 рік – базовий попит 30 шт.; 2024 р. – приблизно +20 шт. до базового року; 2025 р. – +30 шт. до базового року; 2026 р. – +40 шт. до базового року.

Можливе збільшення чистого прибутку $\Delta П_i$, що його може отримати потенційний інвестор від комерціалізації нашої розробки на ринок, становитиме:

$$\Delta П_i = \sum_{1}^{n} (\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\upsilon}{100}), \qquad (4.10)$$

де $\Delta Ц_o$ – покращення основного якісного показника від впровадження результатів нашої розробки у цьому році, $\Delta Ц_o$ = (180 – 150) = + 30 тисяч грн; N – основний кількісний показник, який визначає обсяг діяльності у році до впровадження результатів розробки; N = 30 шт.; ΔN – покращення основного кількісного показника від впровадження результатів розробки; $Ц_o$ – основний

якісний показник (тобто ціна), який визначає обсяг діяльності у році після впровадження результатів розробки, $Ц_о$= 180 тисяч грн; n – кількість років, впродовж яких очікується отримання позитивних результатів від впровадження розробки; для нашого випадку n = 3; λ – коефіцієнт, який враховує сплату податку на додану вартість; $\lambda = 0{,}8333$; ρ – коефіцієнт, який враховує рентабельність продукту, ρ= (0,2...0,5), нехай ρ= 0,5; υ – ставка податку на прибуток, υ = 18%.

Тоді можливе зростання чистого прибутку $\Delta П_1$ для потенційного інвестора складе:

$$\Delta П_1 = \left(30 \cdot 30 + 180 \cdot 20\right) \cdot 0{,}8333 \cdot 0{,}5 \cdot \left(1 - \frac{18}{100}\right) \approx 1537 \text{ тис. грн.}$$

$$\Delta П_2 = \left(30 \cdot 30 + 180 \cdot 30\right) \cdot 0{,}8333 \cdot 0{,}5 \cdot \left(1 - \frac{18}{100}\right) \approx 2152 \text{ тис. грн.}$$

$$\Delta П_3 = \left(30 \cdot 30 + 180 \cdot 40\right) \cdot 0{,}8333 \cdot 0{,}5 \cdot \left(1 - \frac{18}{100}\right) \approx 2767 \text{ тис. грн.}$$

Приведена вартість зростання всіх чистих прибутків від можливого впровадження нашої розробки становитиме:

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (4.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, впродовж яких виявляються результати виконаної та впровадженої роботи, грн; т – період часу, впродовж якого виявляються результати впровадженої роботи, т = 3 роки; τ – ставка дисконтування, τ = 0,10 (10%); t – період часу від

моменту початку розроблення нашого інтелектуального модуля до моменту отримання можливих чистих прибутків потенційним інвестором.

Тоді приведена вартість зростання всіх можливих чистих прибутків ПП, що їх може отримати потенційний інвестор від комерціалізації нашої розробки, складе:

$$ПП = \frac{1537}{(1+0,1)^2} + \frac{2152}{(1+0,1)^3} + \frac{2767}{(1+0,1)^4} \approx 1270 + 1617 + 1890 = 4777 \text{ тисяч грн.}$$

Теперішня вартість інвестицій PV, що повинні бути вкладені для реалізації нашої розробки:

$$PV = (1,0 \ldots 5,0) \times В_{заг}. \qquad (4.12)$$

$$PV = (1,0 \ldots 5,0) \times 129 = 5 \times 129 = 645 \text{ тисяч грн.}$$

Абсолютний ефект від можливих вкладених інвестицій $Е_{абс}$:

$$Е_{абс} = ПП - PV. \qquad (4.13)$$

$$Е_{абс} = 4777 - 645 = 4132 \text{ тисяч грн.}$$

Далі розрахуємо внутрішню дохідність $Е_в$ вкладених інвестицій:

$$Е_в = \sqrt[Т_ж]{1 + \frac{Е_{абс}}{PV}} - 1, \qquad (4.14)$$

де $Т_ж$ – життєвий цикл розробки, $Т_ж = 4$ роки.

Для нашого випадку отримаємо:

$$E_в = \sqrt[4]{1+\frac{4132}{645}} - 1 = \sqrt[4]{1+6,4062} - 1 = \sqrt[4]{7,4062} - 1 = 1,649 - 1 = 0,649 = 64,9\%.$$

Далі визначимо ту мінімальну дохідність, нижче за яку потенційному інвестору не вигідно буде займатися комерціалізацією нашої розробки.

Мінімальна дохідність або мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau_{мін} = d + f, \qquad (4.15)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках, в 2023 році в Україні d = (0,10...0,12); f – показник, що характеризує ризикованість вкладень; f = (0,05...0,30).

Для нашого випадку отримаємо:

$$\tau_{мін} = 0,12 + 0,30 = 0,42 \text{ або } \tau_{мін} = 42\%.$$

Оскільки величина $E_в = 64,9\% > \tau_{міц} = 42\%$, то потенційний інвестор у принципі може бути зацікавлений у комерціалізації розробленого модуля.

