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

Бакалаврська кваліфікаційна робота на тему:

«ДОСЛІДЖЕННЯ РОЙОВОГО АЛГОРИТМУ АМЕБИ З МУЛЬТИСТАРТАМИ»

Виконала: студентка IV курсу, групи 1ІСТ-21б
Спеціальність 126 – Інформаційні системи та технології
Корнєєва А. Р.

Керівник: к.т.н., доцент каф. АІІТ
Кривогубченко С. Г.
«12» 06 2025 р.

Рецензент: к.т.н., професор каф. КСУ
Биков М.М.
«12» 06 2025 р.

Допущено до захисту
зав. кафедри АІІТ
д.т.н., проф. Бісікало О. В.
«13» 06 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет

Факультет інтелектуальних інформаційних технологій та автоматизації

Кафедра автоматизації та інтелектуальних інформаційних технологій

Рівень вищої освіти перший (бакалаврський)

Галузь знань 12 – «Інформаційні технології»

Спеціальність 126 – «Інформаційні системи та технології»

Освітньо-професійна програма Інтелектуальні інформаційні системи

ЗАТВЕРДЖУЮ

завідувач кафедри АІІТ

д.т.н., проф. Бісікало О. В.

«13» 06 2025 р.

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Корнєєвій Анні Ростиславівні

1. Тема роботи: Дослідження алгоритму амеби з мультистартами.

   Керівник роботи: к.т.н., доцент каф. АІІТ Кривогубченко С. Г.

   затверджені наказом ВНТУ від «20» 03 2025 року № 97.
2. Термін подання студентом роботи: «до 23» 06 2025 р.
3. Вихідні дані до роботи: розмірність простору пошуку; кількість особин; довжина ребра та координати вершин амеби; параметри віддзеркалення, стиснення, розширення, редукції; кількість рестартів; набір тестових функцій.
4. Зміст текстової частини: вступ; аналіз предметної області; розробка математичної моделі ройового алгоритму амеби; розробка програмного забезпечення та експериментальні дослідження; висновки; список використаних джерел.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень): можливі варіанти переміщення амеб; приклад пошуку однією амебою; схема програми.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1–3	Кривогубченко С. Г. к.т.н., доцент каф. АІІТ Іванов Ю. Ю. к.т.н., доцент каф. АІІТ	21.03.2025	02.06.2025

7. Дата видачі завдання: «21»__03__ 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз предметної області	21.03.2025	04.04.2025	Виконано
2	Розробка математичної моделі ройового алгоритму амеби	05.04.2025	26.04.2025	Виконано
3	Розробка програмного забезпечення та експериментальні дослідження	27.04.2025	16.05.2025	Виконано
4	Оформлення пояснювальної записки та ілюстративного матеріалу	17.05.2025	27.05.2025	Виконано
5	Попередній захист роботи	28.05.2025	02.06.2025	Виконано
6	Остаточний захист роботи	16.06.2025	23.06.2025	Виконано

Студентка _____ (підпис) Корнєєва А. Р. (прізвище та ініціали)

Керівник роботи _____ (підпис) Кривогубченко С. Г. (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 70 сторінок формату А4, на яких є 26 рисунків, 4 таблиці, список використаних джерел містить 25 найменувань.

Для розв'язання задачі пошуку екстремуму неперервних функцій розроблено, проаналізовано та досліджено алгоритм, заснований на поведінці амеб із мультистартами. Результатом роботи є програмне забезпечення, яке дозволяє розв'язати поставлену задачу оптимізації та виконати експериментальні дослідження.

Ключові слова: оптимізація, екстремум, амеба, рій, мультистарт.

ABSTRACT

The bachelor's thesis consists of 70 A4 pages with 26 figures, 4 tables, the list of sources used contains 25 items.

An algorithm, based on the behaviour of an amoeba with the multistarts, for solving the task of finding the extremum of continuous functions has been developed, analyzed and researched. The result of the work is software, that allows us to solve the given optimization task and to perform experimental researches.

Keywords: optimization, extremum, amoeba, swarm, multistart.

ЗМІСТ

ВСТУП

Актуальність. Розвиток промисловості залежить від прогресу науки та техніки, які вимагають застосування різних методів оптимізації для розв'язання різноманітних задач. Задачі оптимізації, які потрібно вирішувати на етапі попереднього оцінювання та проєктування складних систем, володіють властивостями, які суттєво ускладнюють їх розв'язання: дискретні, неперевні або змішані змінні, алгоритмічно задані цільові функції, відсутність зручних для оптимізації властивостей або, принаймні, відсутність інформації про такі властивості тощо. Тому в таких задачах можуть бути застосовані лише алгоритми прямого пошуку, що не потребують інформації про властивості оптимізованої функції [1].

Найперспективнішими нині визнані метаевристики обчислювального інтелекту, теоретичний аналіз і практичне застосування яких викликає низку труднощів. Їх основу складають запозичені у природи принципи, які дозволяють використовувати моделювання поведінки природних процесів і явищ на комп'ютері [2–5].

Оскільки розробка та дослідження таких алгоритмів є важливим питанням для інженерів-дослідників, то за цим напрямком відбувається збільшення кількості наукових праць різних авторів, зокрема С. Такенаги [6], П. Мехта [7], Л. Жу [8], Д. Максимця [9] та інших. У цих роботах виділяється алгоритм безумовної неперервної оптимізації функції від багатьох змінних, відомий як алгоритм амеби, особливістю якого є використання симплексу та

набору простих обчислювальних операцій. Тому дослідження даного методу є актуальним.

Мета і задачі дослідження. Метою роботи є розширення функціональних можливостей програмного забезпечення для розв'язання задач неперервної оптимізації функцій за рахунок використання ройового алгоритму амеби з мультистартами.

Для досягнення мети необхідно розв'язати наступні задачі:

– проаналізувати основні поняття теорії оптимізації;

– розробити математичну модель алгоритму амеби із застосуванням мультистартів;

– розробити програмні засоби та оцінити ефективність їх роботи.

Об'єктом дослідження є оптимізація функції від багатьох змінних.

Предметом дослідження є моделі, методи та інструментальні засоби для розв'язання задачі оптимізації функції від багатьох змінних.

Методи дослідження. У роботі використано поняття теорії оптимізації та обчислювального інтелекту під час дослідження роботи алгоритму амеби у ході розв'язання задач оптимізації функцій. Для аналізу та перевірки достовірності отриманих теоретичних положень застосовано експериментальне дослідження. Оброблення даних виконано за допомогою мови програмування *Python*.

Науково-технічний результат роботи полягає у тому, що на основі теоретичних положень розроблене математичне, алгоритмічне та програмне забезпечення, яке дозволяє оптимізувати функцію алгоритмом амеби з мультистартами.

У роботі отримані результати, які мають *практичну цінність*, а саме: наведено експериментальні дослідження розв'язання задачі оптимізації функцій з використанням алгоритму амеби з мультистартами.

Апробація результатів та публікації. За результатами даної роботи опубліковано тези доповіді [10] на Всеукраїнській науково-технічній конференції підрозділів ВНТУ (м. Вінниця, 2025).

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Теоретичні відомості

У природі й техніці трапляються випадки, коли явища і процеси описуються більше, ніж двома величинами. Наприклад, прибуток підприємства залежить не тільки від кількості виробленої продукції, а й від її собівартості, від ринкового попиту і, як наслідок, ціни продажу, від курсу валют і ще від багатьох факторів. Не завжди можна розкласти розв'язання таких задач на одновимірні складові. Для вивчення такого роду залежностей вводиться поняття функції від багатьох змінних $y = f(x)$, яка представляє правило, що ставить у відповідність кожному значенню аргументу x єдине значення y. Функція є однопараметричною, якщо аргумент x – скаляр, і – багатопараметричною, якщо x – вектор, тобто $x = (x_1, x_2, ..., x_n)^T$ [2, 11].

Точкою глобального мінімуму функції $y = f(x)$ на множині D є точка

$$f(x^*) \le f(x), \forall x \in D, \tag{1.1}$$

а точкою локального мінімуму

$$f(x^*) \le f(x), \forall x \in D \cap O_\varepsilon(x^*), \tag{1.2}$$

де $O_\varepsilon(x^*) = \{x \in R^n \,||\, x - x^*| \le \varepsilon\}$ – ε–окіл точки x^*.

Точки мінімуму (максимуму) разом називають точками оптимуму

(екстремуму). Функція може не мати точок екстремуму ($y = x^3$), мати один локальний мінімум ($y = x^2$) або локальний максимум ($y = -x^2$), мати кілька локальних мінімумів і максимумів ($y = \sin(x)$) тощо. Якщо формули (1.1) та (1.2) виконуються суворо, то x^* – суворе рішення у глобальному або локальному сенсі відповідно (рисунок 1.1) [12].

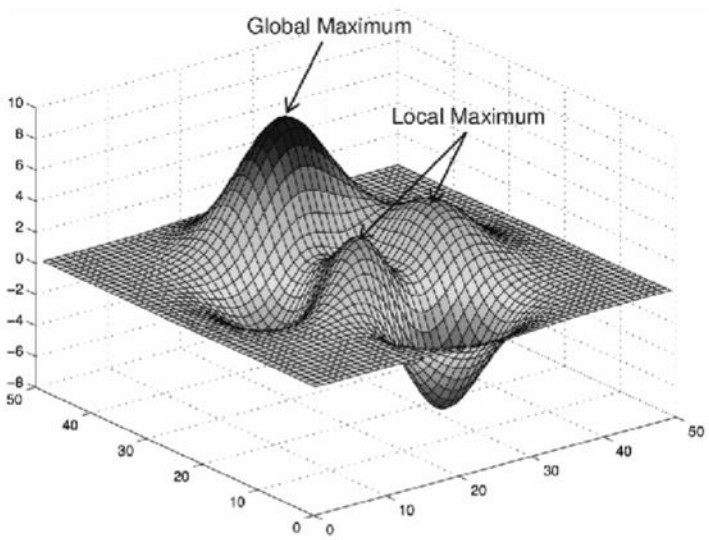


Рисунок 1.1 – Функція з локальними та глобальним екстремумами

Якщо $x^* \in D$ – точка глобального мінімуму $y = f(x)$ на множині D, то:

$$x^* = \arg\min_{x \in D} f(x), \tag{1.3}$$

$$f(x^*) = \min_{x \in D} f(x). \tag{1.4}$$

Множину всіх точок глобального мінімуму на множині D позначають:

$$\arg\min_{x \in D} f(x) = \{x^* \in D \mid f(x^*) = f^*\}. \tag{1.5}$$

Лінією рівня функції $f(x)$ називають множину точок, в яких функція приймає однакові значення. Для функцій двох змінних поверхні рівня представляють собою лінії рівня (рисунок 1.2) [2, 12].

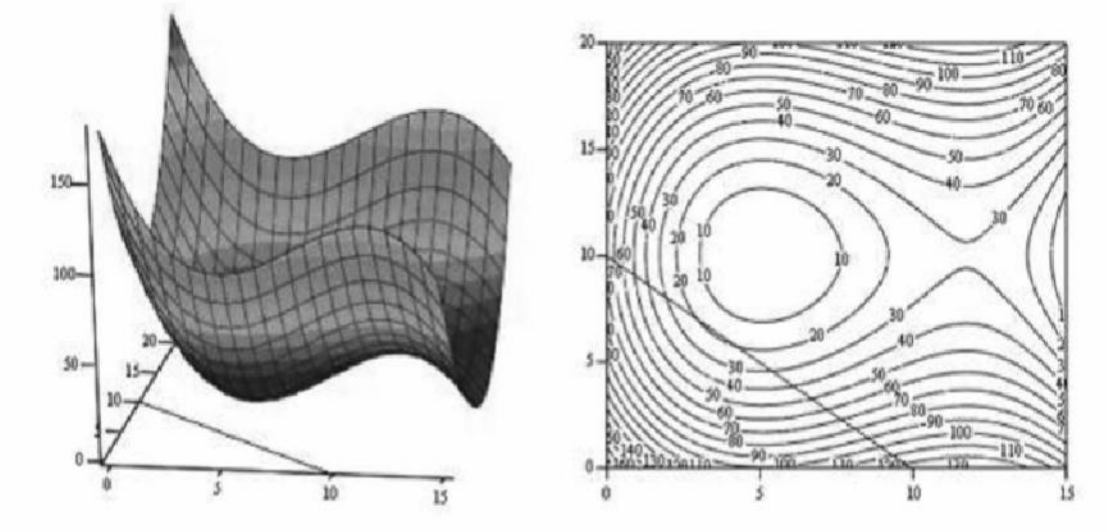


Рисунок 1.2 – Приклад ліній рівня

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

$$\nabla f(x) = \begin{pmatrix} \dfrac{\partial f(x)}{\partial x_1} & \dfrac{\partial f(x)}{\partial x_2} & \text{K} & \dfrac{\partial f(x)}{\partial x_n} \end{pmatrix}^T. \qquad (1.6)$$

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 [13]

$$\|\nabla f(x)\| = \sqrt{\sum_{i=1}^{n}\left(\frac{\partial f(x)}{\partial x_i}\right)^2}\,. \tag{1.7}$$

Необхідна умова екстремуму: якщо у певній точці диференційована функція $f(x)$ має екстремум, то її частинні похідні в цій точці дорівнюють нулю. Функція може мати екстремум в точках, де хоча б одна із похідних не існує.

Матрицею Гессе $H(x)$ неперервної диференційовної в точці x функції $f(x)$ називають матрицю частинних похідних другого порядку [2]:

$$H(x) = \begin{pmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \text{K} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2^2} & \text{K} & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\ \text{M} & \text{M} & \text{O} & \text{M} \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & \text{K} & \frac{\partial^2 f(x)}{\partial x_n^2} \end{pmatrix}. \tag{1.8}$$

Матриця Гессе має розмір $n \times n$ і є симетричною відносно головної діагоналі [12, 13].

Квадратична форма $\Delta x^T H(x) \Delta x$, а також відповідна матриця Гессе $H(x)$ називається:

– нульовою ($H(x) = 0$), якщо для будь-якого Δx виконується рівність $\Delta x^T H(x) \Delta x = 0$;

– додатньо визначеною ($H(x) > 0$), якщо для будь-якого ненульового Δx виконується строга нерівність $\Delta x^T H(x) \Delta x > 0$;

– від'ємно визначеною ($H(x) < 0$), якщо для будь-якого ненульового Δx виконується строга нерівність $\Delta x^T H(x) \Delta x < 0$;

– додатньо напіввизначеною ($H(x) \geq 0$), якщо для будь-якого ненульового Δx виконується нестрога нерівність $\Delta x^T H(x) \Delta x \geq 0$;

– від'ємно напіввизначеною ($H(x) \leq 0$), якщо для будь-якого ненульового Δx виконується нестрога нерівність $\Delta x^T H(x) \Delta x \leq 0$;

– невизначеною ($H(x) <> 0$), якщо існують такі вектори Δx і $\Delta \bar{x}$, що виконуються нерівності $\Delta x^T H(x) \Delta x < 0$, $\Delta \bar{x}^T H(x) \Delta \bar{x} > 0$.

