Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інфокомунікацій
(повна назва)

Кафедра Інформаційно-мережної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіті другий (магістерський)

Дослідження стохастичних методів оптимізації

(тема)

Виконав:
здобувач 2 року навчання,
групи ІМІм-23-2
Пугач К.О.
(прізвище, ініціали)

Спеціальність 172 Електронні комунікації та радіотехніка
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Інформаційно-мережна інженерія
(повна назва освітньої програми)

Керівник: доц. Омельченко С.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Безрук В.М.
(підпис) (прізвище, ініціали)

2025 р.

Не містить відомостей, заборонених до відкритого публікування

Студент _____ / Пугач К.О. /
(підпис) (прізвище та ініціали)

Керівник _____ / Омельченко С.В. /
(підпис) (прізвище та ініціали)

Харківський національний університет радіоелектроніки

Факультет Інфокомунікацій

Кафедра Інформаційно-мережної інженерії

Рівень вищої освіти другий (магістерський)

Спеціальність 172 Електронні комунікації та радіотехніка

(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформаційно-мережна інженерія

(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
« 28 » жовтня 2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Пугач Катерині Олегівні
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження стохастичних методів оптимізації

затверджена наказом по університету від« 28» жовтня 2024 р. № 1148 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 23 січня 2025 р.

3. Вхідні дані до роботи Дослідити сутність, завдання та види поширених методів оптимізації у машинному навчанні. Розглянути специфіку, переваги та недоліки стохастичних методів. Обрати для дослідження ряд методів опттмізації стохастичної групи. Дослідити кожен з методів. Для одного з методів розглянути програмну реалізацію. Пропонується обрати метод відпалу, алгоритм косяку риб та групу бджолиних алгоритмів.

4. Перелік питань, що потрібно опрацювати у роботі Вступ

1 . Методи оптимізації

2. Алгоритм імітації відпалу

3. Методи стохастичної оптимізації на основі бджолиних алгоритмів

4. Алгоритм косяка риб

Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) слайди презентації в форматі Power Point (назва та мета роботи, методи оптимізації, алгоритм відпалу, бджолині алгоритми, алгоритм косяка риб, висновки)

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Вступ		виконано
2	Методи оптимізації		виконано
3	Алгоритм імітації відпалу		виконано
4	Методи стохастичної оптимізації на основі бджолиних алгоритмів		виконано
5	Алгоритм косяка риб		виконано
6	Висновки		виконано
7	Оформлення пояснювальної записки		виконано

Дата видачі завдання 28 жовтня 2024 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Омельченко С.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 72 с., 11 рис., 18 джерел, 2 додатка.

Об'єкт дослідження – методи оптимізації, які функціонують за стохастичними принципами.

Мета роботи – дослідження сутності та особливотей побудови стохастичних методів оптимізації.

Досліджуються базові засади та класифікація методів оптимізації. Розглядаються особливості реалізації ряду поширених стохастичних алгоритмів – методу відпалу, бджолиних алгоритмів та алгоритму косяку риб. Досліджується програмна реалізація методу косяку риб.

СТОХАСТИЧНА ОПТИМІЗАЦІЯ, ЛОКАЛЬНИЙ ЕКСТРЕМУМ, ФУНКЦІЯ РОЗЕНБРОКА, МЕТОД ВІДПАЛУ, ВІДПАЛ КОШІ, АЛГОРИТМ КОСЯКА РИБ, IMPROVED BEE ALGORITHM.

THE ABSTRACT

Explanatory note: 72 p., 11 fig., 18 sources, 2 app.

The object of research is optimization methods that function according to stochastic principles.

The purpose of the work is to study the essence and features of building stochastic optimization methods.

The basic principles and classification of optimization methods are studied. Features of the implementation of a number of common stochastic algorithms are considered - the annealing method, bee algorithms, and the shoal of fish algorithm. The software implementation of the shoal of fish method is being studied.

STOCHASTIC OPTIMIZATION, LOCAL EXTREMUM, ROSENBROCK FUNCTION, ANNEALING METHOD, BASKET ANnealing, SCHOOL OF FISH ALGORITHM, IMPROVED BEE ALGORITHM.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

OSI – (Open System Interconnection) – семирівнева модель взаємодії відкритих систем;

LP – (Linear Programming) – лінійне програмування;

NLP – (Non-Linear Programming) – нелінійне програмування;

IP – (Integer Programming) – цілочисельне програмування;

CNLP – (Convex Non-Linear Programming) – опукле нелінійне програмування;

MINLP – (Mixed Integer Nonlinear Programming) – змішане цілочисельне нілінійне програмування;

FSS – (Fish School Search) – алгоритм косяку риб;

BA – (Bee Colony) – бджолиний алгоритм;

ABC – (Artificial Bee Colony Algorithm) – алгоритм бджолиного рою (колонії);

IBA – (Improved Bee Algorithm) – покращений бджолиний алгоритм.

ВСТУП

Останнім часом спостерігається тенденція щодо постійного збільшення кількості об’єктів інфокомунікаційного простору, таких, як,

– користувачів індивідуального та корпоративного типу;

– інформаційно-комунікаційних сервісів різного формату.

Окрім цього, фіксується також постійне ускладнення зав’язків між такими об’єктами.

На тлі цього все більш гостро постає питання забезпечення необхідного рівня надаваних мережевих послуг, що стосується:

– якості обробки клієнтських транзакцій (ураховуючи час обробки та її результативність);

– забезпечення сталого сервісу в умовах мережевого перевантаження, викликаного різними чинниками – як у ході роботи сервісів згідно існуючого регламенту, так і під час обробки клієнтських запитів в позаштатному режимі;

– гарантування доставки пакетів сервісів відносно їхнього пріоритету з позиції затримки та відсотку ймовірних втрат тощо.

У сучасних умовах, беручи до уваги значну кількість як самих сервісів, так і їх користувачів, гарантування можливість надання мережевих послуг відповідно до попередньо встановлених вимог є актуальною науково-прикладною задачею.

Для рішення означеної задачі може бути використано численні методи оптимізації [1-3].

Такі методи розглядають будь-який мережевий процес, на будь-якому рівні моделі EMBBC, або, навіть, на кількох рівнях одночасно, як математичну цільову функцію, яка має свої екстремуми – мінімуми, або максимуми. У рамках цього також розглядається супутня функція – функція втрат, або помилок.

Далі, керуючись означеним підходом, завдання оптимізації функціонування того чи іншого сервісу, або його окремого функціоналу, може розглядатися, як завдання пошуку екстремуму (або екстремумів) функції за умови мінімізації відповідної функції втрат.

При цьому, слід взяти до уваги те, що певний ряд процесів, що мають перебіг у мережі, є, з одного боку, достатньо складними, а з іншого боку –

такими, що не можуть бути описаними єдиною математичною функцією як у певних умовах, так і без огляду на будь-які умови.

Відтак, очевидним є те, що оптимізація таких процесів вимагає застосування методів, які будуть ураховувати специфіку завдань оптимізації, та є апріорі ефективними для заданих умов. Такими методами, у свою чергу. є стохастичні методи оптимізації. Це і зумовлює актуальність теми кваліфікаційної роботи.

1 МЕТОДИ ОПТИМІЗАЦІЇ

1.1 Поняття оптимізації. Класифікація методів оптимізації з використанням різних критеріїв

Термін «оптимізація» у загальному випадку може трактуватися як процес, який використовує один, або кілька механізмів, спрямованих на забезпечення максимізації/мінімізації цільової функції в умова мінімізації супутньої функції втрат F_{loss}. Аналітично це може бути представлено у наступному вигляді [4]:

$$\begin{cases} \mathrm{F(x)} \underset{\mathrm{opt}}{\rightarrow} \max\ (\min); \\ F_{loss} \rightarrow \min, \end{cases} \quad (1.1)$$

де F – цільова функція;

x – аргумент функції;

opt – механізм оптимізації;

F_{loss} – функція втрат.

Водночас, загальне завдання оптимізації може бути представлено великою кількістю різноманітних підкласів завдань.

У свою чергу, специфіка підкласу завдання може визначати загальну ефективність її рішення.

Власне, ймовірна класифікация задач визначається цільовою функцією, яка описує той чи інший процес, а також допустимою областю значень такої функції.

Відтак, відповідно до специфіки підзадач оптимізації, існуючі на сьогодні оптимізаційні методи може бути поділено на [4-7]:

1. Методи локального типу. Завдання, які вирішують методи даної групи, зводяться до пошуку локального экстремуму цільової функції F .

2. Методи глобального типу, які використовуються відносно цільових функцій, які можуть мати багато екстремумів.

Також методи оптимізації, які зараз існують і мають той чи інший рівень застосовуваності, також може бути розподілено на 3 групи, а саме:

– детерміновані методи;

– стохастичні методи;

– комбіновані методи.

Окрім зазначених критеріїв, також може бути виділено інші критерії, за якими виконується класифікація методів оптимізації, наприклад, такі, як зазначено далі [4, 8]:

1. Розмірність допустимої множини. Відповідно до означеного критерію, існуючі методи оптимізації розподіляються на:

- методи одномірної оптимізації;
- методи багатомірної оптимізації.

2. Вимоги щодо гладкості цільової функції, а також існування для цільової функції часткових похідних. Відповідно до даного критерію класифікації, існуючі методи оптимізації може бути поділено на:

− прямі методи. Такі методи вимагають розрахунку величини цільової функції лише виключно у точках наближень;

− методи першого порядку. Такі методи передбачають виконання обчислень перших часткових похудних цільової функції F;

− методи другого порядку. Методи даної групи вимагають розрахунку других часткових похідних цільової функції.

3. Спосіб рішення. У відповідності до означеного критерію, оптимізаційні методи може бути поділено на [6-8]:

– аналітичні (наприклад, умови Каруша-Куна-Таккера, метод множників Лагранжа і т.д.);

– численні методи;

– графічні методи.

Окрім зазначеного, методи оптимізації також розподіляються на:

– детерміновані;

– стохастичні.

Розглянемо окремо детерміновані та стохастичні методи оптимізації.

1.2 Детерміновані методи опттимізації

Детерміновані методи оптимізації вирішують завдання знаходження глобального найкращого результату [7, 8].

При цьому, у теорії надаються гарантії того, що результат оптимізації, який було отримано, є насправді глобально найкращим.

Для цього детерміновані алгоритми оптимізації оперують жорстко формалізованими та якомога точнішими функціями, які описують процес, що підлягає оптимізації.

Отже, звідси виходить, що детермінована оптимізація належить до повних, або жорстких класів алгоритмів згідно з існуючою класифікацією Ноймайєра: це «завершені» алгоритми, які здатні знаходити глобальні найкращі рішення, для чого їм необхідно витратити невизначено тривалі часові проміжки виконання.

Разом з тим, достатнє локальне найкраще рішення може бути знайти за деякий кінцевий час, з урахуванням наперед визначених допусків.

Проте, ефективність детермінованих алгоритмів може бути низькою у разі вирішення завдань на кшталт «чорної скриньки», або коли процес описується надзвичайно складними та мінливими функціями.

Зараз існує ряд класичних алгоритмічних моделей, які слугують для реалізації алгоритмів детермінованої оптимізації.

Серед найпоширеніших, з них може бути виокремлено моделі лінійного програмування (LP), а також - моделі нелінійного програмування (NLP).

Класичними методами лінійного програмування на сьогодні є:

− алгоритм Гоморі;

− симплекс-метод;

− метод потенціалів і т.д.

При цьому, *загальним* завданням лінійного програмування є розрахунок мінімального значення цільової функції лінійного типу, яку може бути подано у вигляді:

$$F(x) = \sum_{j=1}^{n} c_j x_j = c_1 x_1 + c_2 x_2 + ... + c_n x_n, \qquad (1.2)$$

де x – аргумент функції;

n – кількість змінних;

$c_1 ... c_n$ - константи.

Разом з тим, у задачах лінійного програмування можуть бути присутніми обмеження.