Далі розраховуємо термін окупності коштів, вкладених у можливу комерціалізацію розробленого нами програмного продукту:

$$T_{ок} = \frac{1}{E_в}. \qquad (4.16)$$

$$T_{ок} = \frac{1}{0,524} = 1,91 \text{ років} < 3 \text{ років},$$

що свідчить про потенційну доцільність комерціалізації.

Далі проведено моделювання залежності величини внутрішньої дохідності вкладених потенційним інвестором коштів в комерціалізацію розробленого нами інтелектуального модуля від рівня інфляції в країні. Так, якщо рівень інфляції в країні зросте до 20%, то:

$$ПП = \frac{1537}{(1+0,2)^2} + \frac{2152}{(1+0,2)^3} + \frac{2767}{(1+0,2)^4} \approx 1067 + 1245 + 1334 = 3646 \text{ тисяч грн.}$$

Тоді абсолютний ефект від можливого впровадження нашої розробки складе:

$$E_{абс} = 3646 - 645 = 3001 \text{ тисяч грн.}$$

Внутрішня дохідність $E_в$ вкладених інвестицій становитиме:

$$E_в = \sqrt[4]{1 + \frac{3001}{645}} - 1 = \sqrt[4]{1 + 4,6527} - 1 = \sqrt[4]{5,6527} - 1 = 1,542 - 1 = 0,542 = 54,2\%.$$

Оскільки величина $E_в = 54,2\% > \tau_{мін} = 42\%$, то потенційний інвестор також може бути зацікавлений у комерціалізації нашої розробки.

Прийнявши рівень інфляції у 30% отримаємо:

$$ПП = \frac{1537}{(1+0,3)^2} + \frac{2152}{(1+0,3)^3} + \frac{2767}{(1+0,3)^4} \approx 909 + 980 + 969 = 2858 \text{ тисяч грн.}$$

Тоді абсолютний ефект від можливого впровадження нашої розробки складе:

$$E_{абс} = 2858 - 645 = 2213 \text{ тисяч грн.}$$

Внутрішня дохідність $E_в$ вкладених інвестицій становитиме:

$$E_в = \sqrt[4]{1+\frac{2213}{645}} - 1 = \sqrt[4]{1+3,4310} - 1 = \sqrt[4]{4,4310} - 1 = 1,45 - 1 = 0,45 = 45\%.$$

Оскільки величина $E_в = 45\% > \tau_{мін} = 42\%$, то потенційний інвестор у принципі також може бути зацікавлений у комерціалізації нашої розробки.

Прийнявши рівень інфляції у 40% отримаємо:

$$ПП = \frac{1537}{(1+0,4)^2} + \frac{2152}{(1+0,4)^3} + \frac{2767}{(1+0,4)^4} \approx 784 + 784 + 720 = 2288 \text{ тисяч грн.}$$

Тоді абсолютний ефект від можливого впровадження нашої розробки складе:

$$E_{абс} = 2288 - 645 = 1643 \text{ тисяч грн.}$$

Внутрішня дохідність $E_в$ вкладених інвестицій становитиме:

$$E_в = \sqrt[4]{1+\frac{1643}{645}} - 1 = \sqrt[4]{1+2,5472} - 1 = \sqrt[4]{3,5472} - 1 = 1,372 - 1 = 0,372 = 37,2\%.$$

Оскільки величина $E_в = 37,2\% < \tau_{мін} = 42\%$, то потенційний інвестор може бути незацікавлений у комерціалізації нашої розробки, але остаточне рішення щодо цього питання буде прийматися при врахуванні інших обставин (наприклад, шляхом зниження рівня прийнятого ризику з f = 40% до меншої величини або шляхом підняття ціни реалізації нашої розробки тощо). Зроблені розрахунки у вигляді графіків наведено на рисунку 4.1.

Рисунок 4.1 – Моделювання залежності величини внутрішньої дохідності потенційних інвестицій від рівня інфляції в країні (10%, 20%, 30%, 40%)

4.4 Висновки

Аналіз комерційного потенціалу розробки показав, що програмний продукт за своїми характеристиками випереджає можливі аналогічні реалізації і є перспективною розробкою. Він має кращі функціональні показники, а тому є конкурентоспроможним товаром на ринку. Всі задані у технічному завданні прогнозовані результати роботи виконані: витрати на розробку – 129 тис. грн. (не більше 130 тис. грн.); абсолютний ефект від впровадження розробки – 4132 тис. грн. при 10%-інфляції (не менше 4900 тис. грн.); внутрішня норма дохідності інвестицій – 64,9% при 10%-інфляції (не менше 42%); термін окупності інвестицій – 1,91 року (до 3-ох років). Таким чином, основні техніко-економічні показники модифікованого алгоритму виконані.