Наприклад, для функції $f(x) = x_1^2 + x_2^2$ градієнт буде

$$\nabla f(x) = \begin{pmatrix} 2x_1 \\ 2x_2 \end{pmatrix}, \tag{1.9}$$

а його норма

$$\|\nabla f(x)\| = 2\sqrt{x_1^{\ 2} + x_2^{\ 2}}, \tag{1.10}$$

тоді матриця Гессе буде виглядати наступним чином:

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

Відповідно квадратична форма для будь-якого ненульового Δx більша нуля, тобто вона додатньо визначена $H(x) > 0$:

$$\Delta x^T H(x)\Delta x = (\Delta x_1\ \Delta x_2)\begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}\begin{pmatrix} \Delta x_1 \\ \Delta x_2 \end{pmatrix} = 2(\Delta x_1^2 + \Delta x_2^2). \qquad (1.11)$$

Операція множення матриць $\Delta x^T H(x)\Delta x$ є досить трудомісткою, тому для визначення характеру квадратичної форми використовують критерій Сільвестра (перевірка достатніх умов екстремуму за таблицею 1.1 [12]):

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

$$\Delta_1 = h_{11},\ \Delta_2 = \begin{vmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{vmatrix}, \ldots,\ \Delta_n = \begin{vmatrix} h_{11} & \Lambda & h_{1n} \\ \Lambda & \Lambda & \Lambda \\ h_{n1} & \Lambda & h_{nn} \end{vmatrix}; \qquad (1.12)$$

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, прирівнявши градієнт до нуля, та другого порядку, що можна записати наступним чином (таблиця 1.1):

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.

Таблиця 1.1 — Критерії перевірки достатніх та необхідних умов екстремуму другого порядку при $\nabla f(x^*) = 0$

$H(x^*)$	Умови		Тип точки x^*
> 0	Достатні умови	$\Delta_1 > 0, \Delta_2 > 0, \ldots, \Delta_n > 0$	Мінімум
< 0	Достатні умови	$\Delta_1 < 0, \Delta_2 > 0, \ldots, (-1)^n \Delta_n > 0$	Максимум
≥ 0	Необхідні умови другого порядку	Всі головні мінори визначника матриці Гессе $H(x^*)$ невід'ємні	Може бути мінімум, потрібні дослідження
≤ 0	Необхідні умови другого порядку	Всі головні мінори парного порядку невід'ємні, а непарного недодатні	Може бути максимум, потрібні дослідження
$= 0$	Необхідні умови другого порядку	Матриця Гессе складається з нульових елементів	Потрібні додаткові дослідження
$\gtrless 0$	Необхідні умови другого порядку	Не виконуються умови пунктів 1-5	Немає екстремуму

Головними мінорами називаються визначники m-ого порядку ($m \leq n$), 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 – це використання власних значень λ_i матриці Гессе $H(x)$, які можна знайти з характеристичного рівняння, застосувавши детермінант матриці [13]

$$\left|\begin{pmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} - \lambda & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \text{K} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2^2} - \lambda & \text{K} & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\ \text{M} & \text{M} & \text{O} & \text{M} \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & \text{K} & \frac{\partial^2 f(x)}{\partial x_n^2} - \lambda \end{pmatrix}\right| = 0. \qquad (1.13)$$

Можна сформулювати такі правила:

1) Матриця Гессе $H(x)$ додатньо визначена тоді, коли власні значення λ_i додатні;

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

3) при будь-якій іншій комбінації власних значень λ_i квадратична форма буде невизначеною.

1.2 Постановка задачі оптимізації

Під терміном "оптимізація", як правило, в техніці розуміють процес, який дає можливість отримати найкраще рішення в заданих умовах (знайти екстремум певної функції, вибрати найкращий або оптимальний варіант з множини можливих). В інших умовах отриманий результат вже може і не бути найкращим, оптимальним. У процесі розв'язання складних технічних оптимізаційних задач оцінку "найкращого рішення" має приймати проєктувальник, який має значний досвід у цій області або колектив досвідчених науковців [2].

У процесі вивчення та формалізації оптимізаційних задач оперують такими основними поняттями [3, 5, 10]:

1. Цільова функція $f(x)$ (функція втрат, критерій якості) – це функція, яка дозволяє розрахувати значення критерію оптимальності системи.

2. Керовані змінні x (проєктні параметри) – параметри системи, які впливають на значення цільової функції, в процесі оптимізації можуть бути змінені.

3. Обмеження (умови, область допустимих рішень) – множина допустимих значень керованих змінних.

Отже, задана цільова функція $f(x) = f(x_1, x_2, \ldots, x_n)$, визначена на множині допустимих рішень $D \subseteq R^n$. У найбільш загальному вигляді задача безумовної неперервної оптимізації функції від багатьох змінних формулюється так: потрібно знайти глобальний мінімум функції $f(x)$ на множині допустимих рішень D, тобто таку точку $x^* \in D$, що [10]

$$f(x^*) = \min f(x), \tag{1.14}$$

де $x = (x_1, x_2, \ldots, x_n)^T$, $D = \{x \mid x_i \in |LB_i, UB_i|, i = 1, 2 \ldots, n\}$;

LB, UB – нижня та верхня межа кожної змінної по області допустимих рішень D.

У даний час, в зв'язку з доступністю обчислювальної техніки, велика увага приділяється використанню аналітичного та чисельних методів оптимізації, які можна розділити на дві великі групи: методи безумовної та умовної оптимізації. Переважна більшість цих методів призначена для

розв'язання задач пошуку мінімуму, тому задача пошуку максимуму функції $f(x)$ зводиться до задачі пошуку мінімуму шляхом заміни знака перед функцією на протилежний (рисунок 1.3):

$$f(x) \to \max \quad \Rightarrow \quad -f(x) \to \min . \tag{1.15}$$

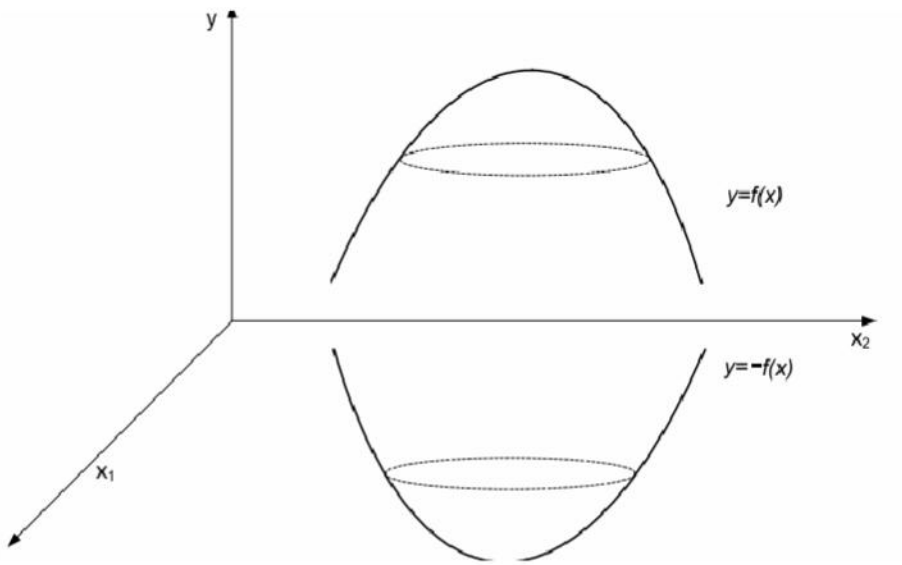


Рисунок 1.3 – Перетворення задачі на максимум в задачу на мінімум

У функції може бути багато локальних мінімумів (багатоекстремальна функція). Глобальний екстремум завжди є локальним, але не навпаки. 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 Аналітичний метод пошуку оптимуму

Точки екстремуму функції можна шукати, використовуючи аналітичний підхід, тобто застосовуючи необхідні і достатні умови екстремуму. Пошук точок локального екстремуму функції від багатьох змінних за допомогою необхідних і достатніх умов екстремуму називають непрямим методом. До недоліків непрямого методу слід віднести необхідність розв'язання системи рівнянь, яку отримують в результаті прирівнювання градієнта до нуля. Алгоритм пошуку екстремуму можна привести у такій формі [2]:

1. Перевірити необхідні умови екстремуму першого порядку, прирівнявши градієнт до нуля, тобто

$$grad\ f(x) = \nabla f(x) = 0. \tag{1.16}$$

Наприклад, для функції $f(x) = -x_1^2 - x_2^2 - x_3^2 + 4x_3 - 6$ обчислюємо градієнт функції та маємо

$$\nabla f(x) = \begin{pmatrix} -2x_1 \\ -2x_2 \\ -2x_3 + 4 \end{pmatrix}. \tag{1.17}$$

2. Знайти стаціонарні точки в результаті розв'язання n рівнянь (у загальному випадку нелінійних алгебраїчних) з n невідомими. Для наближеного розв'язку можна використовувати методи простої ітерації, Ньютона тощо [11].

Прирівнюємо градієнт до нуля і розв'язуємо отриману систему рівнянь:

$$\nabla f(x) = 0 \Rightarrow \begin{cases} -2x_1 = 0; \\ -2x_2 = 0; \\ -2x_3 + 4 = 0. \end{cases}$$

Розв'язком системи буде точка:

$$x^* = \begin{pmatrix} 0 \\ 0 \\ 2 \end{pmatrix}.$$

3. У знайдених точках перевірити достатні умови екстремуму з використанням критерію Сільвестра та, якщо потрібно, необхідні умови екстремуму другого порядку. Можна також застосувати спосіб на основі обчислення власних значень матриці Гессе $H(x)$.

Обчислюємо матрицю Гессе $H(x)$ в точці x^*:

$$H(x^*) = \begin{pmatrix} -2 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -2 \end{pmatrix}.$$

Визначаємо характер отриманої матриці Гессе $H(x)$ за критерієм Сільвестра. Для чого обчислюємо визначники:

$$\Delta_1 = -2,$$

$$\Delta_2 = +4,$$

$$\Delta_3 = -8.$$

За критерієм Сільвестра $H(x^*)$ від'ємно визначена, тоді точка x^* – максимум.

4. Обчислити значення функції $f(x)$ в точках екстремуму.

Значення функції в точці x^* дорівнює -2.

Слід зазначити, що у ході розв'язання інженером задач проєктування та управління технічними системами проявляються їх властивості [2]:

- висока розмірність простору пошуку;
- нелінійність;
- мультимодальність;
- недиференційовність;
- багатокритеріальність;
- робота в режимі "онлайн".

Отже, у центрі уваги інженерів з'являються все більш складні системи, що ускладнює використання фізичних моделей і підвищує значущість математичних моделей та використання обчислювальної техніки для їхнього аналізу. Саме тому в більшості практичних задач застосовують прямі (пошукові) або метаевристичні алгоритми пошуку.

1.4 Висновки

У розділі розглянуто основну термінологію, представлено загальну постановку задачі оптимізації та наведено аналітичний алгоритм пошуку екстремуму. Ефективність алгоритмів оптимізації пов'язана із досягненнями у галузі математики, а саме з розробкою ітераційних обчислювальних схем, ідея формування яких розглянута далі.

2 РОЗРОБКА МАТЕМАТИЧНОЇ МОДЕЛІ РОЙОВОГО АЛГОРИТМУ АМЕБИ

2.1 Особливості пошуку екстремуму чисельними методами

Клас чисельних методів, які використовують похідні 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. Розв'язання задачі мінімізації функції чисельними методами пов'язане зі знаходженням послідовності точок, які задовольняють умові [12]

$$f(x^{k+1}) < f(x^k),\ k = 0, 1, ..., n. \tag{2.1}$$

При цьому пошук мінімуму x^* починається у довільно вибраній точці x^0, а наступні точки послідовності знаходять за формулою [10]

$$x^{(k+1)} = x^{(k)} + l^{(k)} d^{(k)}, \tag{2.2}$$

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 $x^{(k+1)}$;

k – номер ітерації.

Багатовимірні методи, які реалізують процедуру пошуку оптимуму на основі обчислення значень функції, із загальних позицій можна розділити на евристичні та теоретичні. Евристичні методи, як це випливає з назви, реалізують процедуру пошуку за допомогою інтуїтивних геометричних уявлень і забезпечують отримання деяких емпіричних результатів. З іншого боку, теоретичні методи засновані на фундаментальних математичних теоремах і володіють такими операційними властивостями, як збіжність [3].

