Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження гібридного методу правдоподібної кластеризації на основі еволюційного алгоритму
(тема)

Виконав:
студент 2 курсу, групи СШІм-22-3
Пелюшок Б.В.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)
Освітня програма Системи штучного інтелекту
(повна назва спеціалізації)
Керівник доц. Шафроненко А.Ю.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ В.О. Філатов
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

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

Освітня програма Системи штучного інтелекту
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Пелюшку Богдану Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження гібридного методу правдоподібної кластеризації на основі еволюційного алгоритму

затверджена наказом університету від 1 квітня 20 24 р. № 260Ст

2. Термін подання студентом роботи до екзаменаційної комісії 12 червня 20 24 р.

3. Вихідні дані до роботи Науково-технічні публікації, дані Інтернет-джерел та відомих наукових проєктів, документація Python, набір синтетичних даних для тренування та тестування системи, бібліотеки для наукових обчислень та візуалізації (NumPy, Plotly), MongoDB для зберігання параметрів і результатів.

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної галузі
2) Постановка задачі
3) Опис програмного рішення
4) Опис результатів досліджень

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	1.04.2024	виконано
2	Аналіз предметної галузі	2.04.2024	виконано
3	Огляд існуючих метрик оцінювання якості	03.04.2024	виконано
4	Аналіз підходів вирішення задачі детекції	05.04.2024	виконано
5	Експериментальне моделювання та навчання	14.04.2024	виконано
	моделі		
6	Написання пояснювальної записки	23.04.2024	виконано
7	Перевірка на академічний плагіат	02.05.2024	виконано
8	Нормоконтроль	04.05.2024	виконано
9	Підготовка презентації та доповіді	10.05.2022	виконано
10	Попередній захист	11.05.2024	виконано
11	Рецензування	14.05.2024	виконано
12	Захист перед ЕК	12.06.2024	виконано

Дата видачі завдання 1 квітня 20 24 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Шафроненко А.Ю.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 96 с., 25 рис., 1 табл., 2 дод., 25 джерел.

ГІБРИДНИЙ МЕТОД, ДАНІ, ЕВОЛЮЦІЙНИЙ АЛГОРИТМ, ОПТИМІЗАЦІЯ, ПРАВДОПОДІБНА КЛАСТЕРИЗАЦІЯ.

Об'єкт дослідження – розробка та апробація гібридного методу для правдоподібної кластеризації даних.

Предмет дослідження – застосування еволюційних алгоритмів для покращення результатів кластеризації даних.

Мета роботи – розробка та оптимізація гібридного методу, який поєднує в собі переваги правдоподібної кластеризації та еволюційних алгоритмів для підвищення якості кластеризації даних.

Методи дослідження – аналіз наукової літератури з області кластерного аналізу, розробка гібридного методу на основі еволюційного алгоритму та правдоподібної кластеризації, експериментальне тестування методу на штучних даних, порівняльний аналіз результатів.

Проведено теоретичний аналіз принципів правдоподібної кластеризації та еволюційних алгоритмів, а також їхніх переваг та недоліків. Розроблено гібридний метод, який базується на комбінації цих двох підходів з метою отримання більш точних та стабільних результатів. Практичні експерименти показали ефективність запропонованого методу на різних типах даних, включаючи великі та складні набори даних.

Отримані результати підтверджують, що гібридний метод правдоподібної кластеризації на основі еволюційного алгоритму є ефективним і має потенціал для використання в різних областях, де необхідно робити аналіз та групування великих обсягів даних.

ABSTRACT

Master’s thesis contains: 96 pp., 25 fig., 1 tabl., 1 ann., 25 references.

DATA, EVOLUTIONARY ALGORITHM, HYBRID METHOD, OPTIMIZATION, PLAUSIBLE CLUSTERIZATION.

The object of research is the development and testing of a hybrid method for plausible data clustering.

The subject of the study is the application of evolutionary algorithms to improve the results of data clustering.

The purpose of the work is to develop and optimize a hybrid method that combines the advantages of plausible clustering and evolutionary algorithms to improve the quality of data clustering.

Research methods – analysis of scientific literature in the field of cluster analysis, development of a hybrid method based on evolutionary algorithm and plausible clustering, experimental testing of the method on artificial data, comparative analysis of results.

A theoretical analysis of the principles of plausible clustering and evolutionary algorithms, as well as their advantages and disadvantages, has been carried out. A hybrid method has been developed, which is based on the combination of these two approaches in order to obtain more accurate and stable results. Practical experiments have shown the effectiveness of the proposed method on various types of data, including large and complex data sets.