ВИСНОВКИ

Аналіз показав, що у ході розв'язання різноманітних задач проєктування виникають комплексні задачі оптимізації з великою розмірністю простору пошуку та складністю цільової функції. У цьому випадку застосовують ройовий інтелект, з якого виділяється алгоритм кажанів. Описано математичну модель даного алгоритму та виконано її модифікацію.

Розроблено програмне забезпечення, яке дозволяє керувати параметрами алгоритму та візуалізовувати результати. Тестування ефективності роботи модифікованого алгоритму проведено на декількох тестових функціях та задачах проєктування конструкцій. Визначено, що модифікований алгоритм вимагає вдалого підбору параметрів, але має більш високу швидкість збіжності та точність знаходження оптимуму. Отже, даний алгоритм можна успішно використовувати для розв'язання комплексних задач оптимізації.

Аналіз комерційного потенціалу програмного забезпечення показав, що його розробка є перспективною.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Handbook of Test Problems in Local and Global Optimization / C.A. Floudas, P.M. Pardalos, C.S. Adjiman, W.R. Esposito, Z.H. Gumus, S.T. Harding, J.L. Klepeis, C.A. Meyer. Kluwer Academic Publishers, 1999. 442 p.

2. Bio-Inspired Computation: Where We Stand and What's Next / E. Osaba, D. Molina, X.-S. Yang et al. *Swarm and Evolutionary Computation*. 2019. Vol. 48. P. 220–250.

3. Субботін С.О., Олійник А.О., Олійник О.О. Неітеративні, еволюційні та мультиагентні методи синтезу нечіткологічних і нейромережних моделей. Запоріжжя: ЗНТУ, 2009. 375 с.

4. Yang X.-S. A New Metaheuristic Bat-Inspired Algorithm. *Nature Inspired Cooperative Strategies for Optimization*. 2010. Vol. 284. P. 65–74.

5. Yang X.-S., Gandomi A.H. Bat Algorithm: A Novel Approach for Global Engineering Optimization. *Engineering Computations*. 2012. Vol. 29. P. 464–483.

6. Yilmaz S., Kucuksille E.U. A New Modification Approach on Bat Algorithm for Solving Optimization Problems. *Applied Soft Computing*. 2015. Vol. 28. P. 259–275.

7. Hasançebi O., Teke T., Pekcan O. A Bat-Inspired Algorithm for Structural Optimization. *Computers & Structures*. 2013. Vol. 128. P. 77–90.

8. Розроблення інформаційних технологій розв'язування задач дискретної оптимізації на основі ройового інтелекту / В.В. Литвин, Д.І. Угрин, Р.М. Оливко, Я.В. Боровець. *ТАРП*. 2018. №2. 11 с. URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjx-9HijsmBAxUkIRAIHTfbAEsQFnoECA8QAQ&url

=https%3A%2F%2Fjournals.uran.ua%2Ftarp%2Farticle%2Fdownload%2F150512%2F152355%2F331298&usg=AOvVaw34gYgfNyQTg4U4qnmsdaJA&opi=89978449 (дата звернення 20.09.2023)

9. A Comprehensive Review of Bat Inspired Algorithm: Variants, Applications, and Hybridization / M. Shehab, M.-A. Abu-Hashem, M.K.Y. Shambour et al. Archives of Computational Methods in Engineering, Vol. 30. 2023. P. 765–797.

10. Деякі аспекти метаевристичних алгоритмів оптимізації / В.О. Вовковинський, Ю.Ю. Іванов, С.Г. Кривогубченко та інші. *Молодь в науці: дослідження, проблеми, перспективи*: матер. всеукраїнської науково-практичної конференції. Вінниця: ВНТУ, 2023. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2023/paper/view/17145 (дата звернення 20.09.2023).

11. Григорків В.С., Григорків М.В., Ярошенко О.І. Оптимізаційні методи та моделі. Чернівці: Чернівецький національний університет, 2022. 440 с.

12. Kochenderfer M.J., Wheeler T.A. Algorithms for Optimization. Cambridge: The MIT Press, 2019. 500 p.

13. Попов Ю.Д., Тюптя В.І., Шевченко В.І. Методи оптимізації. К.: КНУ, 2003. 215 с.

14. Нефьодов Ю.М., Балицька Т.Ю. Методи оптимізації в прикладах і задачах. К.: Кондор, 2011. 324 с.

15. Промислові мережі та інтеграційні технології в автоматизованих системах / О.М. Пупена, І.В. Ельперін, Н.М. Луцька та інші. К., 2021. 522 с.

16. Xie J., Zhou Y., Chen H. A Novel Bat Algorithm Based on Differential Operator and L'evy Flights Trajectory. *Computational Intelligence and*

Neuroscience, 2013. URL: https://www.hindawi.com/journals/cin/2013/453812/ (дата звернення 11.10.2023)

17. Python documentation. URL: https://docs.python.org/3/index.html (дата звернення 04.11.23).

18. Bat Algorithm for Constrained Optimization Tasks / A.H. Gandomi, X.-S.Yang, A.H. Alavi, S. Talatahari. *Neural Computing and Applications*. 2013. Vol. 22. P. 1239–1255.