У залежності від найвищого порядку частинних похідних функції $f(x)$, які використовуються для знаходження $l^{(k)}$ і $d^{(k)}$, методи мінімізації функції від багатьох змінних поділяють на три групи (рисунок 2.1) [2, 12, 13]:

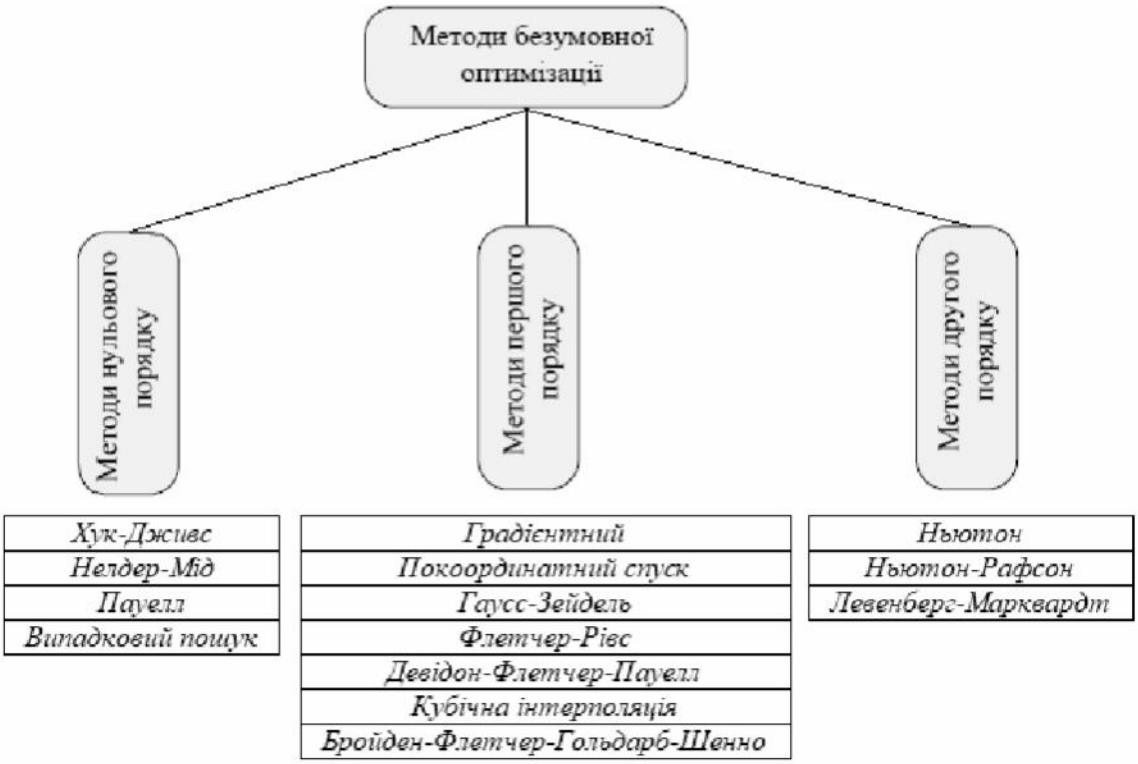


Рисунок 2.1 – Методи розв’язання задач безумовної оптимізації

1) методи нульового порядку, які використовують інформацію тільки про значення функції (методи Хука-Дживса, Розенброка, Пауелла,

випадкового пошуку). Частина цих методів не має строгого теоретичного обґрунтування. Вони побудовані на основі евристичних міркувань. У даному розділі будемо розглядати метаевристику нульового порядку, яка використовує поведінку амеби для пошуку оптимуму;

2) методи першого порядку, які використовують інформацію про перші похідні функції (методи градієнтного спуску, Гаусса-Зейделя, Флетчера-Рівса, Девідона-Флетчера-Пауелла, Бройдена-Флетчера-Гольдарба-Шенно);

3) методи другого порядку, які використовують також інформацію про другі похідні функції (метод Ньютона, Ньютона-Рафсона, Левенберга-Марквардта).

Слід зазначити, що ні один метод або клас методів не відрізняється високою ефективністю при вирішенні оптимізаційних задач різних типів. Зокрема, можливі випадки, коли відбувається переповнення пам'яті ЕОМ. У ряді програм або неможливо, або дуже важко знайти аналітичні вирази для похідних цільової функції. Тому, якщо передбачається використання градієнтних методів, слід застосувати процедуру різницевої апроксимації похідних. У свою чергу це призводить до необхідності експериментального визначення довжини кроків, що дозволяє встановити належну відповідність між помилками округлення і апроксимації. Крім того, зазначені методи можуть поводитися по-різному на різних етапах процесу оптимізації.

Задовільного способу подолання вказаних труднощів не існує. Єдине, що можна зробити в подібній ситуації — це привести дані про результати обчислень в розгорненій формі, що дозволяє порівнювати методи за різними критеріями. Таким чином, інженер змушений налаштовувати кожний 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.

2.2 Ройовий алгоритм амеби

Завичай у ході прямого пошуку область визначення цільової функції замінюється дискретною множиною точок простору керованих змінних. Після чого використовуються різні стратегії зменшення заданої області. Часто дана процедура є еквівалентною рівномірному пошуку у вузлах гратки, що робить її непридатною для розв'язання задач з кількістю змінних $n > 2$. Більш доцільною для використання є ідея, яка полягає у виборі базової точки і оцінюванні значень цільової функції в її околі (оточуючих точках). Наприклад, під час розв'язання оптимізаційної задачі з розмірністю $n = 2$ можна використати квадратний зразок (рисунок 2.2) [7, 14, 15].

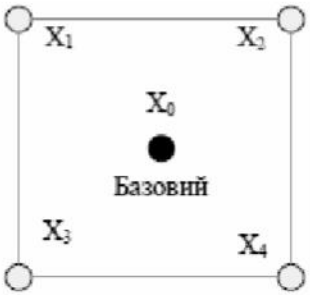


Рисунок 2.2 — Квадратний зразок

Потім "найкраща" з досліджуваних точок стає наступною базовою точкою, навколо якої будується аналогічний зразок (рисунок 2.3). Якщо жодна з точок зразка не має переваги перед базовою точкою, то розміри зразка зменшують, а потім продовжують пошук.

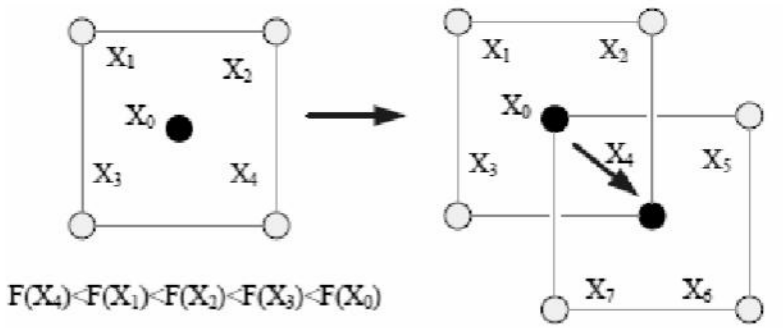
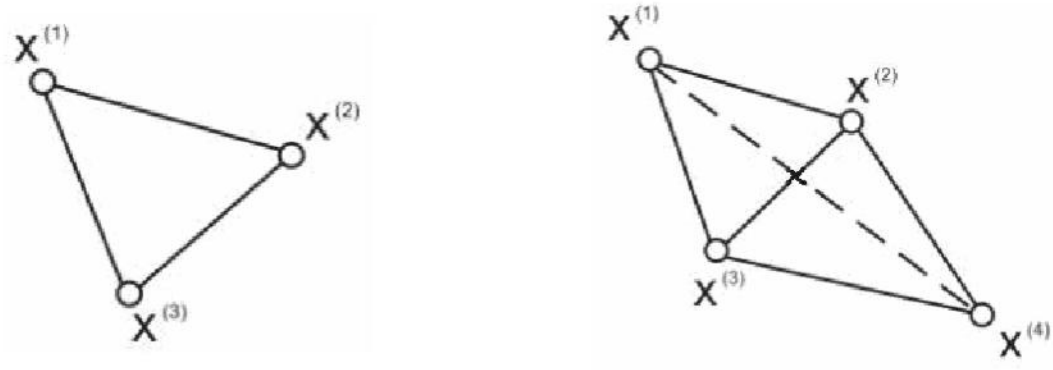


Рисунок 2.3 — Пошук екстремуму за квадратним зразком

У задачах великої розмірності значення $f(x)$ обчислюються в усіх вершинах, а також у центрі ваги гіперкуба (куб в n-мірному евклідовому просторі, тобто множина $S = \{x = x_1, x_2, ..., x_n \in R^n \mid a \le x_i \le b\}$, де a і b — задані числа), тобто в точках кубічного зразка. Якщо розмірність простору, в якому ведеться пошук, дорівнює n, то пошук з використанням кубічного зразка вимагає 2^n+1 обчислень значення $f(x)$ для одного зразка. При збільшенні розмірності задачі n кількість обчислень $f(x)$ зростає надзвичайно швидко. Таким чином, незважаючи на логічну простоту пошуку, на практиці виникає необхідність використання більш ефективних методів прямого пошуку [7].

Процедура пошуку Спендлі-Хекста-Хімсворта (даний метод не має відношення до симплекс-методу лінійного програмування, а схожість назв носить випадковий характер) базується на тому, що зразком, який містить найменшу кількість точок, є регулярний симплекс, який в n-мірному просторі представляє собою багатогранник, утворений $(n+1)$ рівновіддаленою одна від одної точкою (вершин симплекса). Наприклад, у випадку $n=1$ симплексом є відрізок, при $n=2$ – рівносторонній трикутник; в тривимірному просторі – тетраедр (рисунок 2.4). Регулярний симплекс будується з метою швидкого руху в напрямку мінімуму функції $f(x)$ [15].



(а) (б)

Рисунок 2.4 — Приклад дво- (а) та тривимірного (б) симплексів

Загалом в n-вимірному просторі координати вершин симплекса можна задати матрицею, врахувавши, що перша точка X має координати $(x_1,\ldots, x_n)^T$, тобто [6]

$$S_{n\times(n+1)} = \begin{bmatrix} x_1 & x_1+r_1 & x_1+r_2 & \ldots & x_1+r_2 \\ x_2 & x_2+r_2 & x_2+r_1 & \ldots & x_2+r_2 \\ x_3 & x_3+r_2 & x_3+r_2 & \ldots & x_3+r_2 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ x_n & x_n+r_2 & x_n+r_2 & \ldots & x_n+r_1 \end{bmatrix}, \tag{2.3}$$

причому

$$r_1 = L\cdot\frac{\sqrt{n+1}+n-1}{n\sqrt{2}}\ , \tag{2.4}$$

$$r_2 = L\cdot\frac{\sqrt{n+1}-1}{n\sqrt{2}}, \tag{2.5}$$

де L — довжина ребра симплекса.

Отже, щоб задати регулярний симплекс достатньо використати координати початкової точки $X_0 = (x_1,\ldots, x_n)^T$ та довжину ребра симплекса L. У даному алгоритмі застосовано важливу властивість симплексу, згідно з якою новий симплекс можна побудувати на будь-якій грані базового симплексу шляхом переносу вибраної вершини на певну відстань (пропорційну α) уздовж прямої, проведеної через центр ваги інших вершин симплексу. Отже, визначається вершина, якій відповідає найбільше значення цільової функції $f(x)$. Потім знайдена вершина проєктується через центр ваги X_c інших вершин симплекса в нову точку. Отримана точка є вершиною нового симплексу, а вибрана під час побудови вершина базового симплексу видаляється із обчислень. Слід зазначити, що при переході до нового симплексу потрібно обчислити значення цільової функції один раз. Приклад побудови нового симплекса на площині представлено на рисунку 2.5, де базовий симплекс (а) мав вершини X_0, X_1, X_2, а кінцевий — X_2, X_3, X_4.

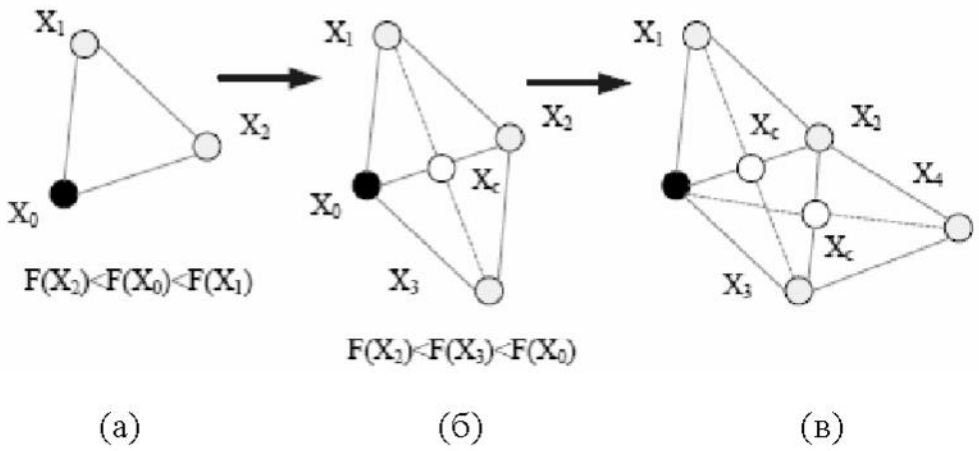


Рисунок 2.5 — Пошук оптимуму за симплексом

Якщо функція спадає досить плавно, то ітерації тривають поки не потрапимо в точку мінімуму, або не почнеться циклічний рух по декільком симплексам. У останньому випадку можна застосувати такі правила [14]:

1. “Накриття” точки мінімуму. Якщо вершина, якій відповідає найбільше значення цільової функції h, була побудована (розглянута) на попередній ітерації, то замість неї береться вершина, якій відповідає наступне за величиною значення цільової функції s.

2. Циклічний рух. Якщо деяка вершина симплекса не виключається в розрахунки більш ніж M ітерацій, то необхідно зменшити розміри симплекса за допомогою коефіцієнта редукції і побудувати новий симплекс, вибравши в якості базової точку ту, якій відповідає мінімальне значення цільової функції l. Наприклад, можна визначати M евристично за формулою:

$$M = 1{,}65 \cdot n + 0{,}05 \cdot n^2, \tag{2.6}$$

де n — розмірність задачі, а M округлюється до найближчого цілого числа.

3. Критерій закінчення пошуку. Пошук завершується тоді, коли розміри симплекса, або різниці між значеннями функції $f(x)$ у його вершинах стають меншими деякого числа $\xi \to 0$ (параметра закінчення пошуку).

У 1989 році з'являється праця Х. Бені і В. Цзіна, присвячена клітинним автоматам, в якій 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 (рисунок 2.6) [16].

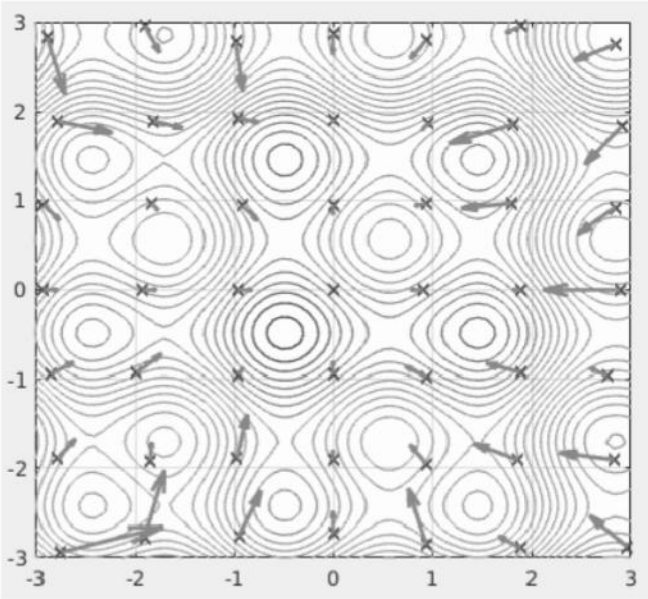


Рисунок 2.6 – Приклад роботи ройового інтелекту

У роботі [6] запропоновано алгоритм амеби, як одного агента рою, який дозволяє задати координати вершин симплекса, причому він може стати нерегулярним (неправильної форми). Алгоритм включає наступні складові:

– ініціалізація амеби S^n;

– сортування значень функції $f(x)$;

– розрахунок центру ваги X_c;

– віддзеркалення амеби S^n (параметр $\alpha > 0$);

– розширення амеби S^n (параметр $\gamma > 1, \gamma > \alpha$);

– стиснення амеби S^n (параметр $0 < \beta < 1$);

– редукція амеби S^n (параметр $0 < \delta < 1$).