У цьому випадку, отримуємо *базову* задачу лінійного програмування, яку, у свою чергу, може бути представлено у вигляді наступної системи нерівностей:

$$\begin{cases} F(x) = \sum_{j=1}^{n} a_{ij} x_j \geq b_i, \ \mathrm{i} = \overline{1,\mathrm{m}}; \\ x_j \geq 0, \ \mathrm{j} = \overline{1,\mathrm{n}}. \end{cases} . \qquad (1.3)$$

Попри все зазначене, лінійне програмування матиме канонічний вигляд в умовах, якщо в описі його базового завдання (вираз 1.3) перший складник буде замінено на систему рівнянь, що має обмеження у вигляді рівності, як показано далі:

$$F(x) = \sum_{j=1}^{n} a_j x_j = b_1, \ \mathrm{i} = \overline{1, \mathrm{m}} \quad . \qquad (1.4)$$

Також маємо зазначити, що базову задачу лінійного програмування може бути зведено до канонічної за рахунок введення у модель додаткових знінних.

У цілому, лінійне програмування розглядає математичну модель процесу, у рамках якої оптимізаційне завдання, та вимоги до нього, моделюються на базі лінійних взаємозв'язків.

При цьому, їх оцінка виконується з використанням лінійних цільових функцій.

З іншого боку, у рамках нелінійного програмування оптимізаційне завдання, існуючі обмеження а також цільові функції можуть містити у собі нелінійні взаємозв'язки.

Такі оптимізаційні завдання можуть виявитися надто складними в контексті детермінованої оптимізації.

Формальний опис задачі нелінійного програмування може бути отримано уточненням системи виразів (1.1) до наступного вигляду:

$$\begin{cases} F(x_1; x_2; ...x_n) \xrightarrow[opt]{} \max\ (\min); \\ g_i(x_1; x_2; ...x_n) \le b_i, i = \overline{1, m}, \end{cases} \quad (1.5)$$

де F – нелінійна цільова функція від n змінних;

g_i – нелінійні функції обмеження;

b_i – задані числа.

Як LP, і NLP є ефективними для рішення широкого класу завдань оптимізації, які передбачають наявність єдиного оптимального рішення, що є при цьому також глобально оптимальним.

Водночас, існують також методи рішення завдань оптимізації, де може існувати деяка множина оптимальних результатів.

Окрім лінійного та нелінійного програмування, сьогодні існують також інші детерміновані моделі оптимізації, як:

- цілочисельне програмування (IP);
- опукле нелінійне програмування (CNLP);
- змішане цілочисельне програмування (MINLP).

1.3 Стохастичні методи оптимізації

На відміну від детермінованих, стохастичні алгоритми використовують елементи випадковості у ході вибору напрямку та/або розміру кроку під час оптимізації [4, 8].

Тут важливо зазначити про те, що стохастичні методи може бути застосовано для рішення детермінованих завдань.

У цьому разі виконується цілеспрямоване введення випадковості у алгоритм для того, щоб сприяти досягненню мети.

При цьому, стохастична оптимізація може розглядатися як особливий клас алгоритмів оптимізації, які у тій чи іншій мірі використовують випадковості у ході знаходження оптимуму функції.

Одним з ключових завдань алгоритмів стохастичної групи є рішення задач оптимізації в умовах, коли функція, яка описує той чи інший процес, має складну форму, при цьому за різних умов перебіг процесу може певною мірою відхилятися від базового сценарію у той чи інший бік.

Зазначені умови, наприклад, є властивими для фізичного моделювання, зокрема, тоді, коли експериментальні дані зазнають впливу викривлень різних типів та інтенсивності.

У такому разі говориться про те, що оптимізація має виконуватися відносно деякого попередньо обраного критерію якості.

Такий критерій, у свою чергу, може не мати математичного опису, натомість може існувати можливість отримати значення функції за результатами виконання експериментальних досліджень.

При цьому, застосування алгоритмів стохастичної оптімизації виконується ітераційно.

Говорячи про стохастичні алгоритми однопараметричної багатоекстремальної оптимізації, зазначимо, що такі алгоритми не потребують обов’язкової неперервності функції (їх, у свою чергу, може бути використано для рішення задач оптимізації гладких функций).

Це є суттєвим для розв’язання практичних завдань, оскільки, частіше за все, властивості цільової функції, яку побудовано математичною моделлю, є невідомими.

Окрім зазначеного, нерідко необхідність отримання точки екстремуму з дуже високим рівнем точності є відсутньою.

Це зумовлюється, у свою чергу, тією обставиною, що для математичної моделі, яка при цьому використовується, апріорі не гарантується абсолютна відповідність реальному процесу.

Слід відзначити, що особливе місце серед алгоритмів стохастичної оптимізації займають такі, основу яких складають ті чи інші процеси, що мають місце у навколишньому світі, як з живої так і неживою природи, наприклад:

− перебіг фізичних процесів;

− процеси, що відбуваються у селекції живих організмів (відбір, розвиток, мутації, зміни поколінь та оцінка характеристик нових популяцій);

− поведінка колективних тварин тощо.

Так чи інакше, ключове оптимізаційне завдання стохастичних методів, формальний опис якого може бути подано виразом (1.1), зводиться до пошук такого поєднання параметрів (незалежних змінних), за якого забезпечується максимізація (мінімізація) можливої сукупності ймовірнісних, якісних та кількісних характеристик розв'язуваного завдання [2].

Також окремої уваги заслуговують стохастичні алгоритми, які базуються на імітації роботи т.з. ройового інтелекту (Particle Swarm Optimization) [9, 10].

Типовими представниками стохастичних методів оптимізації є:

− метод Монте-Карло;

− алгоритм мурашиної колонії;

− метод імітації відпалу;

− алгоритм косяка риб;

− алгоритм рою бджіл;

− диференційна еволюція;

− група еволюційних алгоритмів;

− метод випадкових блукань тощо.

Разом з тим, у відповідності до технічного завдання на кваліфікаційну роботу, необхідно виконати дослідження саме методів стохастичної групи. Виходячи з цього далі здійснимо більш грунтовний розгляд стохастичних методів оптимізаії.

Беручи до уваги усе, що було зазначено вище, класифікацію детермінованих методів оптимізації можемо подати у вигляді схеми, поданої на рисунку 1.1.

При цьому для того, щоб виконати всебічне дослідження стохастичних алгоритмів та їхніх можливостей, для розгляду доцільно обирати такі алгоритми, механізми дії яких є максимально відмінними один від одного. Виходячи з зазначених міркувань, далі пропонується дослідити такі алгоритми стохастичної оптимізації, як:

− алгоритм імітації відпалу;

− алгоритм косяка риб;

− бджолині алгоритми.

Далі розглянемо стислий опис сутності кожного з алгоритмів, перелічених щойно.

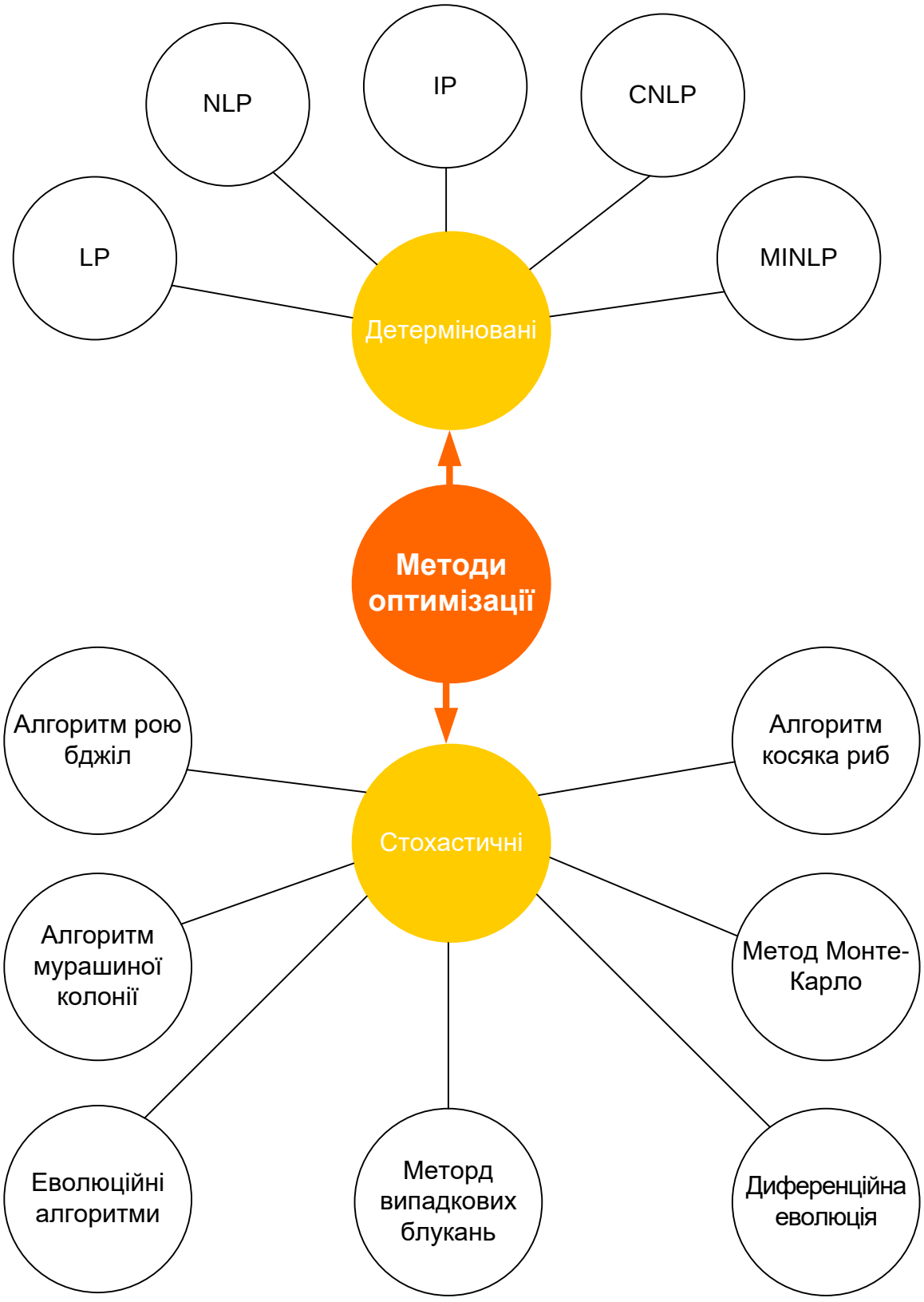

Рисунок 1.1 – Класифікація детермінованих та стохастичних методів оптимізації

1.4 Огляд алгоритмів оптимізації, які підлягають дослідженню

1.4.1 Алгоритм косяка риб

Алгоритм пошуку на базі косяка риб (в оригіналі - Fish School Search, або FSS) було було представлено до розгляду у 2008 році його авторами - Б. Філо та Л. Нето.

Для даного алгоритму ключовими його складовими є [11]:

− рух агенту;

− поведінка агента;

− специфіка поведінки косяку риб.

При цьому, особливості поведінки косяку риб є залежними від пари операторів, а саме - оператору годування, а також оператору переміщення (плавання).

У рамках алгоритму кожен з агентів (окремої риби у косяку) характеризується власною пам'яттю, яка, у свою чергу, використовується для зберігання даних відносно рівня його ваги, а також найкращого розміщення у просторі. Тут також передбачається, що вага окремого агенту буде обмежено деяким максимальним значенням.

Водночас, стартове значення ваги агенту розглядається у вигляді 50% від максимально можливого її значення.

1.4.2 Алгоритм імітації відпалу

Алгоритм імітації відпалу являє собою загальний алгоритмічний інструмент пошуку рішення задач глобальної оптимізації. Може розглядатися як частковий випадок методу Монте-Карло [8, 9, 10].

Базис алгоритму складає імітаціє фізичного процесу кристалізації речовини, що може бути проілюстровано конкретним процесом - відпалом металів. У ході цього, після нагрівання металу до деякої температури, виконується його пасивне охолодження, що супроводжується кристалізацією речовини. У рамках методу робиться припущення про те, що атоми речовини можуть бути практично організовані у кристалічну решітку, проте залишається певна можливість здійснення переходів окремими атомами від одного енергетичного стану до іншого.