19. A Modified Bat Algorithm for Solving Large-Scale Bound Constrained Global Optimization Problems / W.K. Mashwani, I. Mehmood, A.B. Maharani, I. Koçcak. *Mathematical Problems in Engineering*. 2021. 14 p. URL: https://www.hindawi.com/journals/mpe/2021/6636918/ (дата звернення 10.09.2023)

20. New Modifed Controlled Bat Algorithm for Numerical Optimization Problem / B.W. Haider, A. Hameed, J. Ahmad, K. Nisar et al. *Computers, Materials and Continua*. 2022. Vol. 70. P. 2241–2259.

21. New Directional Bat Algorithm for Continuous Optimization Problems / A. Chakri, R. Khelif, M. Benouaret, X.-S. Yang. *Expert Systems with Applications*. 2017. Vol. 69. P. 159–175.

22. Yang X.-S., Fister I.J., Fister D. A Hybrid Bat Algorithm. *Electrotechnical Review*. 2013. Vol . 80. – № 1. P. 1–7.

23. Wang G.-G., Lu M., Zhao X.-J. An Improved Bat Algorithm with Variable Neighborhood Search for Global Optimization. *IEEE Congress on Evolutionary Computation*. 2016. P. 1773–1778.

24. Optimization Test Problems. Test Functions and Datasets. URL: http://www.sfu.ca/~ssurjano/ackley.html (дата звернення 19.11.2023).

25. Abatari H.D., Abad M.S.S., Seif H. Application of Bat Optimization Algorithm in Optimal Power Flow. *Conference on Electrical Engineering*. 2016. P. 793–798.

26. Numerical Study for Single and Multiple Damage Detection and Localization in Beam-like Structures Using Bat Algorithm / S. Khatir, I. Belaidi, R. Serra, M.A. Wahab et al. *Journal of Vibroengineering*. 2016. Vol. 18. P. 424–432.

27. Deb K., Goyal M. Optimizing Engineering Designs Using a Combined Genetic Search. *Proceedings of the 7th international conference on genetic algorithms*. San Francisco: Morgan Kaufmann Publishers, 1997. P. 521–528.

28. Kannan B., Kramer S.N. An Augmented Lagrange Multiplier Based Method for Mixed Integer Discrete Continuous Optimization and its Applications to Mechanical Design. *Journal of mechanical design*. 1994. Vol. 116. P. 405–411.

29. Kaveh A., Talatahari S. An Improved Ant Colony Optimization for Constrained Engineering Design Problems. *Engineering* with *Computers*. 2010. Vol. 1. P. 155–182.

30. Akay B., Karaboga D. Artificial Bee Colony Algorithm for Large-scale Problems and Engineering Design Optimization. *Journal of intelligent manufacturing*. 2012. Vol. 23. P. 1001–1014.

31. Mirjalili S., Mirjalili S.M., Lewis A. Grey Wolf Optimizer. *Advances in engineering software*. 2014. Vol. 69. P. 46–61.

32. Belkourchia Y., Azrar L., Zeriab E.-S. A Hybrid Optimization Algorithm for Solving Constrained Engineering Design Problems. *V International Conference on Optimization and Applications*. Kenitra (Morocco), 2019. P. 1–7.

33. Crystal Structure Algorithm (CryStAl): A Metaheuristic Optimization Method / S. Talatahari, M. Azizi, M. Tolouei, B. Talatahari, P. Sareh. *IEEE Access.* 2021. Vol. 9. P. 71244–71261.

34. Кавецький В.В., Козловський В.О., Причепа І.В. Економічне обґрунтування інноваційних рішень. Вінниця: ВНТУ, 2016. 113 с.

35. Козловський В.О., Лесько О.Й., Кавецький В.В. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт. Вінниця: ВНТУ, 2021. 42 с.

ДОДАТКИ

Додаток А

(обов'язковий)

Технічне завдання

ЗАТВЕРДЖУЮ

завідувач кафедри АІІТ

д.т.н., проф. Бісікало О. В.

«___»_____ 2023 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на магістерську кваліфікаційну роботу

«Розробка інтелектуального модуля для розв'язання задач оптимального проєктування конструкцій»

08-31.МКР.001.02.000 ТЗ

Керівник роботи:

к.т.н., доц. каф. АІІТ

Іванов Ю. Ю.

«___» _____ 2023 р.

Виконавець:

ст. гр. 1АКІТ-22м

Вовковинський В. О.

«___» _____ 2023 р.

Вінниця ВНТУ – 2023

1. Назва та галузь застосування

У даній роботі розглядається ройовий алгоритм кажанів, який можна використовувати для розв'язання різноманітних задач практично при повній відсутності припущень про характер цільової функції.