В основу алгоритму покладена побудова послідовності $k = 1, 2, \ldots,$ систем точок $x^i(k)$, $i = 1, \ldots, n+1$, які є вершинами опуклого багатогранника. Точки системи $x^i(k+1)$ співпадають з точками системи $x^i(k)$, крім точки $x^h(k)$, яка є найгіршою в системі $x^i(k)$, і яку на $k+1$ ітерації заміняють по спеціальним правилам. В процесі виконання цих правил багатогранник

змінює свої розміри і форму, що і обумовило назву алгоритму. Побудова послідовності закінчується тоді, коли значення функції $f(x)$ у вершинах поточного багатогранника відрізняються від значення функції $f(x)$ у центрі багатогранника не більше, ніж на деяку задану величину $\xi \to 0$ або розміри багатогранника стають достатньо малими [10].

Кожна амеба виконує наступні дії [6, 7, 17, 18]:

1. Обчислюють координати початкового положення амеби x^1, x^2, ..., x^{n+1} за формулами (2.3)–(2.5), першу вершину генерують випадково. Задають параметри віддзеркалення α, стиснення β, розширення γ, редукції δ та необхідну точність $\xi \to 0$. Приймають номер ітерації $k = 0$.

2. Знаходять найкращу x^l і найгіршу x^h вершини амеби, які відповідають найменшому та найбільшому значенню функції $f(x)$, і точку x^s, в якій досягається друге за величиною значення функції $f(x)$, тобто

$$f(x^l) = \min_{j=1,\dots,n+1} (f(x^j)), \tag{2.7}$$

$$f(x^h) = \max_{j=1,\dots,n+1} (f(x^j)), \tag{2.8}$$

$$f(x^s) = \max_{\substack{j=1,\dots,n+1 \\ s \neq h}} (f(x^j)). \tag{2.9}$$

3. Знаходять центр ваги всіх вершин *c*, за виключенням точки x^h:

$$x^{n+2} = \frac{1}{n}\left(\left(\sum_{j=1}^{n+1} x^j\right) - x^h\right) = \frac{1}{n}\sum_{\substack{j=1 \\ j \neq h}}^{n+1} x^j . \tag{2.10}$$

4. Критерій закінчення пошуку. Якщо $RMSE \le \xi \to 0$, то алгоритм завершує роботу, а оптимум $x^* = x^l$, інакше — перейти на крок 5:

$$RMSE = \sqrt{\frac{1}{n+1} \cdot \sum_{j=1}^{n+1} \left(f\left(x^j\right) - f\left(x^{n+2}\right) \right)^2} . \qquad (2.11)$$

5. Виконати операцію віддзеркалення (рисунок 2.7) найгіршої вершини x^h через центр ваги x^{n+2}, тобто

$$x^{n+3} = x^{n+2} + \alpha \cdot \left(x^{n+2} - x^h \right). \qquad (2.12)$$

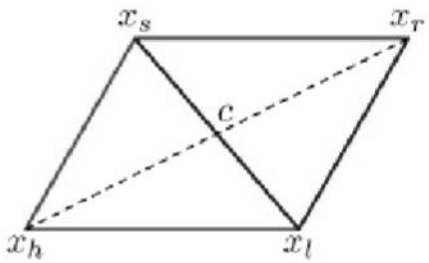


Рисунок 2.7 — Операція віддзеркалення

6. Виконують перевірку умов:

а) якщо $f\left(x^{n+3}\right) \le f\left(x^l\right)$, то виконують операцію розширення (рисунок 2.8):

$$x^{n+4} = x^{n+2} + \gamma \cdot \left(x^{n+3} - x^{n+2} \right). \qquad (2.13)$$

Знаходять вершини нової амеби:

– якщо $f\left(x^{n+4}\right) < f\left(x^{l}\right)$, то вершину x^{h} замінюють вершиною x^{n+4}, приймають $k = k + 1$ і переходять на крок 2;

– якщо $f\left(x^{n+4}\right) \geq f\left(x^{l}\right)$, то вершину x^{h} замінюють вершиною x^{n+3}, приймають $k = k + 1$ і переходять на крок 2;

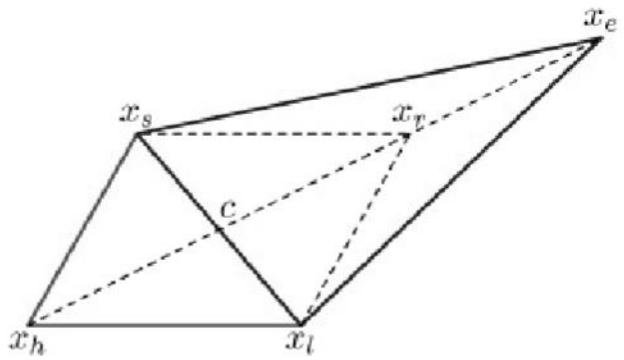


Рисунок 2.8 — Операція розширення

б) якщо $f\left(x^{s}\right) < f\left(x^{n+3}\right) \leq f\left(x^{h}\right)$, виконують операцію стиснення (рисунок 2.9):

$$x^{n+5} = x^{n+2} + \beta \cdot \left(x^{h} - x^{n+2}\right). \tag{2.14}$$

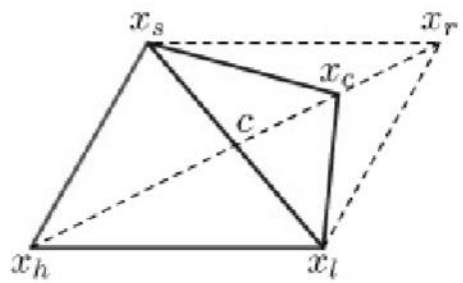


Рисунок 2.9 — Операція стиснення

Знаходять вершини нової амеби:

– вершину x^h замінюють вершиною x^{n+5}, приймають $k = k + 1$ і переходять на крок 2;

в) якщо $f\left(x^l\right) < f\left(x^{n+3}\right) \le f\left(x^s\right)$, то вершину x^h замінюють вершиною x^{n+3}, приймають $k = k + 1$ і переходять на крок 2;

г) якщо $f\left(x^{n+3}\right) > f\left(x^h\right)$, то виконують операцію редукції (рисунок 2.10), у результаті якої формується амеба зі зменшеними сторонами і вершиною x^l:

$$x^j = x^l + \delta \cdot \left(x^j - x^l\right),\ j = 1, \ldots, n+1. \qquad (2.15)$$

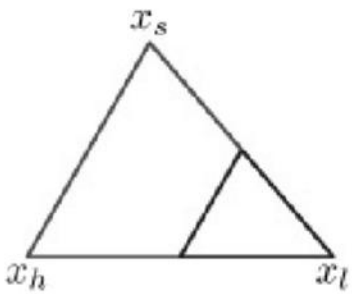


Рисунок 2.10 — Операція редукції

Далі приймають $k = k + 1$ і переходять на крок 2.

7. Рестарт, тобто нова генерація амеби у випадку, якщо алгоритм застряг у локальному оптимумі або завершився час роботи, досягнуто граничної кількості ітерацій руху амеби тощо. Приклад роботи алгоритму амеби наведено на рисунку 2.11.

Наприклад для мінімізації функції $f(x, y) = x^2 + xy + y^2 - 6x - 9y$ алгоритм амеби буде працювати таким чином. Координати початкових точок згенеруємо випадково: $X_1(0;0)$, $X_2(1;0)$, $X_3(0;1)$. Обчислимо значення функції точках $f(X_1) = f(0;0) = 0$, $f(X_2) = f(1;0) = -5$, $f(X_3) = f(0;1) = -8$.

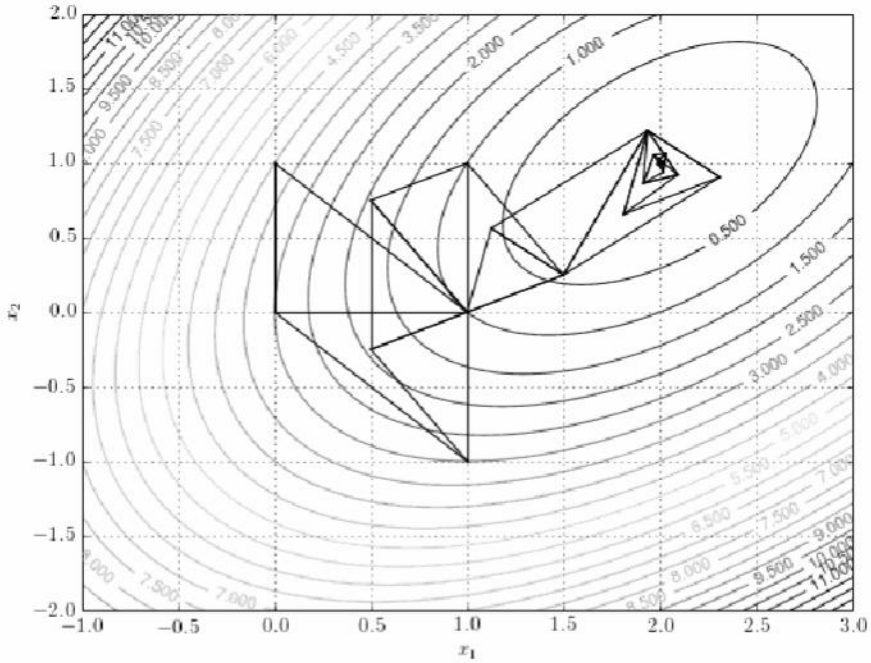


Рисунок 2.11 — Приклад роботи алгоритму

Тоді відсортуємо результати та перепозначимо точки $x^l = best = X_3(0;1)$, $x^s = good = X_2(1;0)$, $x^h = worst = X_1(0;0)$, що показано на рисунку 2.12.

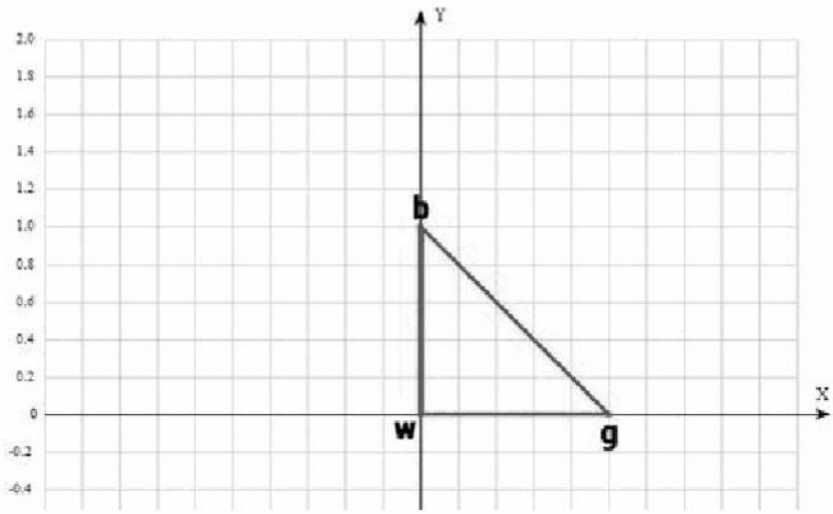


Рисунок 2.12 — Стартова форма амеби

Знаходимо середину відрізка bg (центр ваги):

$$c = mid = \frac{x^l + x^s}{2} = \frac{(0;1)^T + (1;0)^T}{2} = (0{,}5;\ 0{,}5)^T.$$

Розраховуємо *RMSE*:

$$RMSE = \sqrt{\frac{1}{2+1}\sum_{j=1}^{3}\left(f\left(x^j\right) - f\left(0{,}5;0{,}5\right)\right)^2} = 4{,}09 > \varepsilon = 0{,}02.$$

Операція віддзеркалення (рисунок 2.13):

$$x_r = c + \alpha \cdot (c - x^h) = (0{,}5;\ 0{,}5)^T + 1 \cdot ((0{,}5;\ 0{,}5)^T - (0;\ 0)^T) = (1;\ 1)^T.$$

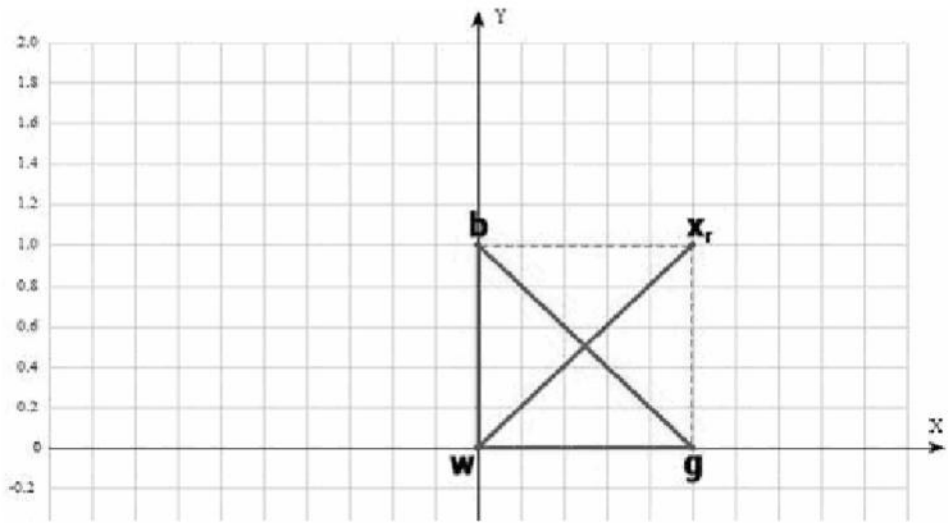


Рисунок 2.13 — Операція віддзеркалення

Перевіряємо точку x_r: $f(x_r) = f(1;1) = -12$, оскільки $f(x_r) < f(x^l)$, то необхідно виконати операцію розширення (рисунок 2.14), тобто

$$x_e = c + \gamma \cdot (x_r - c) = (0{,}5;\ 0{,}5)^T + 2 \cdot ((1;\ 1)^T - (0{,}5;\ 0{,}5)^T) = (1{,}5;\ 1{,}5)^T.$$

Перевіримо значення функції в точці x_e: $f(x_e) = f(1{,}5;1{,}5) = -15{,}75$, тобто точка x_e краща за точку x^l. Отже, отримуємо нові вершини $X_1(1{,}5;1{,}5)$, $X_2(1;0)$, $X_3(0;1)$. Переходимо на другу ітерацію. На 10-ій ітерації показник $RMSE \approx 0{,}018 < \varepsilon = 0{,}2$, тобто закінчуємо ітераційний процес.