Хоча використання означеного алгоритму не надає гарантії розрахунку екстремуму функції, водночас, за умови його коректного налаштування

досягається точки, яку можна вважати квазі-оптимальною навіть за умов коли оптимізована функція є досить складною, а використання детермінованих методів для оптимізації процесу, який вона наближено описує, дає незадовільний результат.

На практиці частіше за все алгоритм відпалу, окрім, безпосередньо, оптимізації процесів, заданих складними функціями, має застосування для:

− реалізаії процесів навчання нейромереж;

− пошуку рішень складних комбінаторних завдань, типовим прикладом яких може бути задача про розміщення ферзей;

− пошуку рішень задачі комівояжера.

1.4.3 Бджолині алгоритми

Як вже зазначалося, алгоритми, що імітують штучній бджолиний рій (алгоритм бджолиного рою, а також бджолиний алгоритм та їхні варіації) є типовими представниками алгоритмів роєвого інтелекту, таких, як алгоритм мурашиної колонії, та алгоритм косяку риб [8-16].

Основу даного підходу складає імітація поведінки рою бджіл.

При цьому, необхідною та першочерговою умовою для можливості його застосування для вирішення того чи іншого завдання є наявність деякої топологічної відстані його аналогу на ймовірній області рішень.

Зрозуміло, що агентами у даному разі є бджоли. При цьому, метод передбачає наявність двох типів бджіл, а саме:

− бджіл-розвідників;

− бджіл-робітників.

У той час, як перший тип агентів виконує завдання дослідження території, яка, у свою чергу, оточує вулик, на предмет наявності нектару, другий тип агентів реалізує функції збору меду, а також оновлення інформації про дану та прилеглі галузі [16].

Відповідно до алгоритму після того, як бджоли-розвідники повертаються до вулику, вони надають інформацію відносно розвіданої кількість нектару, напрямок його розташування, а також відстань до нього.

Далі, бджоли робітники переміщуються до найбільш прийнятних областей. У даному разі від кількості нектару, яку було виявлено розвідниками на передуючому кроці, залежить кількість бджіл-робітників, які переміщуються до тієї чи іншої області.

Також, окрім, безпосередньо, збору меду, бджоли-робітники виконують завдання оновлення даних щодо поточної області та областей, розміщених безпосередньо поруч.

Таким чином, штучна бджолина колонія використовує алгоритм, що є подібним до процесу пошуку та збору нектару медоносними бджолами.

У цьому випадку замість поля з квітами розглядається можлива область рішень. У свою чергу, замість нектару розглядаються критерії задач оптимізації та цільова функція.

У процесі пошуку рішень, на кожній ітерації алгоритму, виконується вибір деякої кількості областей, яким відповідають кращі значення цільової функції. Такі області звуться "кращими", при цьому, далі з областей, які залишилися (які попередньо не було включено до «кращих»), обирається ще дяка кількість кращих, що у даному разі мають назву "перспективних" [14].

Налаштовуючи алгоритм, можна встановити деяку конкретну найменшу відстань між двома сусідніми областями. Тоді, коли виникатимуть накладення, буде здійснюватися відсічення областей, яким відповідає гірше значення цільової функції. Далі замість такої області буде обиратися інша. Алгоритм запам'ятовує такі області і далі, у ході наступного кроку ітерації, до них відправляється деяку кількість агентів.

2 АЛГОРИТМ ІМІТАЦІЇ ВІДПАЛУ

2.1 Опис алгоритму

2.1.1 Фізичне підгрунтя алгоритму

Алгоритм імітації відпалу являє собою типовий приклад стохастичного алгоритму [8, 12].

Основу алгоритму складає імітування фізичного процесу, перебіг якого має місце у ході кристалізації речовини, що також справедливо для операції відпалу металів. Так, у ході процедури відпалу метал спочатку підлягає нагріванню до високої температури, після чого – поступово пасивно охолоджується.

При цьому, робиться припущення про те, що атоми речовини на поточний момент вже майже сформовано у кристалічну решітку, проте залишається можливість того, що окремі атоми зможуть переходити з однієї чарунки до іншої.

Водночас, чим вищою є температура металу (яка у ході процесу зазнає поступового зниження), тим вищою є активність атомів. З іншого боку, зі зменшенням температури поступово скорочується можливість переходів атомів у стани, які характеризуються більшою енергією.

У свою чергу, найнижчий рівень енергії атомів відповідає стійкій кристалічній решітці. Тобто, далі атом або залишається на місці, або виконує перехід до стану з нижчою енергією [4, 8].

Розглянуту модель процедури відпалу може бути використано для оптимізації складних процесів, для яких застосування багатьох інших методів оптимізації (регресії, лінійне програмування і т.д.) з тих чи інших причин є неефективним. Завдання оптимізації таких процесів може бути інтерпретовано як пошук однієї, або певної кількості точок, у яких цільова функція отримує мінімальне (максимальне) значення, тобто, шукається екстремум певної цільової функції $f(x)$, $x \in X$.

У контексті моделі функцію $f(x)$ тут слід сприймати як т.з. енергію системи, x - як стан системи, множину X - як сукупність усіх можливих станів.

2.2 Формалізація алгоритму

Спрощено алгоритм, як перелік формальних технологічних кроків, може бути представлено наступним переліком дій [4]:

1. Вибір точки з простору можливих величин за випадковим законом, для якої далі виконується розрахунок значення функції.

2. Ітеративний вибір точки, якій може відповідати мінімум з подальшим порівнянням значення функції у цій точці з величиною функції на попередній точці.

3. Перехід до нової точки, якщо значення функцій є меншим, ніж передуюче.

4. Якщо значення функції на кроці 3 більше передуючого, виконується перехід до нової точки. При цьому, ймовірність такого переходу визначається величиною одного з параметрів, а саме – температури: чим меншим є значення температури, тим нижчою є ймовірність переходу. Даний підхід дозволяє виходити з локальних мінімумів.

Оскільки алгоритм імітації відпалу належить до стохастичних, у процесі пошуку мінімуму функції на його базі передбачається можливість встановлення випадкових величин для ряду параметрів, а саме:

− стартового стану системи $x_0 \in \mathrm{X}$;

− оператору $A(x;i) : \mathrm{X} \times \mathbb{N} \to \mathrm{X}$, який після виконання $i-го$ кроку випадковим чином здійснює генерацію нового стану системи, ураховуючи поточний стан x; до даного оператору висовуються наступні вимоги – оператор має забезпечувати швидкий пошук, хоча, одночасно, має сприяти відносно вільному обходженню простору X за випадковим законом;

− невід'ємної послідовності $T_i > 0$ яка має тенденцію зменшення до 0, вона являє собою своєрідний аналог температури кристалу, яка поступово зменшується.

Сам алгоритм зводиться до ініціалізації та підтримки процесу переміщення простором X станів за випадковим законом [4, 8].

При цьому, пошук рішення здійснюється шляхом послідовного обчисленням значень точок x_0, x_1, x_2 ... , які, у свою чергу, знаходяться у межах простору X. У ході цього, починаючи з точки x_1, кожна точка простору X може потенційно розглядатися як така, що сприяє пошуку рішення більш ефективно, ніж інші.

Протягом кожного кроку алгоритму за механізмом, який буде розглянуто окремо, виконується розрахунок нової точку, а також зменшується величина параметру, який інтерпретується, як температура.

У свою чергу, отримання послідовності точок (станів системи) виконується наступним чином. Відносно кожної точки x_i застосовується оператор A, наслідком чого є отримання т.з. точки-кандидату $x_i^* = A(x,i)$. Для такої точки дані здійснюється розрахунок відповідної зміни величини енергії системи за виразом:

$$\Delta f_i = f(x_i^*) - f(x_i). \tag{2.1}$$

Далі у випадку, коли рівень енергії знижується, тобто, виконується умова:

$$\Delta f_i \leq 0, \tag{2.2}$$

система переходить до нового стану, що еквівалентно виразу:

$$x_{i+1} = x_i^*. \tag{2.3}$$

У подальшому, коли виконується умова підвищення енергії, тобто:

$$\Delta f_i > 0, \tag{2.4}$$

деяка ймовірність переходу системи до нового стану може зберігатися.

При цьому, конкретне значення даної ймовірності визначатиметься поточною величиною температури та величиною підвищення енергії та поточної температури.

Дану ймовірності на базі значень перелічених величин може бути обчислено згідно з законом розподілу Гіббса [4], як показує наступний вираз:

$$P(x_i^* \rightarrow x_{i+1} \mid x_i) = e^{-\frac{\Delta f_i}{T_i}}. \quad (2.5)$$

У випадку, якщо переходу системи до іншого стану не відбулося, стан системи залишається на рівні $x_{i+1} = x_i$ - тобто, на передуючому рівні.

Зупинку алгоритму виконується тоді, коли буде досягнуто точки, яка відповідає нульовій температурі.

У цілому, можна зазначити, що алгоритм імітації відпалу має риси, схожі з алгоритмом градієнтного спуску.

Разом з тим, з тієї причини, що проміжні точки обираються випадково, а також завдяки можливості виходити з локальних мінімумів, забезпечується набагато нижча ймовірність сходження до локальних точок екстремуму порівняно з методом градієнтного спуску.

2.3 Алгоритмічне та математичне підгрунтя алгоритму

Згідно з чинним технічним завданням, вимоги до візуалізації роботи алгоритму, а також вимоги щодо необхідності забезпечення функціонування існуючих механізмів, що забезпечують роботу алгоритму, відсутні. Звідси виходить, що у нашому випадку розглядається уніфікована ситуація реалізації алгоритму.

Водночас, якщо брати до уваги головні параметри алгоритму, розглянуті раніше, тоді слід зазначити, що ключові параметри, що впливають на результативність алгоритму, це:

– функція, пошук екстремуму якої виконується (у нашому випадку - мімімум) – E;

– значення аргументу, який розраховується (за якого значення функції E буде мінімальним) – s;

– величина, починаючи з якої виконується пошук мінімуму (зазвичай це – випадкове число) – s_0;

– загальний обсяг ітерацій, у межах якого виконуються операції алгоритму – $k\_\max$;

– величина температури – T;

– функція, яка забезпечує зміну температури; зазначимо, що величина температури має знижуватися поступово – *temperature*;

– функція, яка здійснює вибір наступної величини s – *neighbour*;

– функція, яка визначає величину ймовірності переходу з передуючого стану до нового (величина даної ймовірності має скорочуватися зі зменшенням величини температури) – P.

Програмний код, який реалізує алгоритм відпалу у базисному варіанті, наводиться далі [4]:

```
s = s0
k = 0
while k <= k_max:
    T = temperature(k / k_max)
    s_new = neighbour(s)
    e_old = E(s)
    e_new = E(s_new)
    if P(E_old, e_new, T) >= random(0, 1):
        s = s_new
    k += 1

return s
```

Наведений вище програмний код є базовим для алгоритму.

Для регулювання роботою алгоритму використовується маніпулювання величинами кількості ітерацій $k\_\max$, а також функціями *neighbour* та *temperature*.

Разом з тим, у залежності від того, які саме підходи до реалізації алгоритму імітації відпалу буде застосовано, може бути внесено різні варіації для функції температури та функції переходу P.

Так, за замовчуванням для функції переходу P використовується один з наступних виразів:

$$P = \frac{1}{math.\exp(\frac{e\_new - e\_old}{T})}, \tag{2.6}$$

де $math.\exp$ – оператор, що повертає значення виразу ex, у якому x є аргументом методу а e – константа, число Ейлера, основа натурального логарифму;

або:

$$P = math.\exp(-\frac{e\_new - e\_old}{T}). \tag{2.7}$$

У свою чергу, розрахунок зміни температури найчастіше за все виконується з використанням різних підходів. Ряд з підходів, які використовуються частіше за все, розглянемо окремо.