2. Підстави для розробки

Розробку здійснювати на підставі наказу по університету № 247 від 18.09.2023 та завдання до магістерської кваліфікаційної роботи, складеного та затвердженого кафедрою «Автоматизації та інтелектуальних інформаційних технологій».

3. Мета та призначення розробки

Метою роботи є підвищення ефективності розв'язання задачі оптимізації функцій, включаючи задачі оптимального проєктування конструкцій, з використанням ройового алгоритму кажанів за рахунок використання нової моделі міграційного процесу для особин зграї.

4. Джерела розробки

1. Yang X.-S., Gandomi A.H. Bat Algorithm: A Novel Approach for Global Engineering Optimization. *Engineering Computations*. 2012. Vol. 29. P. 464–483.

2. Hasançebi O., Teke T., Pekcan O. A Bat-Inspired Algorithm for Structural Optimization. *Computers & Structures*. 2013. Vol. 128. P. 77–90.

3. Yilmaz S., Kucuksille E.U. A New Modification Approach on Bat Algorithm for Solving Optimization Problems. *Applied Soft Computing*. 2015. Vol. 28. P. 259–275.

5. Показники призначення

Розроблено програмне забезпечення, яке дозволяє провести експериментальні дослідження у ході розв'язання задач оптимального проєктування конструкцій.

Основні технічні вимоги та мінімальні системні вимоги до програми: операційна система *Windows*; кількість ядер комп'ютера – не менше 4 (*Core* i5 або *Ryzen* 5); оперативна пам'ять – не менше 8 ГБ; відеопам'ять – 512 Mb; жорсткий диск – 500 ГБ і більше. Вхідні дані: розмірність простору пошуку; параметри алгоритму; набір задач. Результати роботи програми: оптимальні проєктні параметри, екстремум функції.

6. Економічні показники

До економічних показників входять:

– термін окупності – до 3-ох років;

– приведена вартість прибутку за 3 роки – не менше 130 тис. грн.;

– мінімальна дохідність – не менше 42 %;

– інші економічні переваги у порівнянні з аналогами.

7.Стадії розробки

1. Розділ 1 «Аналіз предметної області роботи» має бути виконаний до до 10.10.23.

2. Розділ 2 «Розробка модифікованої математичної моделі ройового алгоритму кажанів» має бути виконаний до 25.10.23.

3. Розділ 3 «Розробка інтелектуального програмного модуля та експериментальні дослідження» має бути виконаний до 10.11.23.

4. Економічний розділ має бути виконаний до 26.11.23.

8. Порядок контролю та приймання

1. Рубіжний контроль провести до 04.12.23.

2. Попередній захист магістерської кваліфікаційної роботи провести до 05.12.23.

3. Захист магістерської кваліфікаційної роботи провести до 19.12.23.

Додаток Б

(обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА ІНТЕЛЕКТУАЛЬНОГО МОДУЛЯ ДЛЯ РОЗВ'ЯЗАННЯ ЗАДАЧ ОПТИМАЛЬНОГО ПРОЄКТУВАННЯ КОНСТРУКЦІЙ

Зав. кафедри АІІТ _____ д-р техн. наук, професор каф. АІІТ Бісікало О. В.

(підпис) (науковий ступінь, вчене звання, ініціали та прізвище)

Керівник роботи _____ канд. техн. наук, доцент каф. АІІТ Іванов Ю. Ю.

(підпис) (науковий ступінь, вчене звання, ініціали та прізвище)

Тех. контроль _____ канд. техн. наук, доцент каф. АІІТ Іванов Ю. Ю.

(підпис) (науковий ступінь, вчене звання, ініціали та прізвище)

Нормоконтроль _____ канд. техн. наук, доцент каф. АІІТ Іванов Ю. Ю.

(підпис) (науковий ступінь, вчене звання, ініціали та прізвище)

Опонент _____ _____

_____

(підпис) (науковий ступінь, вчене звання, ініціали та прізвище)

Студент гр. 1АКІТ-22м _____ Вовковинський В. О.

(підпис) (ініціали та прізвище)