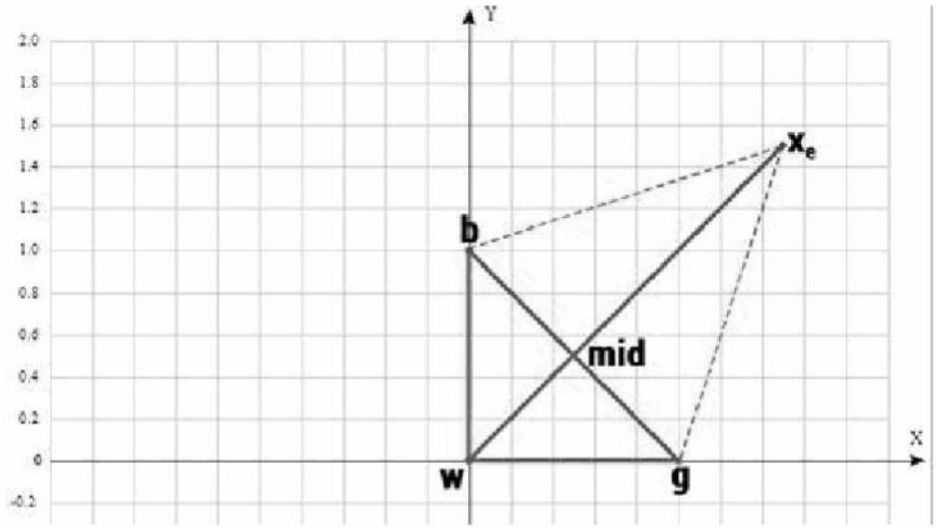


Рисунок 2.14 — Операція розширення

Алгоритм дозволяє отримати оптимум $f(0{,}990;4{,}002) = -20{,}995$. За аналітичними розрахунками екстремум даної функції становить $f(1; 4) = -21$.

2.3 Процедура центрування

У алгоритмі пошуку оптимуму з використанням поведінки амеби центр тяжіння рівномірно віддалений від перших n вершин амеби (крім h). У зваженому алгоритмі амеби спочатку обчислюють ваги μ_i, $i = \overline{1,n}$ [6, 7]:

$$\mu_i = \frac{f(x^h) - f(x_i)}{\rho(x^h, x_i)}, \tag{2.16}$$

де $\rho(x^h, x_i) = \sqrt{\sum_{j=1}^{n} (x_j^h - x_{i,j})^2}$ – відстань між вершинами x^h та x_i;

$(x_{i,1}, x_{i,2}, ..., x_{i,n})$ – координати вершин x_i, $i = \overline{1,n}$, крім вершини h.

Ваговий коефіцієнт μ_i представляє собою відносну зміну цільової функції $f(x)$ у вершинах амеби. Чим він більший, тим більша ймовірність того, що значення цільової функції у напрямку до вершини x_i зменшуються швидше. Доцільно нову вершину шукати у напрямку до тих вершин амеби, для яких вагові коефіцієнти μ_i більші. Відзначимо, що вагові коефіцієнти μ_i не задовольняють умові $\sum_{i=1}^{n} \mu_i = 1$, тому необхідно виконати їх нормування та знайти нормовані коефіцієнти λ_i за виразом

$$\lambda_i = \frac{\mu_i}{\sum_{i=1}^{n} \mu_i}, i = \overline{1,n}. \tag{2.17}$$

Для будь-яких *i, j,* якшо $\mu_i < \mu_j$, то $\lambda_i < \lambda_j$, тобто відношення вагових коефіцієнтів не змінюється. Передбачається вибрати вагові коефіцієнти λ_i, $i = \overline{1,n}$ так, щоб координати центру x^w були ближчими до тієї вершини амеби, у напрямку до якої цільова функція швидше спадає. Замість виразу (2.10) розраховується середньозважений центр грані, яка знаходиться навпроти найгіршої вершини з координатами x^h:

$$x^w = \sum_{\substack{j=1 \\ j \neq h}}^{n} \lambda_j \cdot x^j = \lambda_1 x_1 + \lambda_2 x_2 + \ldots + \lambda_n x_n, \tag{2.18}$$

$$\sum_{\substack{i=1 \\ i \neq h}}^{n} \lambda_i = 1,\ \lambda_i > 0,\ i = \overline{1,n}. \tag{2.19}$$

Відповідно центр тяжіння амеби буде ближчим до тієї вершини, для якої ваговий коефіцієнт більший. Подібна модифікація змінює напрямок пошуку нової вершини амеби, що підвищує ефективність алгоритму.

Наприклад, потрібно мінімізувати $f(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 1)^2 - x_1 \cdot x_2$, яка має мінімальне значення $f(x^*) = f(2;2) = -2$. Нехай отримано точки $x_1 = (2{,}5; 0{,}3)^T$; $x_2 = (-1; 1{,}2)^T$; $x_3 = (0{,}6; -2{,}3)^T$, тоді значення цільової функції у них буде $f(x_1) = 1{,}99$; $f(x_2) = 5{,}24$; $f(x_3) = 12{,}43$, тобто вершина x_3 – найгірша, а x_1 – найкраща. Тоді розрахуємо коефіцієнти μ_i:

$$\mu_1 = \frac{f(x_3) - f(x_1)}{\rho(x_3, x_1)} = \frac{12{,}43 - 1{,}99}{3{,}22} \approx 3{,}2422,$$

$$\mu_2 = \frac{f(x_3) - f(x_2)}{\rho(x_3, x_2)} = \frac{12{,}43 - 5{,}24}{3{,}8484} \approx 1{,}8683.$$

Виконаємо нормування коефіцієнтів:

$$\lambda_1 = \frac{\mu_1}{\sum_{i=1}^{n} \mu_i} = \frac{3{,}2422}{3{,}2422 + 1{,}8683} = 0{,}6344,$$

$$\lambda_2 = \frac{\mu_2}{\sum_{i=1}^{n} \mu_i} = \frac{1{,}8683}{3{,}2422 + 1{,}8683} = 0{,}3656.$$

Тоді можна розрахувати центр тяжіння за формулою (2.18):

$$x_c^{'} = 0{,}6344 \cdot x_1 + 0{,}3656 \cdot x_2 .$$

Із рисунку 2.15 видно, що отриманий середньозважений центр тяжіння амеби $x_c^{'}$ знаходиться ближче до найкращої вершини x_1 і до точки мінімуму цільової функції x^*, ніж центр тяжіння x_c, який визначається базовим алгоритмом амеби. Тим самим, з великою ймовірністю можна сказати, що на точках лінії l_2 (проходить через точки $x_c^{'}$ та x_3) цільова функція буде набувати менші значення, ніж на точках лінії l_1 (проходить через точки x_c та x_3). Для досягнення глобального мінімуму цільової функції потрібно повторити представлену процедуру пошуку декілька разів.

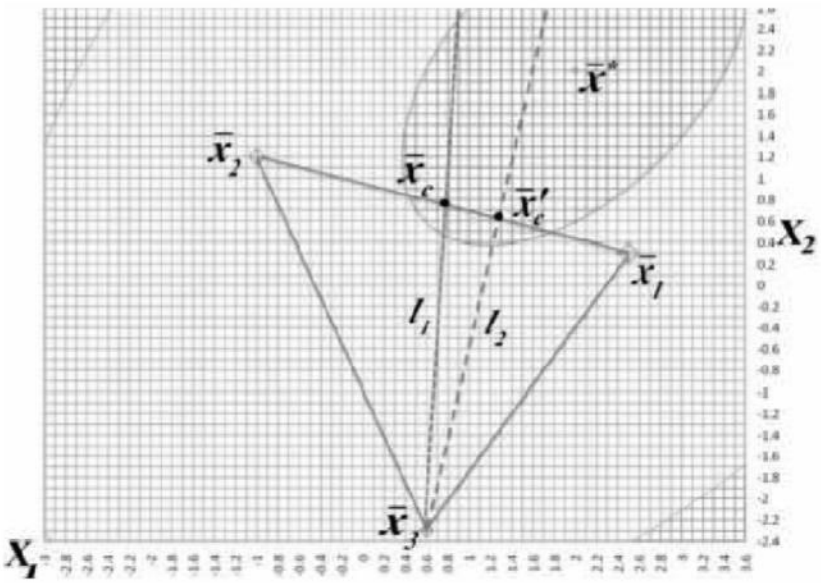


Рисунок 2.15 — Графічне відображення результатів пошуку оптимуму

Представлений алгоритм має декілька очевидних переваг [6, 7]:

– розрахунки та логічна структура вирізняються простотою;

– рівень вимог до об'єму оперативної пам'яті невисокий;

– використовується невелика кількість визначених параметрів;

– працює досить швидко, оскільки не використовуються похідні.

Даний алгоритм також має низку істотних недоліків [6, 7]:

– евристична оцінка поверхні функції;

– відсутність теорії збіжності.

2.4 Висновки

У розділі проаналізовано чисельні методи пошуку екстремуму функції. Для подальших досліджень вибрано алгоритм амеби зі зваженим центром ваги. У наступному розділі проведемо експериментальні дослідження ефективності роботи даного алгоритму пошуку екстремуму функції у розробленому програмному додатку.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

3.1 Вибір мови програмування

Однією з найпоширеніших мов програмування різноманітних методів обчислювального інтелекту є Python. Це інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією, розроблена в 1990 році Г. ван Россумом [19]. Перевагами, які сприяють збільшенню популярності даної мови, є структури даних високого рівня в поєднанні з динамічною семантикою, можливість динамічного зв'язування, що дозволяє швидко розробляти програми та легко поєднувати вже існуючі компоненти. Рейтинг мов програмування від IEEE Spectrum наведено на рисунку 3.1 [20].

Рисунок 3.1 Рейтинг мов програмування

Програми, написані на мові Python, можуть бути виконані на будь-якому персональному комп’ютері під керуванням десктопних операційних систем Windows, MacOS, Linux. Крім того, відсутні механізми, які потенційно призводять до помилок (наприклад, перетворення типів з втратою точності) та присутній обов'язковий контроль виняткових ситуацій.

Функціонування програми повністю визначається виконавчим середовищем. Відсутні покажчики та інші механізми для безпосередньої роботи з фізичною пам'яттю й іншим апаратним забезпеченням комп'ютера. Зручним є механізм автоматичного генерування документації на основі коду.

Висока модульність програмного забезпечення та легке повторне використання коду допускається завдяки тому, що Python підтримує пакети модулів, причому стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах. Окрім цього, Python підтримує кілька парадигм програмування, а саме об'єктно-орієнтовану, процедурну, функціональну та аспектно-орієнтовану. Багато допоміжних задач, які зустрічаються при розробці програмного забезпечення, вже вирішені в рамках стандартних та користувацьких бібліотек.

Мова Python представляє потужний інструмент для вирішення низки наукових і прикладних задач у різноманітних галузях: розробка систем управління, моделювання об'єктів, проєктування інформаційно-комунікаційних систем, оброблення сигналів і зображень, обчислювальна біологія, фінансове моделювання тощо. Отже, мова програмування Python досить зручна, надійна та потужна, що і обумовило необхідність її застосування у даній роботі.

3.2 Опис програми

Оскільки розглядаються специфічні питання наукового характеру, то можна відмітити, що існує мала кількість подібних аналогів-програм. Тому будемо розглядати результати наукових досліджень, порівнюючи результати з функціоналом математичних середовищ MatLab (аналог) та Python (базовий інструмент). У таблиці 3.1 наведені основні технічні показники аналога і нового програмного продукту.

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.

У роботі розроблено програмний додаток з графічним інтерфейсом користувача, який дає можливість:

– відображати поверхню та лінії функції $f(x)$, екстремум якої необхідно знайти;

– будувати на графіку шлях амеби;

– представляти результати пошуку оптимуму.

Для ініціалізації алгоритму потрібно вказати кількість амеб N, розмірність простору пошуку n, координати стартової вершини X_{i0} кожної i-ї амеби та довжину ребра L, а також кількість рестартів амеб. Сортування вершин "за якістю" відбувається швидко з використанням порівнянь:

$$if\,(a > b)\colon\ a = b;$$

$$if\,(b > c)\colon\ b = c;$$

$$if\,(a > b)\colon\ a = b.$$

Таблиця 3.1 – Основні технічні показники аналога і нового програмного продукту

Показники	Аналог	Розробка	Відношення параметрів нової розробки до параметрів аналога
Ліцензія	пропрієтарне забезпечення	вільне розповсюдження	перевага у розробки
Функціональність	можна змінювати код	можна прискорити код, змінювати його	перевага у розробки
Надійність	забезпечено необхідний рівень працездатності	забезпечено необхідний рівень працездатності	подібні
Простота	графічний інтерфейс + консоль, простіша мова програмування	графічний інтерфейс + консоль, більш складна мова програмування	перевага у аналога
Економія ресурсів	швидкодія, можливість задавати критерій зупинки	швидкодія, можливість роботи з графікою під час розрахунків	подібні

Щоб задати точність обчислення ξ для пошуку екстремуму потрібно вибрати від'ємний показник степеня для числа 10. Для того, щоб впевнитись,

що метод збігається потрібно провести експерименти p разів, кожний раз випадково вибираючи координати першої вершини амеб. У програмі реалізовано декілька тестових функцій.

Геометрична інтерпретація (рисунок 3.2) та розрахунки доводять, що програма працює правильно. Дослідження роботи програмного забезпечення на інших прикладах показало коректні результати.