2.3.1 Больцманівський відпал

Одним з них є Больцманівский відпал, у рамках якого функція зміни температури виззначається наступним виразом [4, 8]:

$$T(k) = \frac{T_0}{\ell n(1+k)}, \tag{2.8}$$

де T_0 – початкове значення температури.

При цьому, свого часу для Больцманівської схеми було доведено, что за умови великого початкового значення температури T, а також при тому. що кількість кроків ітерації буде великою, глобальний мінімум функції може бути гарантовано знайдено.

2.3.2 Відпал Коші (швидкий відпал)

Аналіз виразу (2.8) показує, що для випадку Больцманівського відпалу характерним є дуже повільне падіння рівня температури, тобто, у даному випадку не забезпечується швидкого сходження алгоритму.

Відтак, для того, щоб у тих чи інших умовах, за потреби, забезпечити швидкий пошук екстремуму функції, замість даного підходу використовується т.з. відпал Коші, сутність якого пояснюється наступним виразом:

$$T(k) = \frac{T_0}{k}. \tag{2.9}$$

Зараз метод імітації відпалу у веріаціях, розглянутих раніше, існує у вигляді Python-бібліотеки SciPy [8]. Для цього слугує метод `scipy.optimize.anneal`.

Проте, у версіях SciPy, починаючи з 0.14, даний метод було визнано таким, що на поточний час застарів. Тому натомість розробником було рекомендовано для застосування замість `scipy.optimize.anneal` метод `scipy.optimize.basinhopping`.

3 МЕТОДИ СТОХАСТИЧНОЇ ОПТИМІЗАЦІЇ НА ОСНОВІ БДЖОЛИНИХ АЛГОРИТМІВ

3.1 Загальні відомості відносно бджолиних алгоритмів

Виконаємо дослідження основних принципів роботи групи стохастичних алгоритмів, в основі яких лежить модель, яка відповідає поведінці бджіл у живій природі під час пошуку нектару. Це стосується таких алгоритмів, як [9, 10, 14-16]:

– бджолиний алгоритм, або Bee Algorithm (BA);

– модернізований бджолиний алгоритм;

– алгоритм колонії бджіл, або Artificial Bee Colony Algorithm (ABC) , який, у свою чергу, імітує поведінку бджіл у пошуках нектару.

Як вже говорилося, алгоритм бджолиного рою, який імітує поведінку бджіл у живій природі, належить до стохастичних оптимізаційних алгоритмів.

Слід зазначити, що у рамках реалізації даного алгоритму не виконується формування моделі життя бджіл, яка є тією чи іншою мірою наближеною до реальної природної екосистему.

Натомість мова йде про імітацію поведінки колонії, як оптимізаційного засобу, у якій система буде мати відмінності від природної.

Розглянемо загальний принцип функціонування алгоритму бджолиного рою, спочатку у спрощеній формі.

3.2 Принципи роботи бджолиного алгоритму

3.2.1 Класичний бджолиний алгоритм

Першим кроком роботи алгоритму є ініціалізація деякої множини (популяції) бджіл [14].

Далі виконується розміщення бджіл на випадкових локаціях.

На наступному кроці виконується операція сортування бджіл з огляду на величини цільових функцій у межах тих ділянок, де їх було розміщено.

Після цього ініціюється технологічний етап визначення т.з. «елітних ділянок», до яких далі надсилаються n бджіл-розвідників.

У свою чергу, такі бджоли формують нові ділянки на околицях елітних ділянок.

Далі до околиць інших $(m-e)$ ділянок, виходячи з величини цільових функцій, які їм відповідають, надсилаються робочі бджоли кількістю ℓ, яка визначається, як:

$$\ell = N - n. \tag{3.1}$$

У цілому, беручи до уваги розглянутий принцип функціонування алгоритму, слід зазначити, що специфіка робота алгоритму буде визначатися значеннями перелічених далі параметрів, зокрема:

– загальний обсяг N бджіл-розвідників;

– сумарна кількість m ділянок;

– обсяг e ділянок, які вважаються елітними;

– кількість n бджіл-розвідників, які знаходяться у межах елітних ділянок;

– кількість ℓ бджіл, які присутні на інших ділянках, яких є $(m-e)$;

– стартовий обсяг S ділянок (розмір ділянок, беручи також до уваги їхні околиці;

– найбільша можлива кількість I ітерацій.

Сама ідея бджолиного алгоритму полягає у тому, що усі бджоли, протягом кожного кроку, обиратися для дослідження як ділянок елітного типу, так і ділянок, що локалізуються на околицях елітних ділянок [9, 14].

Такий підхід дозволяє, у першу чергу, виконати урізноманітнення популяції рішень протягом наступних кроків ітерацій.

Окрім цього, також забезпечується збільшення ймовірності виявлення рішень, які є наближеними до оптимальних.

Далі наведемо алгоритмічний опис бджолиного алгоритму, схему якого зображено на рисунку 3.1.

Отже, на початку роботи алгоритму, виконується розташування за випадковим законом множини бджіл загальною кількістю N на поверхнях ділянок.

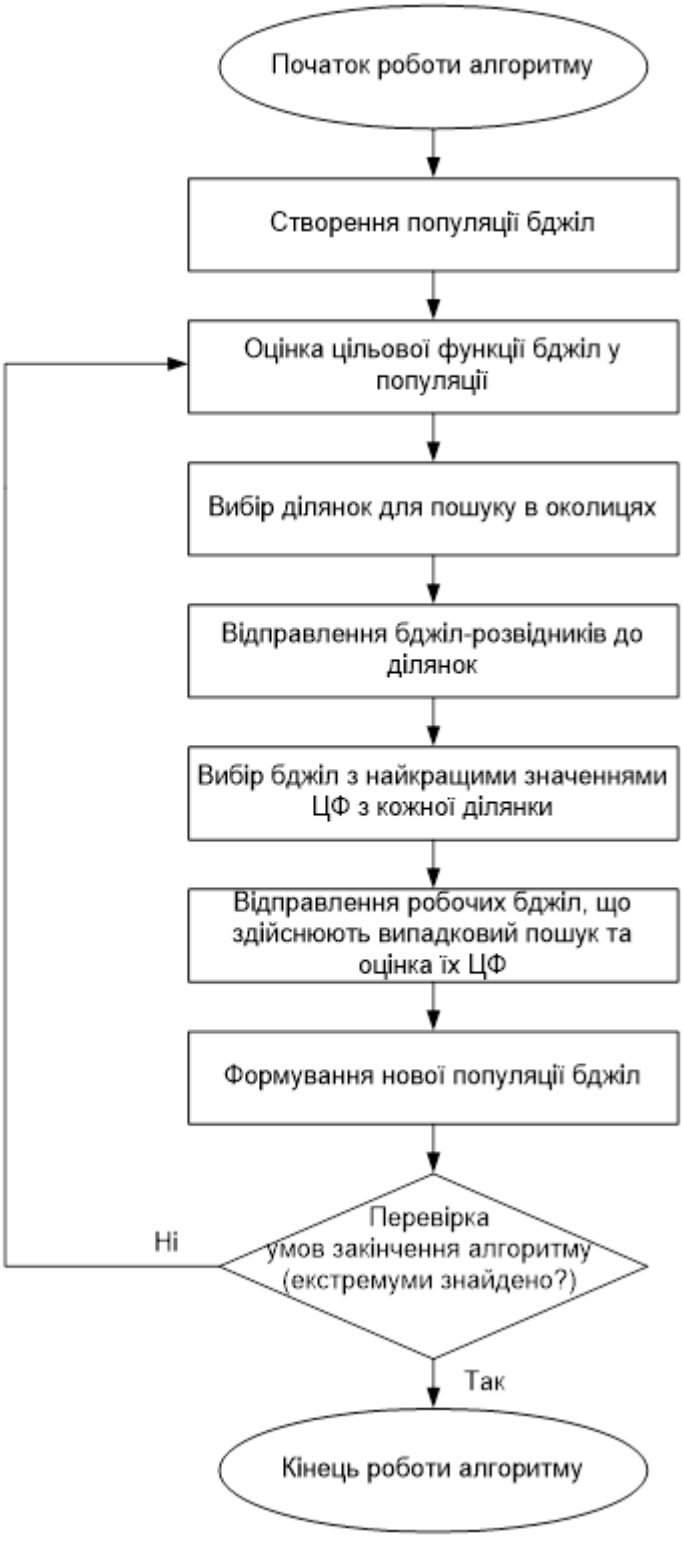

Рисунок 3.1 – Загальний перелік етапів виконання бджолиного алгоритму

Після цього виконується розрахунок цільових функцій ділянок. У подальшому ті з ділянок, яким відповідають найбільші величини цільової функції (тобто, такі ділянки є елітними), обираються для реалізації подальшого пошуку рішень у межах їхніх околиць [14].

Отже, на початку роботи алгоритму, виконується розташування за випадковим законом множини бджіл загальною кількістю N на поверхнях ділянок.

Після цього виконується розрахунок цільових функцій ділянок. У подальшому ті з ділянок, яким відповідають найбільші величини цільової функції (тобто, такі ділянки є елітними), обираються для реалізації подальшого пошуку рішень у межах їхніх околиць [14].

У даному випадку, у межах попередньо обраних ділянок виконуються більш детальні дослідження, тобто. До них надсилається більша кількість бджіл, ніж до кожної з $(m-e)$ ділянок.

Далі, протягом наступного кроку, виконується оцінка величин цільових функцій, а далі виконується вибір найкращих бджіл виходячи з величин цільових функцій ділянок, які ними досліджуються. У подальшому на базі цих бджіл реалізується формування нової популяції ймовірних рішень, що буде брати участь у ході наступної ітерації алгоритму.

На наступних кроках робочі бджоли будуть виконувати випадковий пошук в околиці елітних ділянок з метою знаходження нових рішень. Заначений перелік дій буде тривати доти, поки не буде досягнено критерію зупинки алгоритму (знайдено екстремум функції).

Звідси виходить, що ключовою операцією у межах бджолиного алгоритму є аналіз перспективних областей та їх околиць [9].

Далі, по закінченню алгоритму, ймовірна популяція рішень буде містити у собі дві частини, а саме: множину бджіл, яким відповідають найкращі значенням цільових функцій елітних ділянок, та множину робочих бджіл, яким відповідають випадкові величини цільових функцій.

3.2.2 Покращений бджолиний алгоритм

Виконаємо далі розгляд бджолиного алгоритму покращеного типу, або Improved Bee Algorithm.

Сенс даного алгоритму зводиться до того, що протягом кожної з виконуваних ітерації здійснюється зменшення початкового розмір ділянок S. Таке зменшення, у свою чергу, виконується за рахунок розподілу віихідного розміру на адаптивне значення R. У цьому разі R являє собою параметр редукції [9, 14, 16].

При цьому, величина означеного параметру може варіюватися з урахуванням рівня якості рішення, одержаного впродовж кожної ітерації.

Звідси виходить, що чим гіршим буде виходити отримуване рішення, тим параметр R буде ближчим до 1.

3.2.3 Алгоритм бджолиного рою

Уперше даний алгоритм було презентовано у 2005 році. Його автором є турецький вчений Д. Карабога, який використав алгоритм бджолиного рою дла того, щоб розв'язати завдання параметричної оптимізації [15].

Сутність алгоритму бджолиного рою безпосерендім чином полягає у побудові моделі поведінки колонії бджіл у пошуках нектару.

При цьому, у реальних умовах живої природи основу функціонування вулику бджіл складає чіткий розподіл обов'язків між окремими його індивідами.

Разом з цим, усіх бджіл у межах вулику може бути розділено на такі групи, як:

1) робочі бджоли;

2) бджоли-дослідники;

3) бджоли-розвідники.

У даному разі до функцій робочих бджіл належать:

– пошук ймовірних джерел нектару;

– забезпечення інформацією відносно якості досліджених ділянок для бджіл-дослідників.

У свою чергу, під час активності робочих бджіл, бджоли-дослідники локалізуються у межах вулику, та одержують інформацію відносно об'єктів дослідження виключно лише від робочих бджіл.