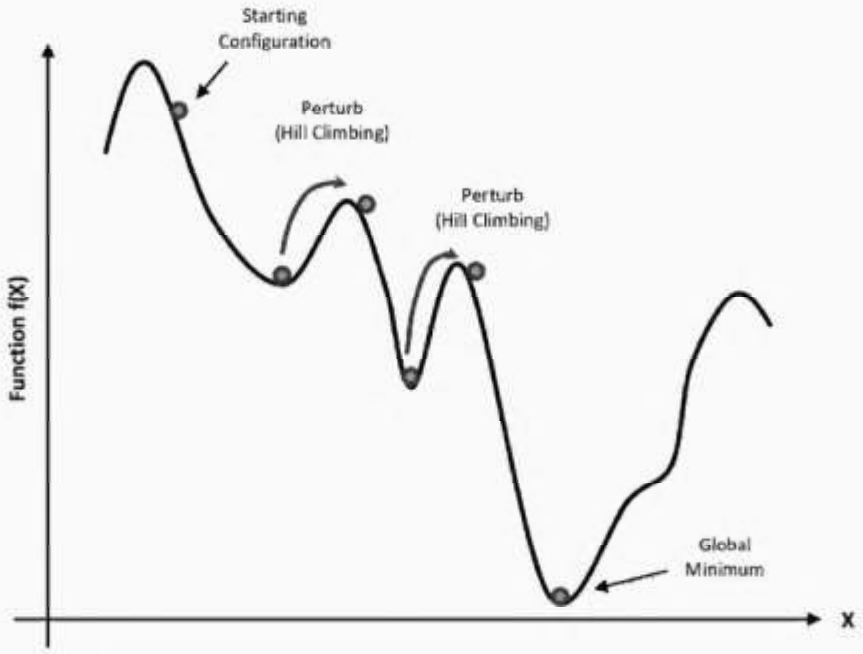

Рисунок Б.1 – Принцип роботи особини рою

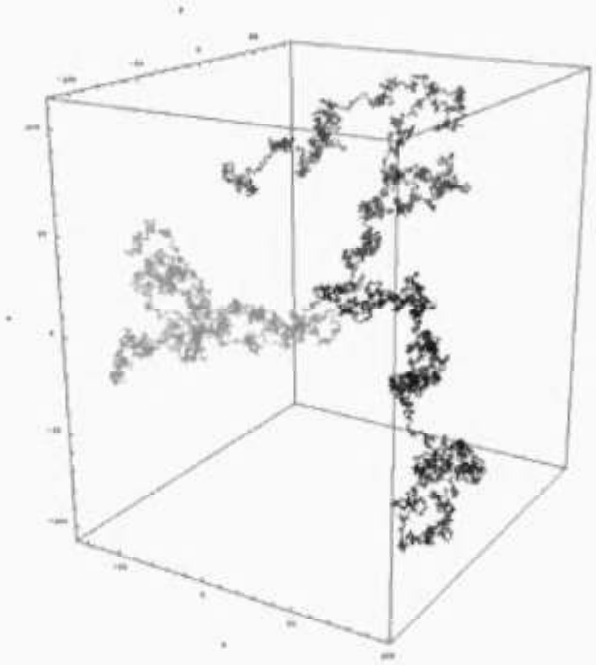

Рисунок Б.2 – Принцип випадкового блукання

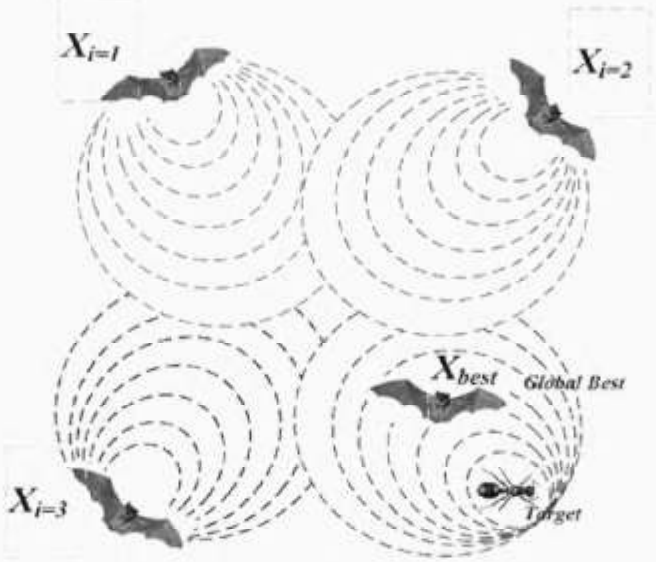

Рисунок Б.3 – Принцип роботи алгоритму кажанів