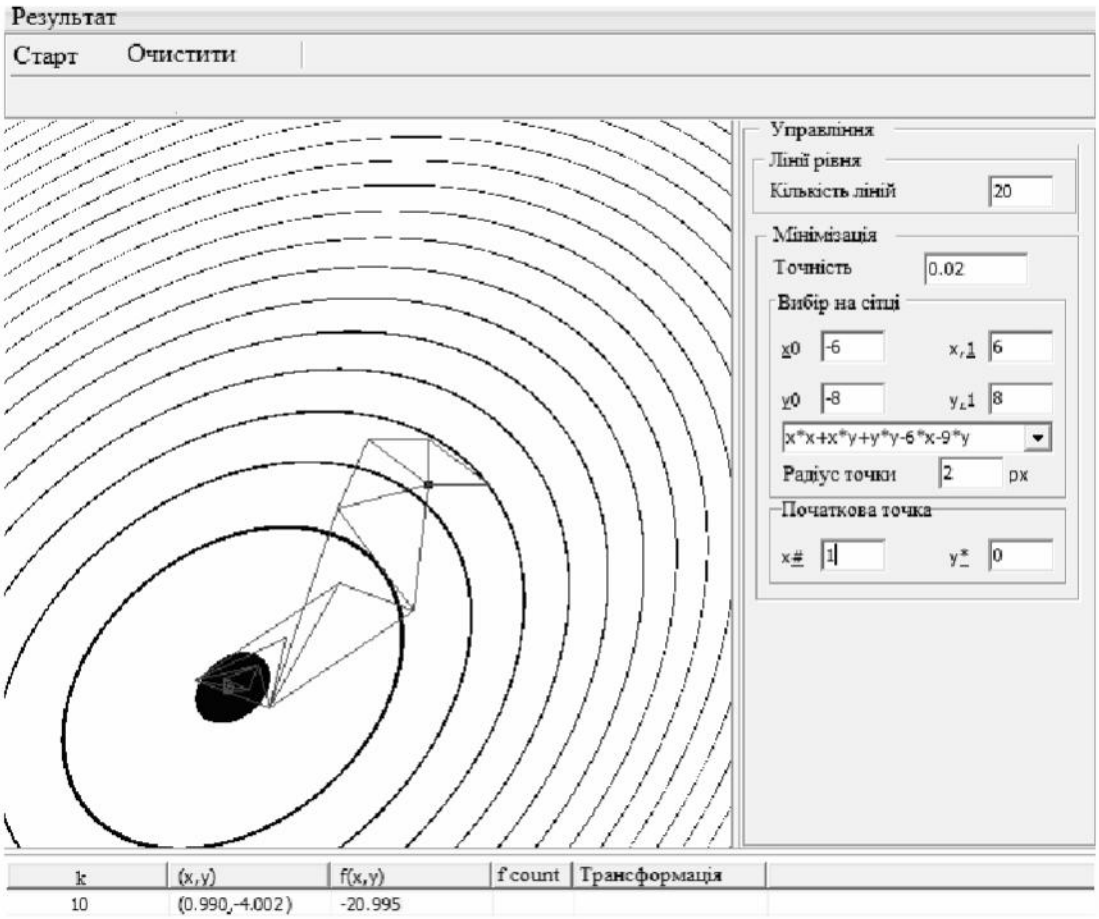


Рисунок 3.2 — Результати роботи програми

При натисканні кнопки “Старт”, з’являється інформація з поточним номером ітерації k, кількістю розрахунків функції $f(x)$, координатами вершин амеб $(x_1,\ldots,x_n)$, мінімальним значенням функції min $f(x)$ та виконаною трансформацією амеби (параметри віддзеркалення, стиснення,

розширення, редукції). На графіку функції відображається знайдена точка екстремуму. Для зручнішої роботи з графіком $f(x)$ передбачено масштабування координатної сітки. Його доцільно використовувати для того, щоб оцінити похибку пошуку екстремуму візуально. Кнопка "Очистити" використовується для очищення усіх полів та формування нового завдання користувачем. Якщо у програмному інтерфейсі натиснути вкладку "Результат", то можна вибрати два пункти — "Зберегти у файл" (формат *.*txt*) та "Зберегти зображення" (формат *.*bmp*).

3.3 Експерименти

Ефективність роботи метаевристики рекомендовано проводити за точністю ξ (значення цільової функції, які менші заданого значення, вважаються глобальним екстремумом), кількістю виконаних ітерацій k та обчислень цільової функції c (швидкість збіжності), надійністю (властивість алгоритму приводити до оптимуму при багаторазовому повторенні пошуку з різних початкових точок), що, в свою чергу, впливає на час розрахунків t. Після чого можна проаналізувати результати роботи методу та визначити доцільність його застосування у даних умовах. Параметри запуску програми вибрано такими:

– розмірність простору пошуку, $n = \{2 \ldots 25\}$;

– кількість амеб, $r = 25$;

– перша вершина X_0 амеби вибирається із області пошуку оптимуму функції $f(x)$ випадково;

– довжина сторони амеби, $L = 1$;

– параметри трансформації амеби:

а) віддзеркалення, $\alpha = 1$;

б) розширення, $\gamma = 2$;

в) стиснення, $\beta = 0{,}5$;

г) редукція, $\delta = 0{,}5$;

– критерій зупинки: час обчислень на запуск програми, t = 600 с або s = 10 рестартів амеби;

– кількість тестових запусків програми, p = 25.

Використано набір із трьох тестових функцій, на якому були проведені експерименти по пошуку оптимуму [21].

1. Функція "капелюх" (рисунок 3.3). Мультимодальна зі складним рельєфом, складність полягає в тому, що навколо глобального оптимуму, по колу розташовані локальні мінімуми, які метод оптимізації може прийняти за глобальний екстремум:

$$f(x_1, x_2) = \frac{1 - \sin\left(\sqrt{x_1^2 + x_2^2}\right)^2}{1 + 0{,}001 \cdot (x_1^2 + x_2^2)},\ x_i \in [-10; 10],\ i = \overline{1, n},\ n = 2. \qquad (3.1)$$

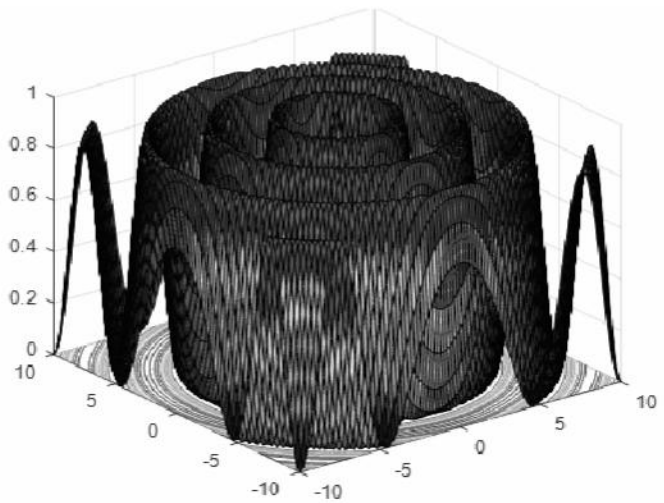


Рисунок 3.3 — Тест-функція 1

Має глобальний максимум $f(x_i^*) = f(0;\ 0) = 1$, або відповідний мінімум $-f(x_i^*) = -f(0;\ 0) = -1$.

2. Функція Захарова (рисунок 3.4). Має глобальний мінімум $f(x_i^*) = f(0,\ldots,0) = 0$, який розташовано на дуже довгій рівнині, саме через яку і вважається складною для оптимізації:

$$f(x_1,...,x_n) = \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=1}^{n} 0.5 \cdot i \cdot x_i\right)^2 + \left(\sum_{i=1}^{n} 0.5 \cdot i \cdot x_i\right)^4,\ x_i \in [-15;15],\ i = \overline{1,n}. \quad (3.2)$$

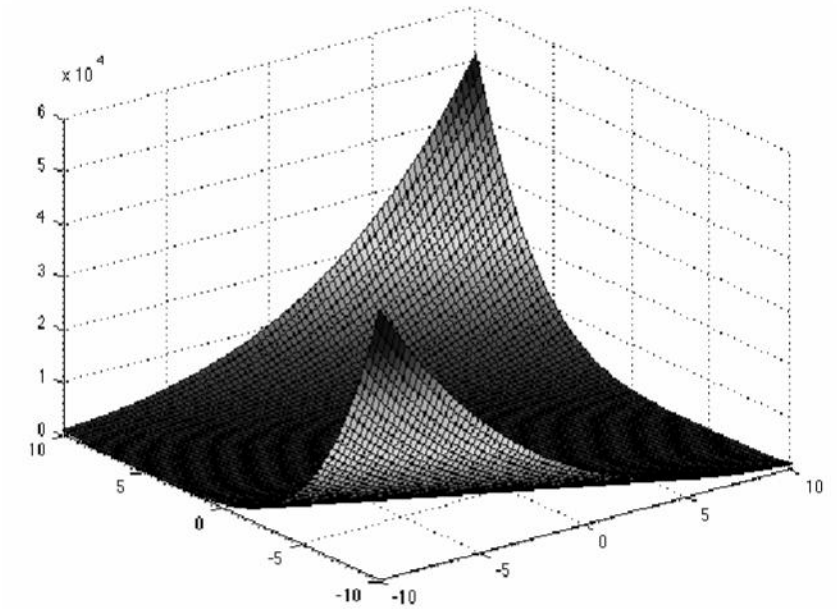


Рисунок 3.4 — Тест-функція 2

3. Функція Гріванка (рисунок 3.5). Має складний рельєф, є мультимодальною та вважається складною для оптимізації. Вона використовує косинусну модуляцію для створення багатьох локальних мінімумів, які розташовані на сітці з певною регулярністю (навколо глобального мінімуму знаходиться дуже багато точок локального мінімуму). Глобальний мінімум $f(x_i^*) = f(0,\ldots,0) = 0$:

$$f(x_1,...,x_n) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right),\ x_i \in [-16; 16],\ i = \overline{1,n}. \qquad (3.3)$$

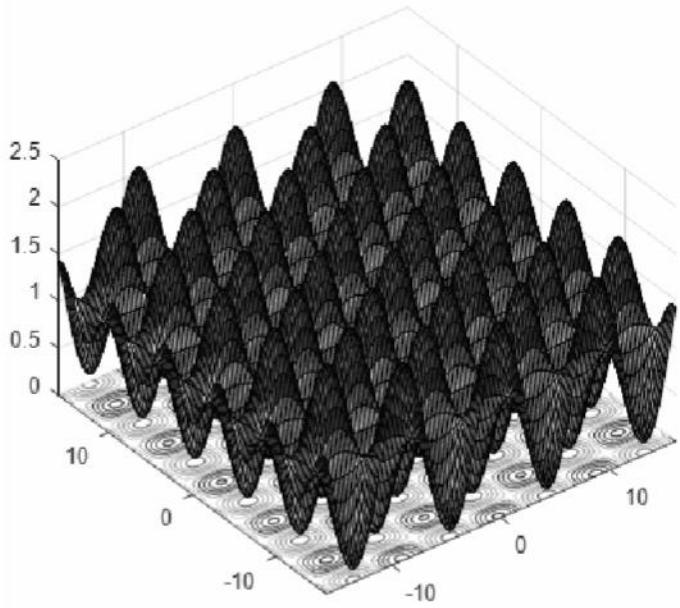


Рисунок 3.5 — Тест-функція 3

Результати експериментальних досліджень наведено в таблиці 3.2, яка відображає отримані мінімальні значення екстремуму функції, середні кількості обчислень функції та часу пошуку оптимуму за 50 запусків програми, фіксація ітерації розв'язку, якщо точність 10^{-5}. Результати обчислень у графічному вигляді проілюстровано на рисунках 3.6 та 3.7, де наведено середні кількості ітерацій та обчислень цільової функції при досягненні алгоритмом оптимуму за заданими умовами.

Результати роботи алгоритму амеби показали його придатність для оптимізації функції багатьох змінних. Особливістю алгоритму є велика кількість перевірок та звернень до цільової функції, що особливо проявляється у випадку функції Гріванка і пояснюється специфікою рельєфу функції та відповідними стрибками.

Таблиця 3.2 – Результати експериментів на тестових функціях

Тестова функція	Розмірність задачі	Мінімальне значення функції	Усереднені результати	
			Кількість ітерацій	Кількість обчислень функції
1	2	0,999997	95,545	24003,533
2	2	0,000000	85,314	21842,427
	10	$2{,}337 \cdot 10^{-10}$	756,754	187599,317
	25	$4{,}124 \cdot 10^{-7}$	1563,346	391183,563
3	2	0,000000	466,081	117079,313
	10	$8{,}325 \cdot 10^{-7}$	1694,651	420205,018
	25	$8{,}325 \cdot 10^{-6}$	2824,345	706713,256