Що стосується бджіл-розвідників, зазначимо, що до їхніх обов’язків належить реалізація операцій випадкового пошуку нових, ще невідомих, джерел нектару [9, 15]. Принцип роботи алгоритму ABC може бути проілюстровано схемою, зображеною на рисунку 3.2.

Таким чином, під час виконання алгоритму, спочатку виконується встановлення простору пошуку $x_i (i = 1...S)$ рішень відновідно до випадкового закону, де - S – ймовірна кількість джерел нектару.

Далі, беручи за основу величину S, виконується розрахунок загальної кількості ℓ' робочих бджіл за виразом:

$$\ell' = \frac{S}{2}. \tag{3.2}$$

Після цього, беручи до уваги значенням ℓ', для кожної робочої бджоли, визначається ділянка v_{ij} для пошуку за виразом:

$$v_{ij} = x_{ij} + \phi_{ij} \times (x_{ij} - x_{kj}), \tag{3.3}$$

де ϕ_{ij} – випадкова величина, що належить діапазону [-1;1];

j – параметрична величина, $j = \overline{1, D}$;

D – розмірність задачі;

k – індекс рішення, яке обрано за випадковим законом з колонії, при цьому [15]:

$$k = \text{int}(rand \times S) + 1. \tag{3.4}$$

Після того, як ділянку v_{ij} визначено, виконується порівняння отриманого у даному разі рішення з відповідною величиною x_i. За результатом даної процедури, робочу бджолу надсилається до тієї з ділянок, яка у результаті порівняння визнається кращою.

Далі виконується вибір джерела нектару бджолами-дослідниками. При цьому, поточна ймовірність p_i такого вибору розраховується наступним чином:

$$p_i = \frac{f_i}{\sum_{j=1}^{S} f_j}, \tag{3.5}$$

де f_i – цільова функція x_i.

Окрім вибору джерела нектару на базі значення ймовірності p_i, виконується визначення нової ділянки у локальній області дослідження,

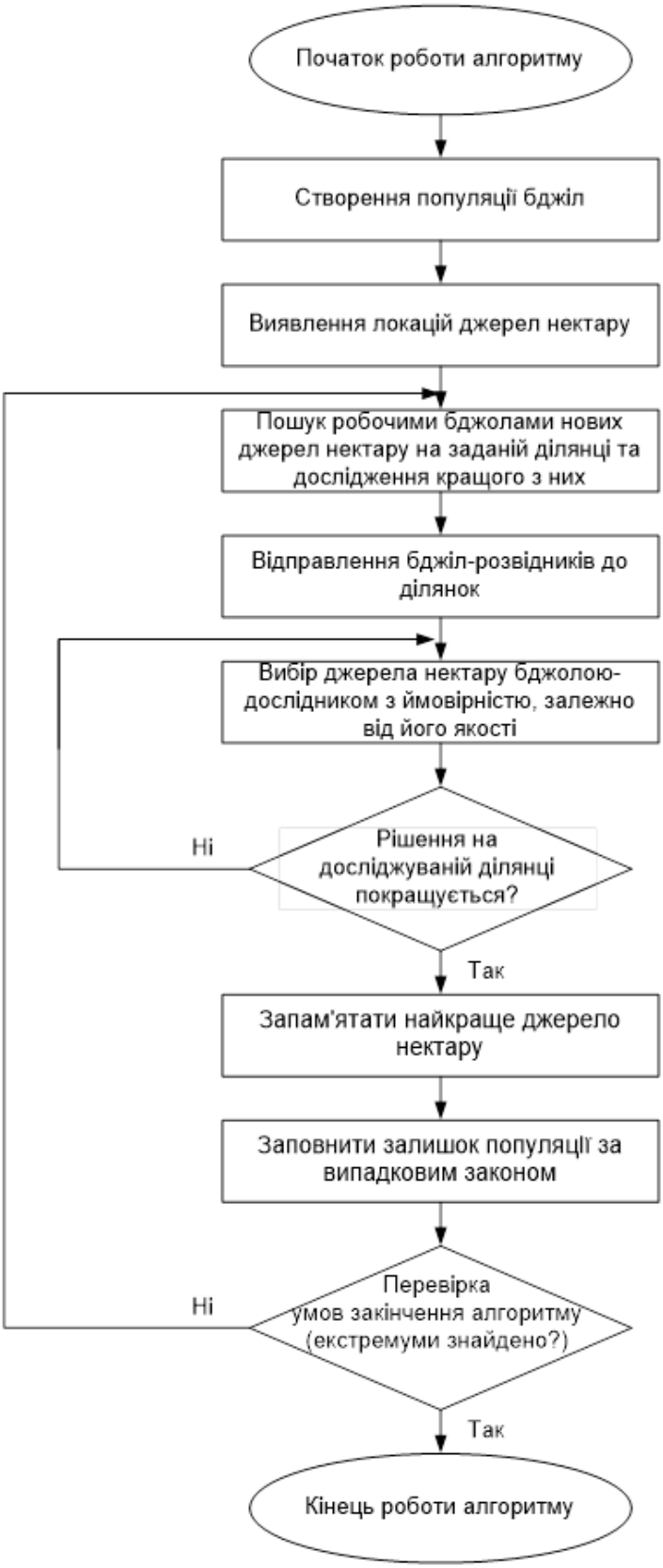

Рисунок 3.2 – Схема, що ілюструє принцип роботи алгоритму ABC

відповідно до виразу (3.2). Нарешті, за умови, що після виконання попередньо встановленої кількості ітерацій, рішення, одержане у межах даної ділянки не покращується, таку ділянку далі виключається з подальшого розгляду.

При цьому, бджоли-дослідники такої ділянки стають бджолами-розвідниками, а відбір рішень далі виконується за випадковим законом на базі наступної формули [9, 15]:

$$x_{ij} = x_j^{\min} + (x_j^{\max} - x_j^{\min}) \times rand\,. \qquad (3.6)$$

3.3 Дослідження ефективності бджолиних алгоритмів

Виконаємо оцінку ефективності розглянутих вище бджолиних алгоритмів – простого бджолиного алгоритму (BA), а також покращеного бджолиного алгоритму (IBA) і алгоритму колонії бджіл (ABC).

Окрім цього, виконаємо порівняння означених вище різновидів бджолиних алгоритмів.

Для цього розглянемо приклади результатів пошуку екстремумів для функцій Растригіна та Розенброка.

3.3.1 Функції, на базі яких виконується дослідження

Як функція Растригіна, так і функція Розенброка, є спеціалізованими функціями, що використовуються для перевірки ефективності методів оптимізації.

Так, функцію Растригіна може бути описано виразом [14]:

$$f(x) = An + \sum_{i=1}^{n} [x_i^2 - A\cos(2\pi x_i)], \qquad (3.7)$$

де A – константа, при цьому, $A = 10$;

$x_i \in [-5{,}12;\ 5{,}12]$.

Тобто, у цьому випадку маємо невипуклу функцію двох змінних.

У свою чергу, даній функції властива велика кількість локальних екстремумів (рис. 3.3) [14].

Зрозуміло, що з цієї причини, а також з причини досить великого простору пошуку, знаходження глобального екстремуму є досить складним завданням.

З іншого боку, це зумовлює цінність функції Растригіна для тестування оптимізаційних алгоритмів. Також звертає на себе увагу те, що аналітично екстремум функції може бути легко визначено – $f(x)=0$ при $x=0$.

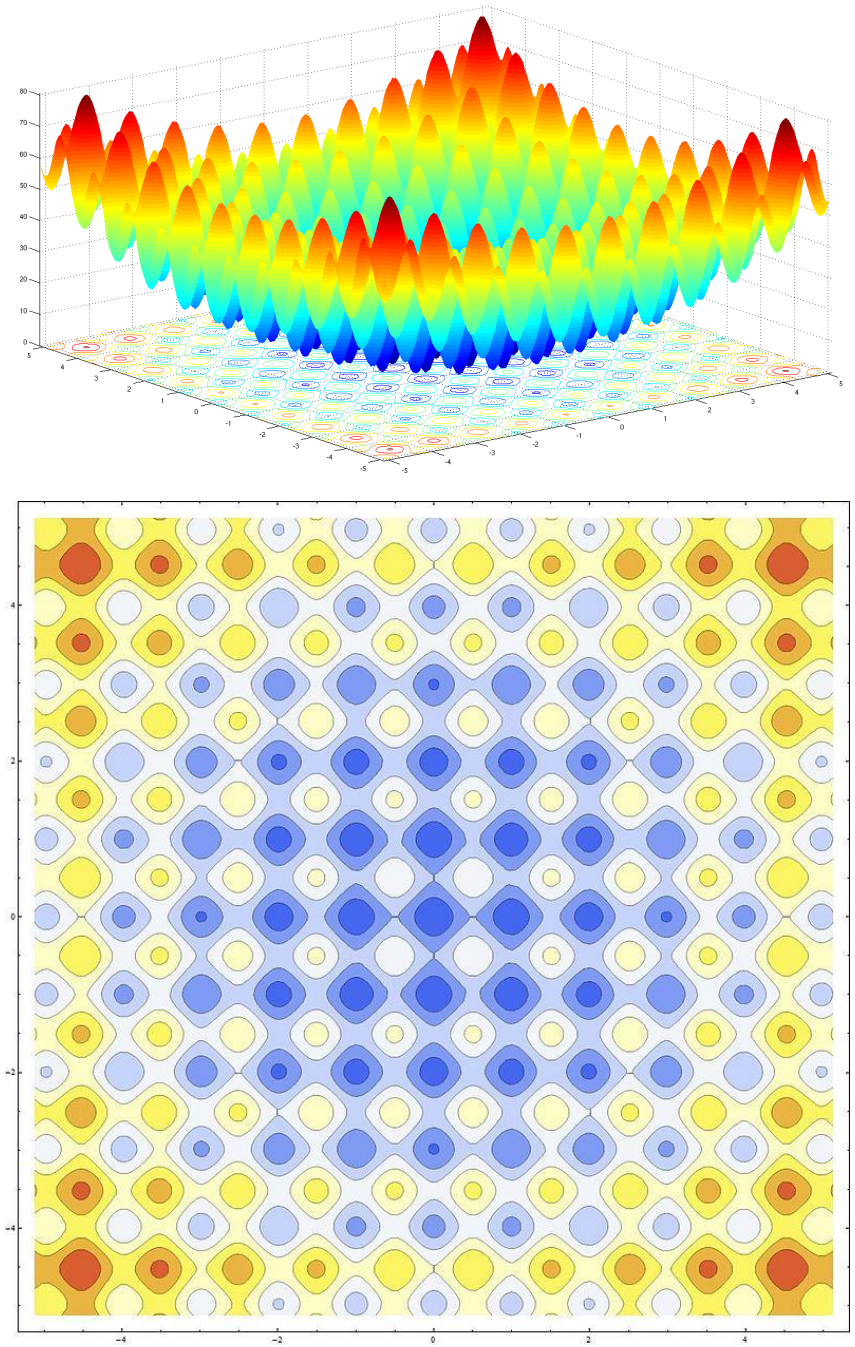

Рисунок 3.3 – Графік функції Растригіна а) та поверхня графіку з ізолініями б)

Функція Розенброка також є функцією двох змінних, та належить до функцій невипуклого типу.

Цінність даної функції для тестування методів оптимізації полягає у тому, що знаходження її точки екстремуму вважається нетривіальним завданням. Функцію Розенброка у канонічному вигляді може бути задано виразом [14]:

$$f(x,y) = (1-x)^2 + 100(y-x^2)^2. \qquad (3.8)$$

З аналізу виразу (3.7) неважко зрозуміти, що глобальний екстремум даної функції знаходиться у точці $(x,y) = (1,1)$, у цьому разі $f(x,y) = (0)$. Графік даної функції має вигляд, як показує рис. 3.4.