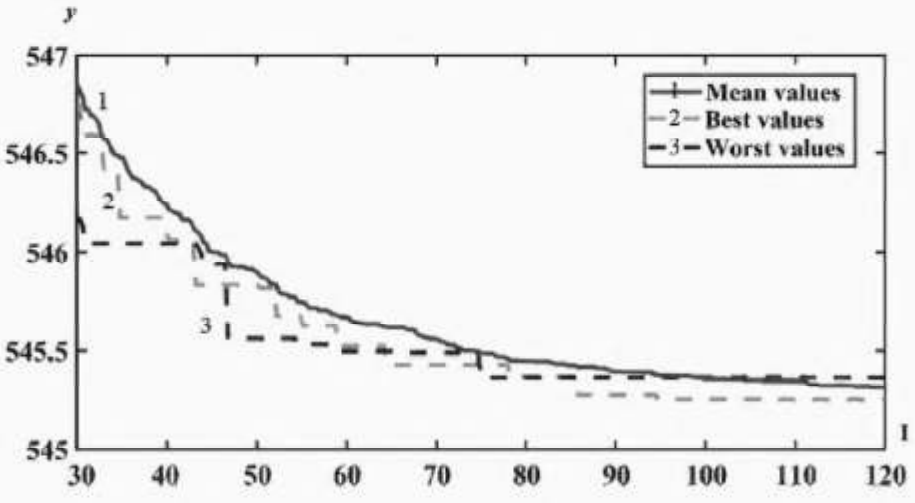

Рисунок Б.4 – Приклад кривої збіжності

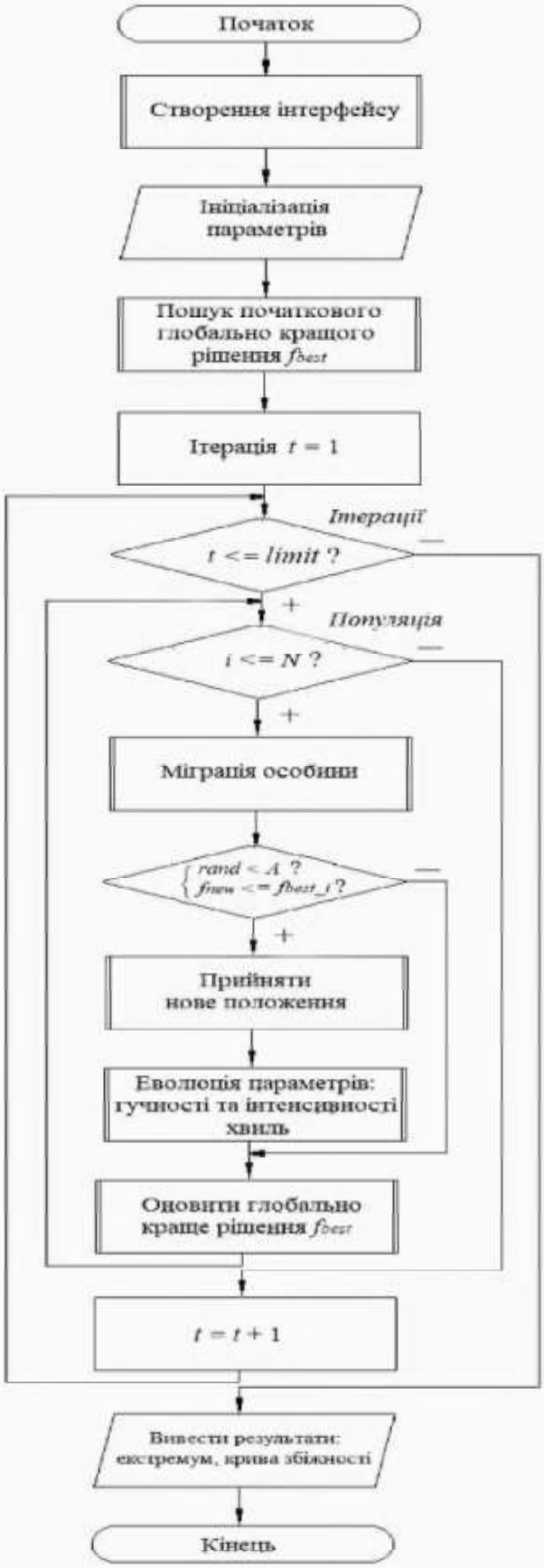

Рисунок Б.5 – Схема програми

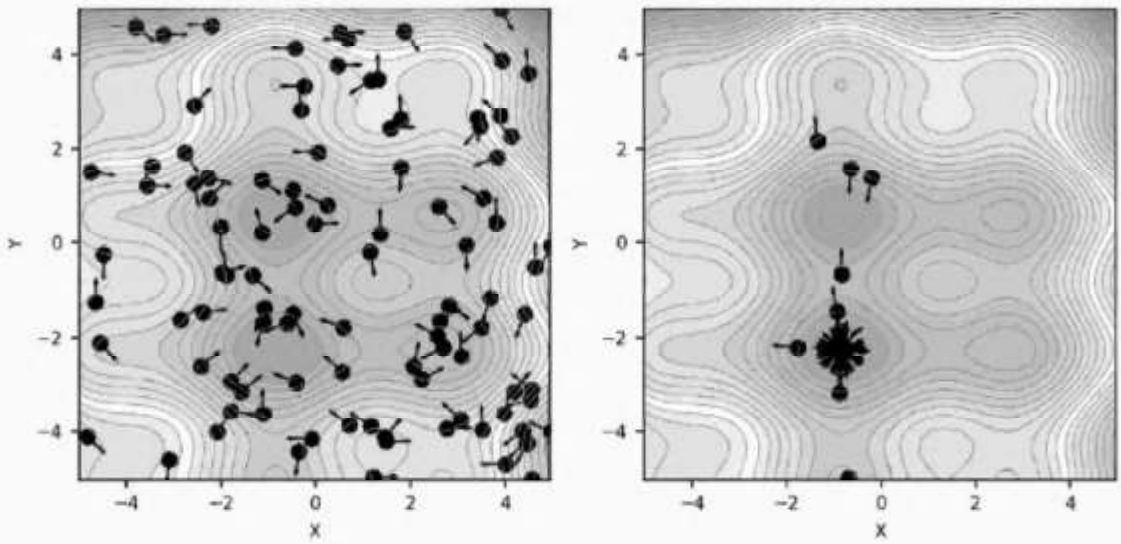

Рисунок Б.6 – Приклад роботи програми

Додаток В

(обов'язковий)