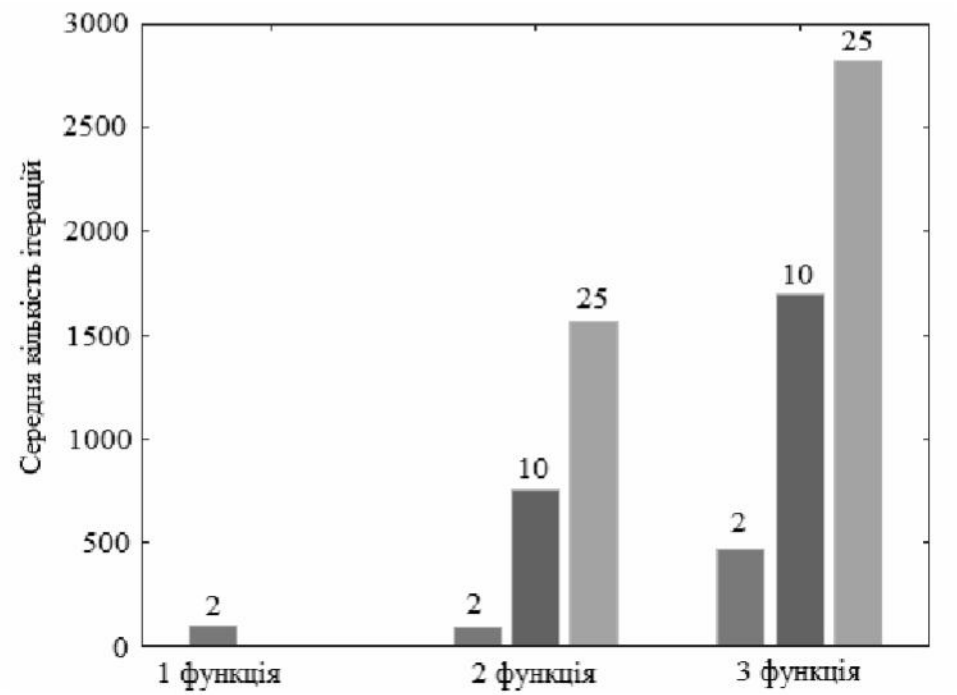


Рисунок 3.6 — Порівняння кількості ітерацій

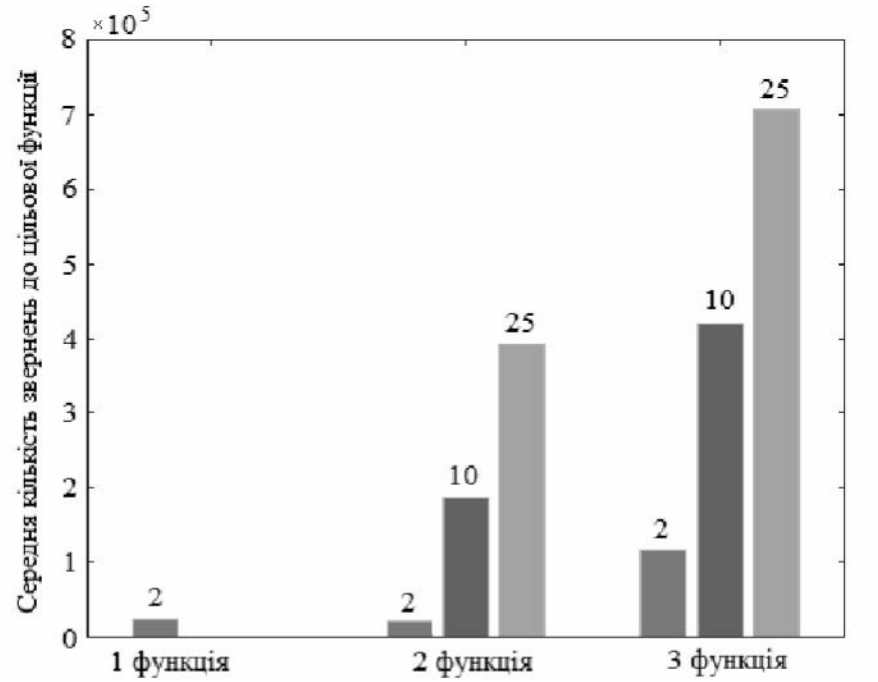


Рисунок 3.7 — Порівняння кількості обчислень функції

Для ілюстрації практичного застосування алгоритму амеб розглянемо інженерну задачу оптимізації тристержневої ферми [21]. Розглядається сталева конструкція, що складається з витягнутих елементів, які розташовано у формі трикутника для утримання навантажень за схемою з рисунку 3.8.

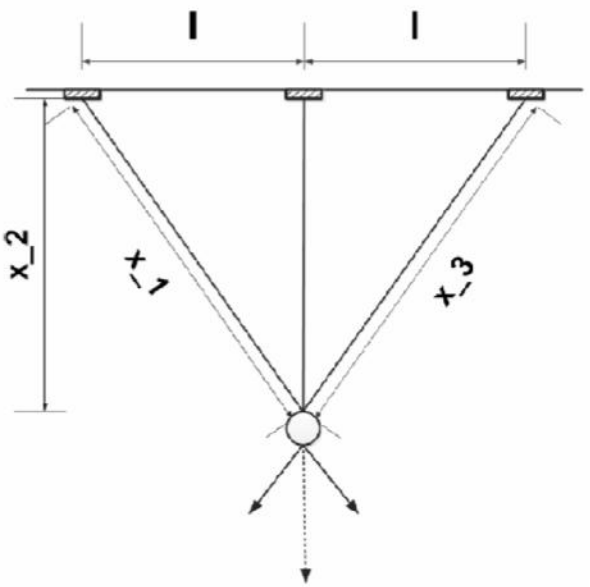


Рисунок 3.8 – Конструкція тристержневої ферми

Елементи x_1 і x_3 мають однакову довжину, оскільки це сторони трикутника, тоді як елемент x_2 розміщений у центрі конструкції. До вершини трикутника прикріплено навантаження, які задають обмеження на змінні x_1, x_2, x_3 та гарантують функціональність конструкції. Цільова функція $f(X) = f(x_1, x_2)$ визначає загальну вагу цієї конструкції, яку необхідно оптимізувати враховуючи наступні рівняння:

$$f(x_1, x_2) = (2\sqrt{2}x_1 + x_2) \cdot L \to \min, \tag{3.4}$$

$$\begin{cases} g_1(x_1, x_2) = \dfrac{\sqrt{2} \cdot x_1 + x_2}{\sqrt{2} \cdot x_1^2 + 2 \cdot x_1 \cdot x_2} \cdot P - \sigma \le 0; \\ g_2(x_1, x_2) = \dfrac{x_2}{\sqrt{2} \cdot x_1^2 + 2 \cdot x_1 \cdot x_2} \cdot P - \sigma \le 0; \\ g_3(x_1, x_2) = \dfrac{1}{x_1 + \sqrt{2} \cdot x_2} \cdot P - \sigma \le 0; \\ 0 \le x_1, x_2 \le 1, \end{cases} \tag{3.5}$$

де L = 100 см; $P = \sigma$ = 2 кН/см^2.

Для знаходження безумовного екстремуму задачі умовної оптимізації необхідно перетворити задану функцію. Для цього застосовано метод штрафів [12], ідея якого полягає в отриманні нової цільової функції $F(X)$, $X = (x_1, ..., x_n)^T$ для області пошуку допустимих рішень за рахунок введення у цільову функцію $f(X)$ вибраної спеціальним чином функції штрафу $P(X)$. Для розв'язання представленої задачі використовуємо метод штрафів (μ_k – константи штрафу; $h = 1$, **якщо** $g(X) > 0$ **і** $h = 0$, інакше) в адитивній формі:

$$F(X) = f(X) + P(X) = f(X) + \sum_{k=1}^{3} \mu_k \cdot g_k^2(X) \cdot h_k \to \min. \tag{3.6}$$

Пошуком оптимального розв'язку даної задачі займалися багато дослідників, застосовуючи різноманітні метаевристики. Порівняння результатів показано в таблиці 3.3.

Таблиця 3.3 – Порівняння результатів роботи алгоритмів

Дослідники	Алгоритм	Розв'язок			Обмеження
		x_1	x_2	f^{best}	g_1, g_2, g_3
P. J. van Laarhoven, E. H. Aarts [22]	Алгоритм імітації відпалу	0,81023	0,63428	292,59565	-0,18004, -1,35152, -0,82851
X. S. Yang [23]	Алгоритм на основі поведінки світляків	0,78489	0,66347	288,34741	-0,14561, -1,30625, -0,83935
T. Ray, P. Saini [24]	Ройовий алгоритм з інтелектуальним обміном інформацією	0,79500	0,39500	264,35996	$-1{,}05\cdot10^{-7}$, -1.46429, -0.53570
C. Gao, Z. Hu, W. Tong [25]	Еволюційний алгоритм із лінійним прогнозуванням	0,78867	0,40825	263,89571	$-5{,}09\cdot10^{-7}$, -1,46411, -0,53591
Дана робота	Ройовий алгоритм амеби з мультистартами	0,78873	0,40808	263,89588	$-1{,}05\cdot10^{-7}$, -1.46429, -0.53570

Алгоритм амеб отримав наступне рішення задачі за 50 запусків (30 хв кожний) програми та 30 рестартів амеб, 25 агентів популяції: $f(0,78873;0,40808)=263,89588$. Слід зазначити, що у випадку однієї амеби рішення не було знайдене, а результат немає змісту порівнювати з науковими роботами. Ефективність алгоритму залежить від генерації стартових положень особин, кількості обчислень цільової функції, також необхідно розробляти процедуру для організації обміну інформацією амеб у процесі пошуку. Враховуючи ці недоліки, на сьогоднішній день відомі результати з кращим розв'язком тестової задачі.

3.4 Висновки

У розділі описано розроблене програмне забезпечення та наведено значення параметрів, які були використані у експериментальних дослідженнях ефективності роботи алгоритму. Тестування програмного забезпечення проведено на низці задач. Результати обчислень підтвердили коректність роботи програмного забезпечення.

ВИСНОВКИ

Система рівнянь, яка отримується в результаті застосування класичного методу пошуку екстремуму, забезпечує лише необхідні умови оптимальності, крім того, вона може бути нелінійною та складною. Якщо задачу не можна розв'язати детермінованими методами оптимізації, застосовують ітеративні схеми – метаевристики.

Тому у роботі проведено дослідження ефективності евристичного алгоритму амеби, у якому використано вагові коефіцієнти для того, щоб центр тяжіння амеби був ближчим до тієї вершини, для якої ваговий коефіцієнт більший. Подібна модифікація змінює напрямок пошуку нової вершини амеби, що підвищує ефективність алгоритму. У результаті розроблено програмне забезпечення та проведено тестування алгоритму на наборі тестових функцій та практичній задачі оптимізації. Порівняння результатів роботи з іншими метаевристиками показало, що алгоритм може бути застосований на практиці, також є можливість його модифікації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Жалдак М.І., Триус Ю.В. Основи теорії і методів оптимізації. Черкаси: Брама-Україна, 2005. 608 с.

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

3. 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. pp. 220–250.

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

5. 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.

6. Takenaga S., Ozaki Y., Onishi M. Practical Initialization of the Ameoba Algorithm for Computationally Expensive Optimization Problems. *Optimization Letters*. 2023. № 17. pp. 1–15.

7. An Ameoba-Infused INFO Algorithm for Optimization of Mechanical Design Problems / P. Mehta, B.S. Yildiz, S.P. Kumar et. al. *Materials Testing*. 2022. Vol. 64. pp. 1172–1182.

8. Zhou L., Zhou X., Yi C. A Hybrid STA Based on Nelder–Mead Simplex Search and Quadratic Interpolation. *Electronics*. 2023. Vol. 12. 26 p. URL: https://www.mdpi.com/2079-9292/12/4/994 (дата звернення 20.04.2025).

9. Максимець Д.В., Тарабаров С.Б. Модифікація симплексного методу Нелдера і Міда. *Міжнародна науково-технічна конференція «Радіотехнічні поля, сигнали, апарати та системи»*. К.: Національний технічний університет України КПІ імені І. Сікорського, 2016. С. 75–77.

10. Адаптивні ройові алгоритми оптимізації / А.Р. Корнєєва, Ю.Ю. Іванов, С.Г. Кривогубченко та інші. *LIV Всеукраїнська науково-технічна конференція підрозділів Вінницького національного технічного університету.* Вінниця, 24-27.03.2025. URL: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2025/paper/view/23529/19765. (дата звернення 10.04.2025)

11. Комп'ютерне моделювання систем та процесів. Методи обчислень. Частина 2 / Р.Н. Квєтний, І.В. Богач, О.Р. Бойко, О.Ю. Софина, О.М. Шушура. Вінниця: ВНТУ, 2013. 234 с.

12. Chapra S.C., Canale R.P. Numerical Methods for Engineers. New-York: McGraw-Hill, 2010. 994 с.

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

14. Sebastian U.S., Anant R., Martin J. Approximate Steepest Coordinate Descent. *Proceedings of the 34th International Conference on Machine Learning*. Australia, 2017. 15 p.

15. Gao F., Han L. Implementing the Ameoba Algorithm with Adaptive Parameters. *Computer Optimization Applications*. 2012. Vol. 51. pp. 259–277.

16. James McCaffrey: Swarm Intelligence Optimization using Python. URL: https://www.youtube.com/watch?v=bVDX_UwthZI (дата звернення 27.03.2025).

17. A New Hybrid Harris Hawks-Nelder-Mead Optimization Algorithm for Solving Design and Manufacturing Problems / S. Mirjalili, A. Yildiz, S. Sait, S. Bureerat, N. Pholdee. *Materialprufung*. 2019. № 8. pp. 60–70.

18. Zitouni F., Harous S. Integrating the Opposition Nelder–Mead Algorithm into the Selection Phase of the Genetic Algorithm for Enhanced Optimization. *Applied System Innovation*. 2023. Vol 6. 31 p. URL: https://www.mdpi.com/2571-5577/6/5/80 (дата звернення 20.04.2025).

19. Python documentation. URL: https://docs.python.org/3/index.html (дата звернення 04.04.24).

20. Рейтинг мов програмування. URL: https://javarush.com/ua/groups/posts/uk.1908.reyting-mov-programuvannja (дата звернення 04.04.24).

21. Global Optimization Benchmarks. URL: http://infinity77.net/global_optimization/index.html (дата звернення 24.03.2025).

22. Van Laarhoven P.J., Aarts E.H. Simulated Annealing: Theory and Applications. Springer, 1987. 198 p.

23. Yang X.S. Firefly Algorithm, Stochastic Test Functions and Design Optimization. *International Journal of Bio-Inspired Computation*. 2010. №2. pp. 78–84.

24. Ray T., Saini P. Engineering Design Optimization Using a Swarm with an Intelligent Information Sharing Among Individuals. *Journal of Engineering Optimization*. 2007. Vol. 33. pp. 735–748.

25. Gao C., Hu Z., Tong W. Linear Prediction Evolution Algorithm: a Simplest Evolutionary Optimizer. *Memetic Computing*. 2021. № 13. pp. 319–339.

ДОДАТКИ

Додаток А

(обов'язковий)

Технічне завдання

ЗАТВЕРДЖУЮ

Завідувач кафедри АІІТ

д.т.н., проф. Бісікало О. В.

«23» 03 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на бакалаврську кваліфікаційну роботу

«Дослідження алгоритму амеби з мультистартами»

08-31.БКР.009.02.000 ТЗ

Керівник роботи:

к.т.н., доц. каф. АІІТ

Кривогубченко С. Г.

«24» 03 2025 р.

Виконавець:

ст. гр. 1ІСТ-21б

Корнєєва А. Р.

«24» 03 2025 р.

Вінниця ВНТУ – 2025

1. Назва та галузь застосування

Розробка програмного забезпечення для розв'язання задачі оптимізації функцій алгоритмом амеби з мультистартами. Задачі оптимізації, які потрібно вирішувати на етапі попереднього оцінювання та проєктування складних систем, оптимального управління, аналізу даних тощо.

2. Підстави для розробки

Розробку здійснювати на підставі наказу по університету №97 від 20.03.2025 та завдання до бакалаврської кваліфікаційної роботи, складеного та затвердженого кафедрою "Автоматизації та інтелектуальних інформаційних технологій".

3. Мета та призначення розробки

Метою роботи є розширення функціональних можливостей програмного забезпечення для розв'язання задач неперервної оптимізації функцій за рахунок використання ройового алгоритму амеби з мультистартами.

4. Етапи виконання робіт

Виконавцем перерахованих у даному розділі етапів є студентка групи 1ІСТ-21б Корнєєва Анна Ростиславівна факультету Інтелектуальних інформаційних технологій та автоматизації Вінницького національного технічного університету, а замовником є кафедра автоматизації та інтелектуальних інформаційних технологій.