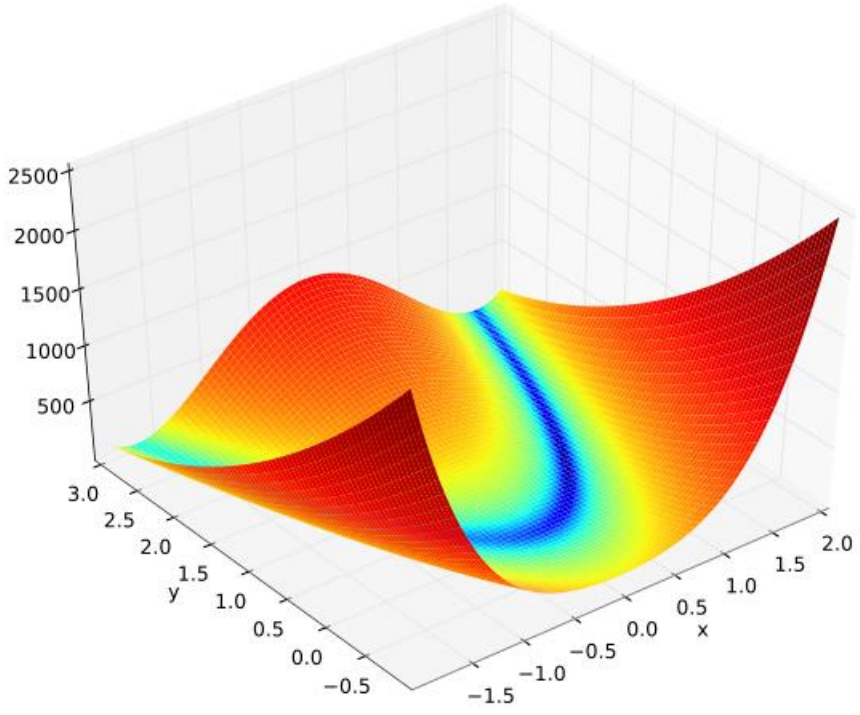

Рисунок 3.4 – Графік канонічної функції Розенброка

3.3.2 Параметри тестування алгоритмів

У нашому випадку було розглянуто такі параметри алгоритмів [15]:

– для алгоритму ВА значення $N = 50$, $e = 5$, $m = 15$, $n = 25$, $S = (0{,}5;\ 3)$, $\ell = 25$;

– для ІВА, відповідно, було узято $N = 60$, $e = 1$, $m = 3$, $n = 50$, $\ell = 10$, при цьому, початковий розмір ділянок $S = 6$, а $\mathrm{R} = 1{,}05$.

– для алгоритму ABC усі параметри відповідають попередньому алгоритмові, завиключенням кількості ділянок пошуку $S = 20$.

Разом з тим, для кожного досліджуваного алгоритму буде використано фіксоване значення загальної кількості ітерацій – $I = 100$.

Як можна бачити з рис. 3.5 та рис. 3.6, алгоритм колонії бджіл демонструє вищу продуктивність, порівняно з простим бджолиним алгоритмом та покращеним бджолиним алгоритмами.

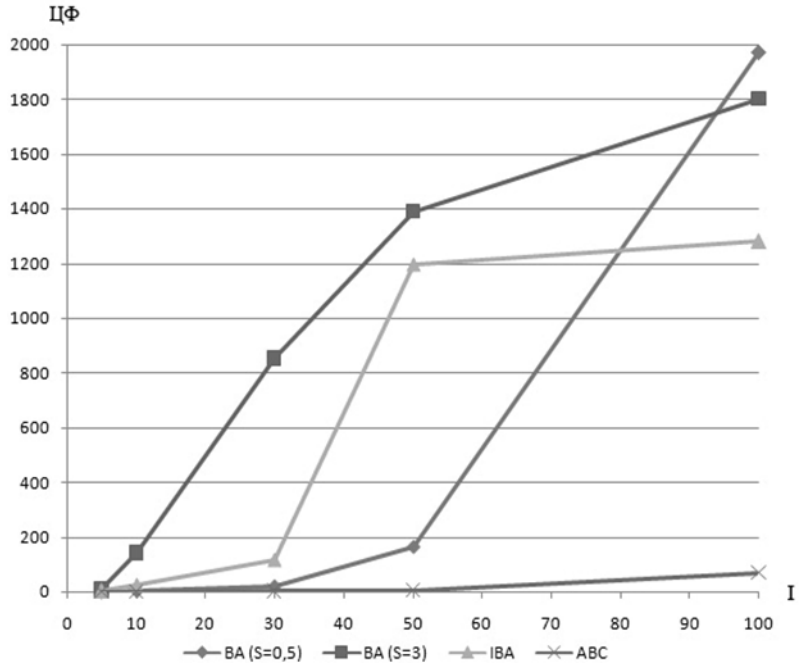

Рисунок 3.5 – Залежність значення цільової функції від кількості ітерацій для функції Розенброку

Слід також зазначити, що алгоритм ABC може забезпечувати знаходження якісних рішень навіть для задач, що характеризуються великою розмірністю.

Ще однією важливою особливістю даного алгоритму є те, що крім основних параметрів будь-якого алгоритму (як-то - розмір популяції, кількість ітерацій) алгоритм колонії бджіл може мати виключно єдиний додатковий параметр, а саме - граничну кількість ітерацій, тоді як бджолиний алгоритм має 5 параметрів (m, e, l, n, S), а покращений бджолиний алгоритм - 6 параметрів.

Отже, підсумкова ефективність функціонування алгоритмів BA та IBA повною мірою залежить від контрольних параметрів.

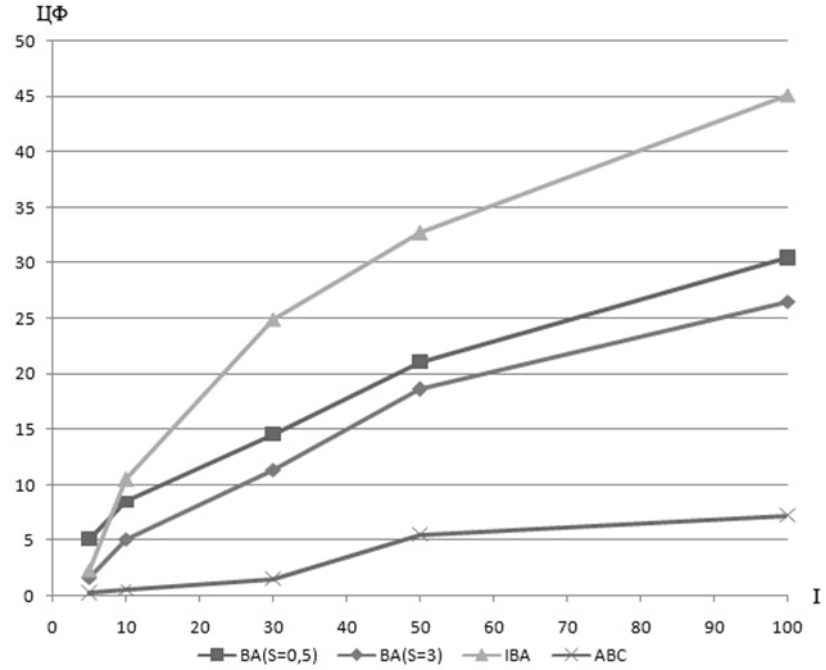

Рисунок 3.6 – Залежність значення цільової функції від кількості ітерацій для функції Растригіна

4 АЛГОРИТМ КОСЯКА РИБ

4.1 Відомості про алгоритм

Для оптимізації процесів, які відбуваються у межах складних систем, сьогодні значний інтерес представляють метаевристичні алгоритми оптимізації, як одна з груп алгоритмів стохастичної групи [9, 11, 17].

Ключовою відмінністю таких алгоритмів від інших є те, що вони не вимагають знання виразу який описує функцію, що підлягає оптимізації. При цьому, можливість сходження такої функції до глобального оптимуму може навіть не бути доведено. Проте, експериментально було визначено, що у більшості випадків такі алгоритми забезпечують можливість знаходження рішення, яке є щонайменше близьким до оптимального у контексту досить широкого переліку завдань.

Як вже зазначалося, певна кількість алгоритмів оптимізації виникла як моделі, які було запозичені у природи. У свою чергу, алгоритм косяка риб також відноситься до даної групи.

Розглянемо принцип роботи даного алгоритму.

Перш за все зазначимо, що у алгоритмі FSS (Fish School Search) область пошуку є акваріумом, у межах якого можуть переміщуватися окремі агенти (риби).

Як відомо зі спостережень, в умовах пошуку їжі реальні риби як у природніх, так і у штучних умовах плавають косяком. Відтак, кінцевою метою виконання алгоритму можемо вважати переміщення усіх агентів до області простору (наприклад, умовний акваріум), яка відповідає екстремуму функції.

До складу алгоритму косяку риб входять такі технологічні етапи, як:

1. Ініціалізація популяції. У ході даного етапу виконується рівномірний розподіл риб у межах акваріуму.

2. Міграція агентів до джерела їжі (аналогія: що більший крок агенти зробили у бік області екстремуму функції, то більше їжі вони отримали)

3. Завершення пошуку.

Умовимося, що дослідження роботи алгоритму буде виконуватися в умовах, коли необхідно вирішити завдання умовної максимізації функції. При цьому слід розуміти, що завдання пошуку мінімальних значень функції може

бути виконано аналогічним чином, орієнтуючись на найменше значення цільового показника.

Далі виконаємо більш детальний огляд роботи алгоритму.

4.2 Деталізація алгоритму

Головний функціонал алгоритму реалізується на етапі міграції агентів.

У свою чергу, процес міграції агентів виконується ітераційно, таким чином, що у ході кожної ітерації виконуються оператори двох груп [11]:

1. Оператори плавання, завданням яких є забезпечення переміщення (плавання) агентів у межах умовного акваріуму.

2. Оператори годування. Їхнім завданням є фіксація успіху дослідження тих чи інших областей акваріума.

Швидкість роботи та точність алгоритму напряму залежать від обраних величин ключових його параметрів, які розглянемо далі.

4.2.1 Ключові параметри алгоритму

Головні параметри алгоритму можна згрупувати до [17]:

− параметрів середовища (акваріуму);

− параметрів мешканців акваріуму (агентів).

Зокрема, до параметрів усього акваріуму у цілому відносяться:

− кількість риб у косяку, або розмір популяції;

− кількість ітерацій, які виконуються на етапі міграції агентів;

− верхня межа пошуку;

− нижня межа пошуку;

− стартовий радіус пошуку їжі навколо агентів;

− підсумковий радіус пошуку їжі навколо агентів;

− найбільша вага агента;

Дані параметри оточення задає користувач користувач. Відтак, маніпулюючи їх величинами може бути забезпечено можливість регулювання співвідношенням точності та часу роботи алгоритму.

У свою чергу, до параметрів самих агентів належать лише 2 величини, а саме:

− позиція агента протягом різних стадій плавання;

− поточна вага агента.

Оскільки алгоритм далі буде розглянуто з позиції програмної реалізації, для розуміння специфіки його функціонування спочатку виконаємо огляд його побудову у вигляді псевдокоду.

Псевдокод алгоритму можемо представити наступним чином [11]:

```
initialize_randomly_all_fish;
while (stop_criterion is not met)
{
for (each_fish)
{
individual_movement;
evaluate_fitness_function;
}
feeding_operator;
for (each_fish)
instinctive_movement;
calculate_barycentre;
for (each_fish) do
{
volitive_movement;
evaluate_fitness_function;
}
update_individidual_step;
}
```

Далі для того, щоб виконати перехід від псевдокоду алгоритму до його функціонального опису, необхідно ввести до розгляду наступні змінні, кожна з яких є окремим параметром з перелічених вище.

А саме [17]:

– кількість рибу косяку - populationSize;

– кількість ітерацій протягом етапу міграції - iterationCount;

– позиція агента на різних стадіях плавання – swimStatePos; - величина поточної ваги агента – weight;

– значення верхньої та нижньої меж пошуку – lowerBoundPoint та higherBoundPoint відповідно;

− величина початкового та кінцевого радіусів пошуку їжі навколо агентів – individStepStart та individStepFinal відповідно;

− найбільша вага агента - weightScale.