Лістинг програми

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
import random
import numpy as np

class BatAlgorithm():
        def __init__(self, D, NP, N_Gen, A, r, Qmin, Qmax,
        Lower, Upper, function):
        self.D = D
        self.NP = NP
        self.N_Gen = N_Gen
        self.A = A
        self.r = r
        self.Qmin = Qmin
        self.Qmax = Qmax
        self.Lower = Lower
        self.Upper = Upper
        self.f_min = 0.0
        self.Lb = [0] * self.D
        self.Ub = [0] * self.D
        self.Q = [0] * self.NP
        self.v = [[0 for i in range(self.D)] for j in
        range(self.NP)]
        self.Sol = [[0 for i in range(self.D)] for j in
        range(self.NP)]
        self.Fitness = [0] * self.NP
        self.best = [0] * self.D
        self.Fun = function

    def best_bat(self):
        i = 0
        j = 0
        for i in range(self.NP):
            if self.Fitness[i] < self.Fitness[j]:
                j = i
        for i in range(self.D):
            self.best[i] = self.Sol[j][i]
        self.f_min = self.Fitness[j]

    def init_bat(self):
        for i in range(self.D):
            self.Lb[i] = self.Lower
            self.Ub[i] = self.Upper
```

```
        for i in range(self.NP):
            self.Q[i] = 0
            for j in range(self.D):
                rnd = np.random.uniform(0, 1)
                self.v[i][j] = 0.0
                self.Sol[i][j] = self.Lb[j] + (self.Ub[j] -
                self.Lb[j]) * rnd
            self.Fitness[i] = self.Fun(self.D, self.Sol[i])
        self.best_bat()

    def simplebounds(self, val, lower, upper):
        if val < lower:
            val = lower
        if val > upper:
            val = upper
        return val

    def move_bat(self):
        S = [[0.0 for i in range(self.D)] for j in
        range(self.NP)]
        self.init_bat()
        for t in range(self.N_Gen):
            for i in range(self.NP):
                rnd = np.random.uniform(0, 1)
                self.Q[i] = self.Qmin + (self.Qmax -
                self.Qmin) * rnd
                for j in range(self.D):
                    self.v[i][j] = self.v[i][j] +
                    (self.Sol[i][j] - self.best[j]) *
                    self.Q[i]
                    S[i][j] = self.Sol[i][j] + self.v[i][j]
                    S[i][j] = self.simplebounds(S[i][j],
                    self.Lb[j],self.Ub[j])
                rnd = np.random.random_sample()
                if rnd > self.r:
                    for j in range(self.D):
                        S[i][j] = self.best[j] + 0.001 *
                        random.gauss(0, 1)
                        S[i][j] = self.simplebounds(S[i][j],
                        self.Lb[j], self.Ub[j])
                Fnew = self.Fun(self.D, S[i])
                rnd = np.random.random_sample()
                if (Fnew <= self.Fitness[i])
                        and (rnd < self.A):
                    for j in range(self.D):
                        self.Sol[i][j] = S[i][j]
                    self.Fitness[i] = Fnew
```

```
                if Fnew <= self.f_min:
                    for j in range(self.D):
                        self.best[j] = S[i][j]
                    self.f_min = Fnew
        print(self.f_min)

import random
from BatAlgorithm import *

def Fun(D, sol):
    val = 0.0
    for i in range(D):
        val = val + sol[i] * sol[i]
    return val

for i in range(10):
   a = np.random.random_sample()
   b = np.random.random_sample()
   Algorithm = BatAlgorithm(5, 30, 5000, a, b, 0.0, 2.0,
                            -62.0, 62.0, Fun)
   Algorithm.move_bat()
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Додаток Г

(обов'язковий)

Довідка про впровадження

Додаток Д

(обов'язковий)

Протокол перевірки МКР

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка інтелектуального модуля для розв'язання задач оптимального проєктування конструкцій.

Тип роботи: магістерська кваліфікаційна робота

Підрозділ: кафедра автоматизації та інтелектуальних інформаційних технологій, факультет інтелектуальних інформаційних технологій та автоматизації

Показники звіту подібності Plagiat.pl (StrikePlagiarism)

Оригінальність _____% Схожість _____%

Аналіз звіту подібності (відмітити потрібне):

- ■ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату
- □ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на розгляд експертної комісії кафедри.
- □ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень

Особа, відповідальна за перевірку _____ Маслій Р. В.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Plagiat.pl (StrikePlagiarism) щодо роботи.

Автор роботи _____ Вовковинський В. О.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Іванов Ю. Ю.
(підпис) (прізвище, ініціали)