№	Назва та зміст етапу	Термін виконання	
		початок	закінчення
1	Аналіз предметної області	21.03.2025	04.04.2025
2	Розробка математичної моделі ройового алгоритму амеби	05.04.2025	26.04.2025
3	Розробка програмного забезпечення та експериментальні дослідження	27.04.2025	16.05.2025
4	Оформлення пояснювальної записки і графічного матеріалу	17.05.2025	27.05.2025

5. Джерела розробки

1. An Ameoba-Infused INFO Algorithm for Optimization of Mechanical Design Problems / P. Mehta, B.S. Yildiz, S.P. Kumar et. al. *Materials Testing*. 2022. Vol. 64. pp. 1172–1182.

2. Takenaga S., Ozaki Y., Onishi M. Practical Initialization of the Ameoba Algorithm for Computationally Expensive Optimization Problems. *Optimization Letters*. 2023. № 17. pp. 1–15.

3. Zhou L., Zhou X., Yi C. A Hybrid STA Based on Nelder–Mead Simplex Search and Quadratic Interpolation. *Electronics*. 2023. Vol. 12. 26 p. URL: https://www.mdpi.com/2079-9292/12/4/994 (дата звернення 20.04.2025).

6. Показники призначення

Основні технічні вимоги та мінімальні системні вимоги до програми: операційна система *Windows* 7, 8 або 10; підключення клавіатури та миші; процесор *Core* i5 (*Ryzen* 5) або кращий; оперативна пам'ять – не менше 8 ГБ; відеопам'ять – 512 Mb; жорсткий диск – 500 ГБ і більше.

Вхідні дані: розмірність простору пошуку; кількість особин; довжина ребра та координати вершин амеби; параметри віддзеркалення, стиснення, розширення, редукції; кількість рестартів; набір тестових функцій. Результати роботи програми: оптимальне значення функції та відповідні параметри.

Студентка гр. 1ІСТ-21б _____ Корнєєва А. Р.

(підпис) (ініціали та прізвище)

Додаток Б

(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

ДОСЛІДЖЕННЯ АЛГОРИТМУ АМЕБИ З МУЛЬТИСТАРТАМИ

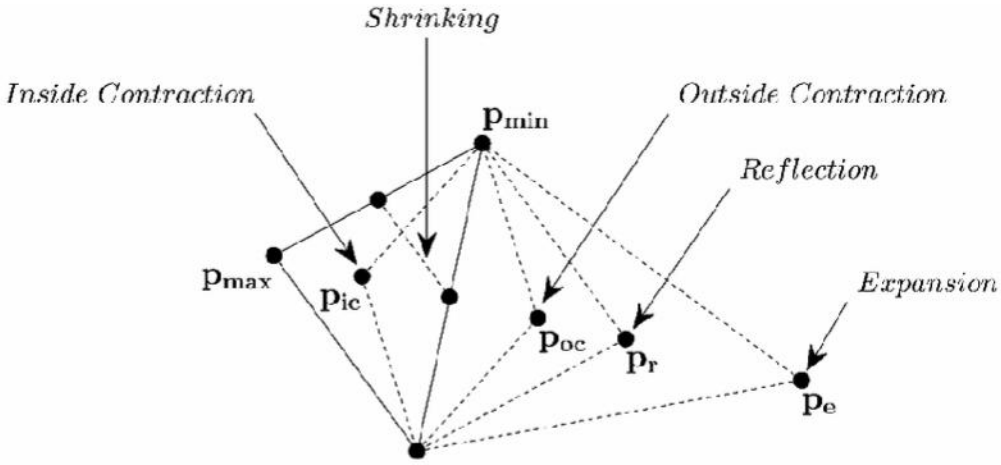


Рисунок Б.1 – Можливі варіанти переміщення амеб

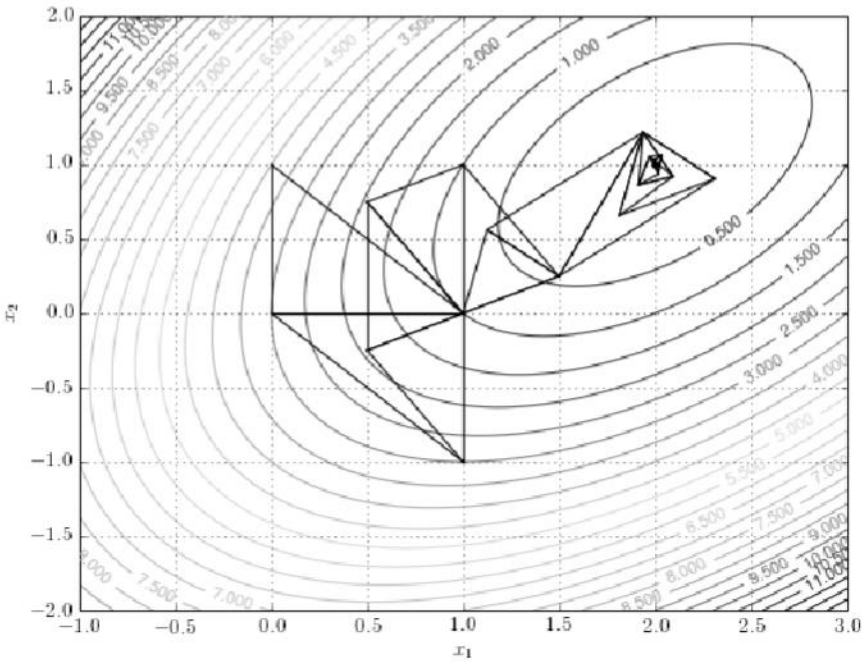


Рисунок Б.2 — Приклад пошуку однією амебою

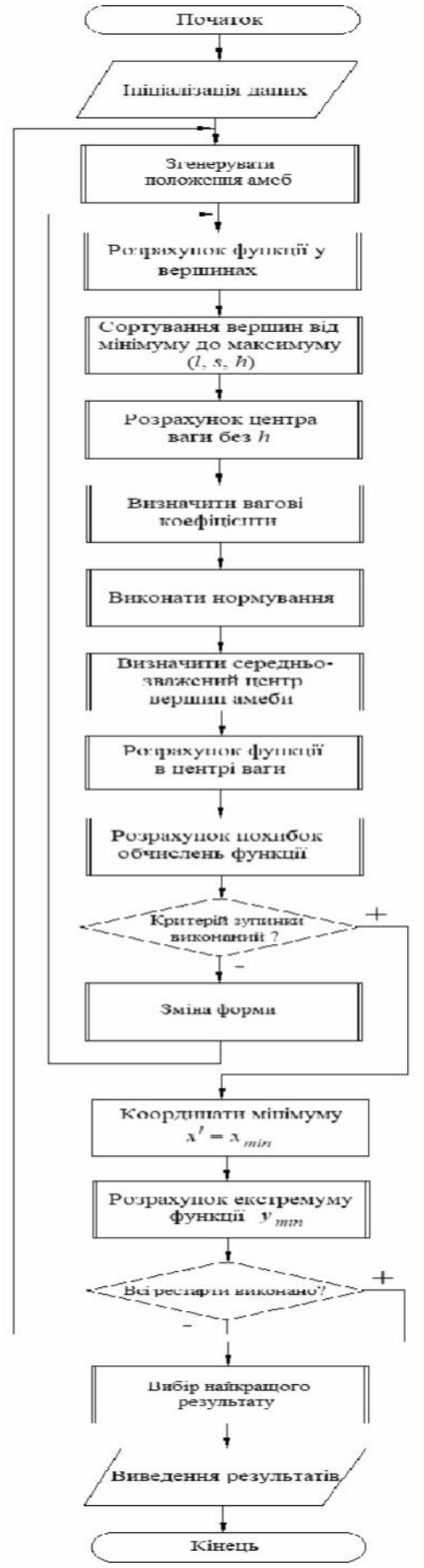


Рисунок Б.3 – Схема програми

Додаток В

(обов'язковий)

Лістинг програмного забезпечення

```
#################################################################
def ameoba(f, x0, tol=1e-8, maxit=1e4):
    init_guess = x0
    fx0 = f(x0)
    dist = 10.
    curr_it = 0
    n = np.size(x0)
    alpha = 1.
     beta = 2.
     gamma = .5
     delta = .5
    simplex_points = np.empty((n+1, n))
    pt_fval = [(x0, fx0)]
    simplex_points[0, :] = x0

    for ind, elem in enumerate(x0):
        if np.abs(elem) < 1e-14:
            curr_tau = 0.00025
        else:
            curr_tau = 0.05

        curr_point = np.squeeze(np.eye(1, M=n, k=ind)*curr_tau + x0)
        simplex_points[ind, :] = curr_point
        pt_fval.append((curr_point, f(curr_point)))

    if iter_returns is not None:
        ret_points = []
    else:
        ret_points = None

    while dist > tol and curr_it < maxit:
        pt_fval = sorted(pt_fval, key=lambda v: v[1])
        xbar = x0*0

        for i in range(n):
            xbar = xbar + (pt_fval[i][0])/(n)
```

```
if iter_returns is not None and curr_it in iter_returns:
    ret_points.append(pt_fval)

x1, f1 = pt_fval[0]
xn, fn = pt_fval[n-1]
xnp1, fnp1 = pt_fval[n]

xr = xbar + alpha*(xbar - pt_fval[-1][0])
fr = f(xr)
if f1 <= fr < fn:
    xnp1, fnp1 = (xr, fr)
    pt_fval[n] = (xnp1, fnp1)
elif fr < f1:
    xe = xbar + beta*(xr - xbar)
    fe = f(xe)
    if fe < fr:
        xnp1, fnp1 = (xe, fe)
        pt_fval[n] = (xnp1, fnp1)
    else:
        xnp1, fnp1 = (xr, fr)
        pt_fval[n] = (xnp1, fnp1)
elif fn <= fr <= fnp1:
    xoc = xbar + gamma*(xr - xbar)
    foc = f(xoc)
    if foc <= fr:
        xnp1, fnp1 = (xoc, foc)
        pt_fval[n] = (xnp1, fnp1)
    else:
        for i in range(1, n+1):
            curr_pt, curr_f = pt_fval[i]
            new_pt = x1 + delta*(curr_pt - x1)
            new_f = f(new_pt)
            pt_fval[i] = new_pt, new_f
elif fr >= fnp1:
    xic = xbar - gamma*(xr - xbar)
    fic = f(xic)
    if fic <= fr:
        xnp1, fnp1 = (xic, fic)
        pt_fval[n] = (xnp1, fnp1)
    else:
        for i in range(1, n+1):
            curr_pt, curr_f = pt_fval[i]
            new_pt = x1 + delta*(curr_pt - x1)
            new_f = f(new_pt)
            pt_fval[i] = new_pt, new_f
```

```
    dist = abs(fn - f1)
    curr_it = curr_it + 1

if curr_it == maxit:
    raise ValueError("Max iterations; Convergence failed.")

if ret_points:
    return x1, f1, curr_it, ret_points
else:
    return x1, f1, curr_it

x = np.linspace(-2.5, 2.5, 500)
y = np.linspace(-2.5, 2.5, 500)
X, Y = np.meshgrid(x, y)
Z = rosenbrock([X, Y])
fig = plt.figure(figsize=(14, 8))
ax1 = fig.add_subplot(121)
ax2 = fig.add_subplot(122, projection="3d")
fig.suptitle("Rosenbrock Function", size=24)
ax1.set_axis_bgcolor("white")
ax1.pcolormesh(X, Y, Z, cmap=matplotlib.cm.viridis,
            norm=LogNorm())
ax1.scatter(1, 1, color="k")
ax1.annotate('Global Min', xy=(1, 1), xytext=(-0.5, 1.25),
         arrowprops=dict(facecolor='black', shrink=0.05))
ax2.set_axis_bgcolor("white")
ax2.plot_surface(X, Y, Z, norm = LogNorm(), cmap=matplotlib.cm.viridis,
             linewidth=0)
ax2.view_init(azim=65, elev=25)
ax2.scatter(1., 1., 0., color="k")
xa, ya, _ = proj3d.proj_transform(1,1,0, ax2.get_proj())
ax2.annotate("Global Min", xy = (xa, ya), xytext = (-20, 30),
    textcoords = 'offset points', ha = 'right', va = 'bottom',
    arrowprops=dict(facecolor='black', shrink=0.05))
plt.tight_layout()
plt.show()
iterstosee = [0, 1, 2, 3, 4, 5, 10, 15, 30, 45, 75, 90]
x, fx, its, ret_tris = nelder_mead(rosenbrock, x0=np.array([-1.5, -1.]),
tol=1e-12, iter_returns=iterstosee)
fig, axs = plt.subplots(nrows=6, ncols=2, figsize=(16, 24))
axs = axs.flatten()
```

```
    for i, curr_ax in enumerate(axs):
        curr_simplex = np.vstack([ret_tris[i][0][0], ret_tris[i][1][0], ret_tris[i][2][0]])
        curr_ax.pcolormesh(X, Y, Z, cmap=matplotlib.cm.viridis,
                 norm=LogNorm())
        curr_ax.set_title("This is simplex for iteration %i" %iterstosee[i])
        curr_ax.scatter(curr_simplex[:, 0], curr_simplex[:, 1])
    plt.tight_layout()
    plt.show()

def rosenbrock(x, a=1, b=100):
    y = x[1]
    x = x[0]
    return (a - x)**2 + b*(y - x**2)**2
##############################################################################
```

Додаток Г

(обов’язковий)

Протокол перевірки БКР

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Дослідження ройового алгоритму амеби з мультистартами.

Тип роботи: бакалаврська кваліфікаційна робота (БКР)
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ: кафедра автоматизації та інтелектуальних інформаційних технологій, факультет інтелектуальних інформаційних технологій та автоматизації, ІІСТ-21б
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 1,18 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне):

- ■ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту.
- □ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.
- □ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Зав. каф. АІІТ, д.т.н., професор Бісікало О. В. _____ (підпис)
(прізвище, ініціали, посада)

Секретар кафедри АІІТ Янчук Л. М. _____ (підпис)
(прізвище, ініціали, посада)

Особа, відповідальна за перевірку _____ (підпис) Овчинников К. В.
(прізвище, ініціали)

З висновком експертної комісії ознайомлена.

Керівник _____ (підпис) доцент Кривогубченко С. Г.
(прізвище, ініціали, посада)

Здобувач _____ (підпис) Корнєєва А. Р.
(прізвище, ініціали)