Тоді у деталізованому алгоритмі виокремимо такі технологічні кроки, як:

1. Процедура ініціалізація усієї популяції. Дана процедура зводиться до випадкового вибору позиції агента у межах акваріуму (swimStatePos[0]), а також задання значення ваги для усіх агентів, яка дорівнює на даній стадії половині від максимальної, тобто [17]:

$$weight = \frac{weightScale}{2}. \tag{4.1}$$

2. Основний цикл алгоритму, який характеризує стадію міграції агентів до джерела їжі. Тут за замовчуванням у якості критерію зупинки використовується параметр кількості ітерацій протягом етапу міграції, або iterationCount.

3. Стадія індивідуального плавання агентів. Даний крок характеризується тим, що на його протязі усі агенти у межах деякої області навколо себе (individStep) намагаються знайти найкраще значення функції, що еквівалентно виразу:

$$swimStatePos[1] = swimStatePos[1] + rand(-1;1) \times individStep. \tag{4.2}$$

При цьому, якщо протягом поточного кроку найкраще значення функції знаходиться, даний крок фіксується.

В іншому випадку вважається, що попередньо не було виконано переміщення, як показує наступний вираз:

$$if\ \ \mathrm{f}(swimStatePos[1]) < f(swimStatePos[0]) \mid (!swimStatePos[1].IsIn\,\mathrm{Re}\,gion) \tag{4.3}$$

4. Закріплення результатів, отриманих у ході індивідуальної стадії плавання. Для цього використовується параметр "Вага". У свою чергу, значення цього параметру дорівнює зміні функції пристосованості даного агента до і після індивідуальної стадії, нормованого максимальною зміною функції серед популяції, тобто:

$$weight = \frac{\Delta f_i}{\max \Delta f}. \quad (4.4)$$

Слід зазначити, що відмінною особливістю розглядуваного алгоритму є те, що відсутня необхідність запам'ятовувати найкращих агентів на попередніх ітераціях.

5. Інстинктивно-колективна стадія плавання. На даному кроці виконується розрахунок величини загального кроку m міграції для усього косяка риб, для чого використовується наступний вираз:

$$m = \frac{\sum_{i=1}^{populationSize} ((swimStatePos_i[1] - swimStatePos_i[0]) \times \Delta f_i)}{\sum_{i=1}^{populationSize} \Delta f_i}. \quad (4.5)$$

Сенс даного кроку полягає у наступному. Протягом кроку на кожного з агентів здійснює вплив уся популяція у цілому. При цьому, рівень впливу окремо узятого агенту є пропорційним його успіхам протягом індивідуальної стадії переміщення. Далі виконується переміщення усієї популяції на раніше обчислену величину m за наступним принципом [17]:

$$swimStatePos[2] = swimStatePos[1] + m. \quad (4.6)$$

6. Розрахунок центру ваги barycenter усього косяка перед наступним кроком плавання відповідно до виразу:

$$barycenter = \frac{\sum_{i=1}^{populationSize} (swimStatePos_i[2] \times weight_i)}{\sum_{i=1}^{populationSize} weight_i}. \quad (4.7)$$

7. Стадія колективно-індивідуального плавання. На даній стадії необхідно виконати розрахунок ваги популяції, щоб відстежити її зміну відносно передуючої ітерації.

У цьому випадку, якщо вага збільшилася, це свідчить про те, що популяція наблизилася до області максимуму функції. За цієї умови необхідно виконати звуження кола пошуку, тим самим проявлюються інтенсифікаційні характеристики моделі.

З іншого боку, коли спостерігається зменшення ваги косяка, це свідчить про те, що агенти виконують пошук максимуму у області. Яка є віддаленою від нього. Відтак, тут необхідно змінити напрямок траєкторії, тим самим скориставшись диверсифікаційними властивостями моделі.

У свою чергу, для переміщення агентів на стадії індивідуально-колективного плавання рекомендовано використовувати значення кроку, яке є удвічі більшим значення індивідуального кроку пошуку [11].

При цьому, позиція агента на даній стадії розраховується згідно з виразом:

$$swimStatePos[3] = swimStatePos[2] \pm collStep \times rand(0;1) \times \\ \times \frac{swimStatePos[2] - barycentre}{dist(swimStatePos[2];\ barycentre)}, \quad (4.8)$$

де $collStep$ – крок переміщення;

$dist$ - оператор, який виконує розрахунок відстані між двома точками в евклідовому просторі.

8. Лінійне скорочення кроку індивідуального пошуку для наступної ітерації, яке здійснюється за такою формулою:

$$individStep = \frac{individStepStart - individStepFinal}{iterationCount}. \quad (4.9)$$

4.3 Тестування алгоритму

Для того, щоб дослідити процес роботи алгоритму, та зробити висновки відносно його ефективності, розглянемо 2 різні функції, наприклад, такі, як [18]:

$$y = -(0{,}1x^2 + 0{,}1z^2) + 10, \quad (4.10)$$

та:

$$y = 10 \times (\sin 0{,}1x + \sin 0{,}1z), \quad (4.11)$$

Для тестування встановлюються такі параметри:

– область пошуку: нижня межа, lowerBoundPoint (-100; -100), higherBoundPoint (100; 100);

– кількість ітерацій, або iterationCount: 100;

– розмір популяції populationSize: 50;

– початковий individStepStart індивідуальний крок: 10;

– кінцевий individStepFinal індивідуальний крок: 0.1;

– максимальна вага weightScale риби: 50.

Для даної функції, у результаті роботи алгоритму найбільше значення функції було рівним близько 9,999978. У свою чергу, середнє значення функції – порядку 9,999148. У цьому разі графік зміни величини середнього значення функції виглядає так, як показано на рисунку 4.1.

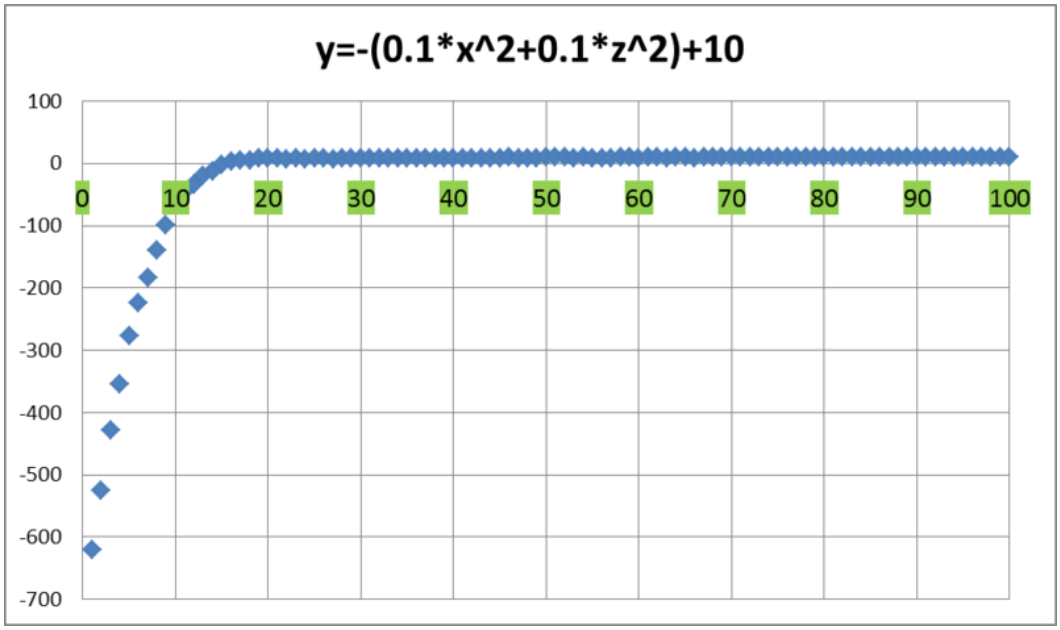

Рисунок 4.1 – Графік досліджуваної функції

Як можна бачити з графіку на рисунку 4.1, вже на 15-й ітерації косяк риб знаходився у межах позитивній півплощини за віссю 0Y.

Разом з тим, середнє значення функції не зменшувалася нижче 9-ти вже з 48-ї ітерації [18].

У динаміці процес пошуку максимуму функції зображено на рисунку 4.2.

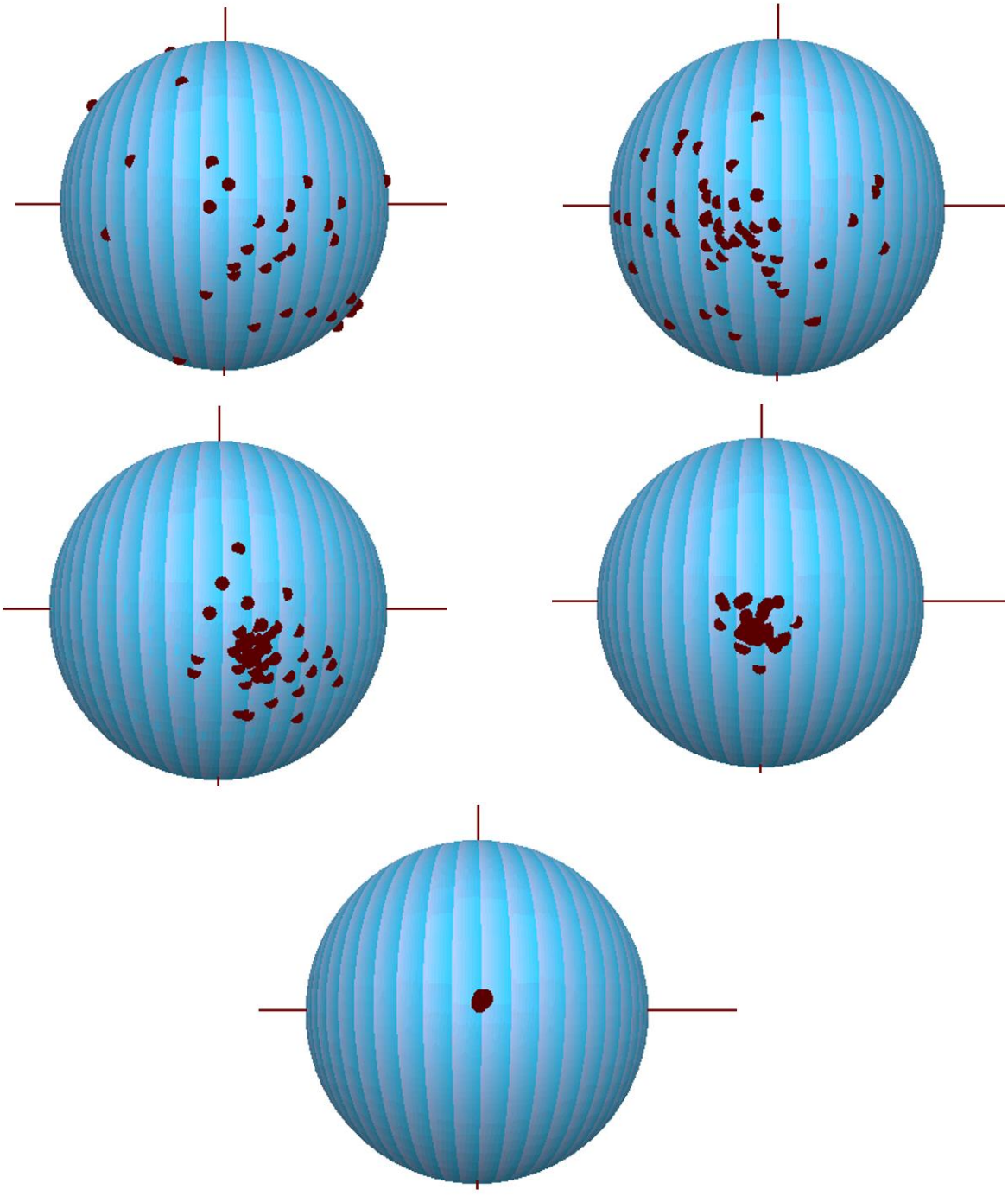

Рисунок 4.2 – Переміщення агентів у процес пошуку максимуму функції на 10-й а), 20-й б), 30-й в), 80-й г) та 100-й ітерації д)

Для тестування функції $y = 10 \times (\sin 0{,}1x + \sin 0{,}1z)$ використовуються параметри, аналогічні попередньому випадку [18].

На відміну від попередньо розглянутої функції, дана функція є багатоекстремальною.

За результатами роботи алгоритму визначено, для даної функції найбільша її величина була рівна 19,999996, тоді як середня величина - 19,9994.

Графік зміни середньої величини функції показано на рисунку 4.3.

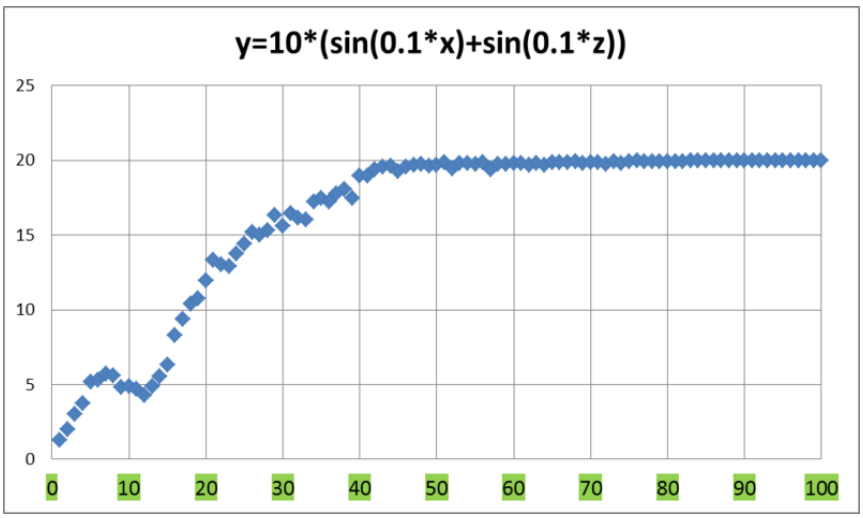

Рисунок 4.3 – Графік досліджуваної функції

З графіку на рисунку 4.3 можна бачити, що з 42 кроку ітерації середня величина функції знаходиться на рівні, що перевищує 19.

Динаміка розвитку процесу пошуку максимуму функції для даного випадку показано на рисунку 4.4.

Таким чином, як бачимо, в обох випадках спостерігається гарантоване знаходження максимумів функцій.

Виходячи з того, що в обох випадках використано нетривіальні функції (зокрема, функція, задана виразом (4.11), є подібною до функції Растригіна), при цьому, в обох випадках метод продемонстрував свою результативність, можна вважати його ефективним для рішення універсальних завдань з опттимізації.

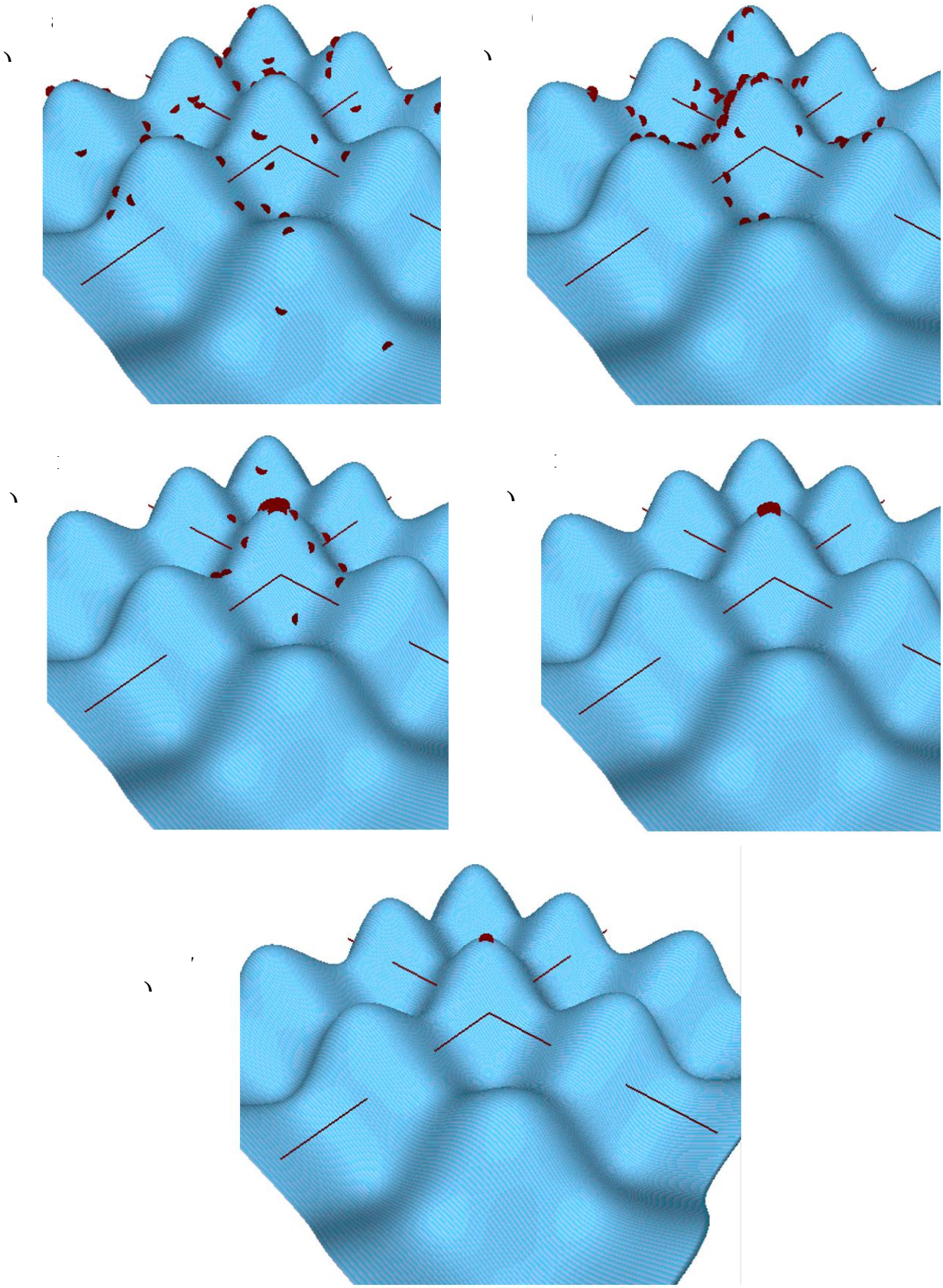

Рисунок 4.5 - Переміщення агентів у процес пошуку максимуму функції на
5-й а), 10-й б), 30-й в), 50-й г) та 100-ї ітерації д)

ВИСНОВКИ

Головне завдання, яке мало бути вирішено у ході виконання даної кваліфікаційної роботи, зводилося до дослідження методів стохастичної оптимізації.

Вибір саме цих оптимізаційних методів було зумовлено тим, що саме методи даної групи здатні виконувати обробку процесів, математичний опис яких є або занадто складним, або умовно-формалізованим для того, щоб застосовувати методи, що належать іншим групам.

Зокрема, було досліджено:

− метод імітації відпалу;

− алгоритм косяку риб;

− алгоритм бджолиного рою.

У ході дослідження було визначено, що у першому випадку було досліджено алгоритм, що імітує фізичний процес охолодження металу після розігріву.

Водночас, два інших алгоритму імітують поведінку т.з. «ройового інтелекту».

Кожен з цих алгоритмів, при цьому, у сутності, містить у собі ту чи іншу кількість агентів (риб, бджіл і т.д.), які здатні виконувати як індивідуальні дії, так і дії групового характеру.

У підсумку, загальне рішення завдань оптимізації на базі алгоритмів ройового інтелекту визначається як результатами індивідуальних, так і колективних дій агентів.

При цьому, процес пошуку екстремумів функцій на базі алгоритму косяку риб було розглянути у динаміці.

У свою чергу, для методу імітації відпалу, а також для алгоритму бджолиного рою, досліджено математичне та алгоритмічне підгрунтя побудови.

Результати виконаних досліджень доводять, що:

− методи, які було розглянуто, теоретично можуть бути використані для оптимізації будь-якого процесу. Незалежно від його складності;

− сходимість алгоритмів, їх швидкодія та підсумкова точність значною мірою залежать від вибору налаштувань, які встановлює користувач;

− якщо зробити припущення, що для рішення тієї чи іншої оптимізаційної задачі може бути використано не абсолютне значення екстремуму функції і відповідного аргументу, а, деякою мірою, наближені величини, можна говорити про те, що стохастичні методи, за цих умов, гарантують можливість рішення.

Отже усі завдання до кваліфікаційної роботи опрацьовано повною мірою.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. How Machine Learning Can Improve Telecommunications [Електронний ресурс] - Режим доступу: https://www.linkedin.com/pulse/how-machine-learning-can-improve-telecommunications-ugonna-alinnor?ysclid=lyvlx7i0nh576533072.

2. Telecommunications and Machine Learning: Enhancing Network Performance [Електронний ресурс] - Режим доступу: https://moldstud.com/articles/p-telecommunications-and-machine-learning-enhancing-network-performance.

3. Machine Learning for Telecommunication [Електронний ресурс] - Режим доступу: https://s3.amazonaws.com/solutions-reference/machine-learning-for-telecommunication/latest/machine-learning-for-telecommunication.pdf.

4. Мерфі К. Ймовірнісне машинне навчання. ДМК-Прес. 2023, 990 с. ISBN: 978-5-93700-119-1.

5. Малярець Л.М. Дослідження операцій та методи оптимізації : практикум у 2-х ч. Частина 1– Харків : ХНЕУ ім. С. Кузнеця, 2017. – 169 с.

6. Панченко С.В., Медиченко М.П., Лисечко В.П. Методи оптимізації та моделювання: Навч. Посібник. – Харків: УкрДАЗТ, 2015. – Ч.1. – 128 с.

7. Ладієва Л.Р. Методи оптимізації та пошуку оптимальних рішень. Навч. Посібник. – Київ, :КПІ ім. Ігоря Сікорського, 2023. – 75 с.

8. Хуттер Ф., Кроттхофф Л., Ваншорен Х. Введення в автоматизоване машинне навчання (AutoML). ДМК-Прес, 2023. 256 с. ISBN: 978-5-93700-196-2.

9. Лебедєв Б.К., Лебедєв В.Б. Планування на основі ройового інтелекту і генетичної еволюції. Інтелектуальні САПР. – 2009. – № 4 (93). – С. 25-33.

10. Курейчик В.В., Курейчик В.М., Родзін С.И. Концепція еволюційних обчислень, інспірованих природними системами. Інтелектуальні САПР. – 2009. – № 4 (93). – С. 16-24.

11. Fish school search algorythm [Електронний ресурс] - Режим доступу: www.fbin.pro.br/fss.

12. Ципкін Я.З. Основи теорії систем, що навчаються. - М.: Наука. - 1970.

13. Скобцов Ю.А. Основи еволюційних обчислень. - Донецьк: ДонНТУ, - 2008. - 326с.

14. Pham D.T. The Bees Algorithm. Technical Note, Manufacturing

Engineering Centre, Cardiff University, UK, 2005.

15. Basturk B. An artificial bee colony (abc) algorithm for numeric function optimization. IEEE Swarm Intelligence Symposium 2006. – Indianapolis, Indiana, USA, 2006.

16. Karaboga D. An idea based on honey bee swarm for numerical optimization. Technical Report TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

17. Fish school search as swarm intelligence algorythm example [Електронний ресурс] - Режим доступу:
http://cs.fit.edu/~rmenezes/Teaching/Entries/2011/8/22_OLD__CSE5801__Swarm_Intelligence_files/01930261.pdf

18. FishSchoolSearch [Електронний ресурс] - Режим доступу: https://bitbucket.org/ankulikov/fishschoolsearch/src/master