The obtained results confirm that the hybrid method of plausible clustering based on the evolutionary algorithm is effective and has the potential to be used in various areas where it is necessary to analyze and group large amounts of data.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних;

ШІ – штучний інтелект;

AI – Artificial Intelligence – штучний інтелект;

API – Application Programming Interface – інтерфейс програмування додатків;

CGWO – Chaotic Gray Wolf Optimizer – хаотичний алгоритм сірого вовка;

DB – Database – база даних;

DBSCAN – Density-Based Spatial Clustering of Applications with Noise – просторове кластерування застосувань з шумом на основі густини;

GWO – Gray Wolf Optimizer – алгоритм сірого вовка;

HTML – HyperText Markup Language – мова розмітки гіпертексту;

PSO – Particle Swarm Optimization – оптимізація рою частинок.

ВСТУП

У сучасному світі обробка та аналіз даних відіграють важливу роль у різних сферах, включаючи науку, бізнес, медицину та інформаційні технології. Одним з ключових завдань аналізу даних є їхнє класифікування та групування у відповідності до спільних ознак чи характеристик. У цьому контексті кластерний аналіз виступає як ефективний інструмент для виявлення схожих патернів або груп в наборах даних.

Не зважаючи на успіх існуючих методів кластеризації, існують певні обмеження, які впливають на їхню ефективність [1]. Один з таких обмежень полягає у складності вибору оптимального числа кластерів та недостатній обробці шуму та викидів у даних.

Щоб подолати ці проблеми, виникає необхідність в розробці нових підходів та методів, які б забезпечували кращу якість кластеризації та були б менш чутливими до шуму [2].

Одним із потенційних рішень цих проблем є використання гібридних методів, які комбінують в собі переваги різних підходів до кластеризації. Еволюційні алгоритми, відомі своєю здатністю до пошуку оптимальних рішень у складних просторах, можуть бути ефективними інструментами для вдосконалення результатів кластеризації [3] шляхом оптимізації параметрів алгоритмів або навчання структур кластерів.

В рамках цього дослідження будуть розглянуті основні принципи та підходи до кластерного аналізу, описано еволюційні алгоритми та їхні можливості у контексті вдосконалення процесу кластеризації. Також буде надано огляд існуючих гібридних методів у цій області та обговорено їхні переваги та недоліки. Нарешті, буде представлено опис розробленого гібридного методу, його алгоритму та особливостей використання.

Зазначено, що однією з основних перешкод у застосуванні класичних методів кластеризації є вибір оптимального числа кластерів. Це завдання може бути нетривіальним, особливо коли дослідник має справу з даними

великого обсягу або з непередбачуваними структурами. Деякі алгоритми вимагають попередньо заданого числа кластерів, що може призвести до неправильних або неповних результатів, особливо у випадку, коли структура даних складна або неоднорідна.

Попередня обробка даних є ще однією ключовою складовою процесу кластеризації. Вона може включати в себе широкий спектр дій, таких як видалення викидів, нормалізація ознак, вибір репрезентативних ознак та інші. Важливою є правильна підготовка даних до подальшого аналізу, оскільки це може суттєво вплинути на якість та стабільність результатів кластеризації. Гібридні методи кластеризації, які комбінують в собі різні підходи та алгоритми, є перспективним напрямком досліджень. Такі методи можуть поєднувати переваги різних алгоритмів та компенсувати їхні недоліки, що призводить до отримання більш точних та стабільних результатів кластеризації.

У цьому дослідженні основна увага буде приділена розробці гібридного методу кластеризації на основі еволюційного алгоритму. Еволюційні алгоритми, такі як генетичні алгоритми, відомі своєю здатністю до пошуку оптимальних рішень в складних просторах параметрів. Їхня застосовність у вдосконаленні результатів кластеризації полягає у здатності адаптуватися до змінних умов та навколишнього середовища.

Також у цьому дослідженні буде враховано важливі аспекти оптимізації параметрів еволюційного алгоритму, щоб забезпечити ефективне функціонування методу та збільшити його швидкодію. Це включає в себе вибір відповідних параметрів алгоритму, таких як розмір популяції, ймовірність мутації та кросоверу, а також критерії зупинки алгоритму та методи обробки результуючих кластерів.

Такий гібридний підхід може бути ефективним у вирішенні проблеми вибору оптимального числа кластерів та зменшенні чутливості до шуму в даних. Окрім того, він може виявити потенціал у різних сферах застосування:

- аналіз соціальних мереж;
- медична діагностика;
- фінансовий аналіз тощо.

Дослідження такого методу може принести значний внесок у розвиток області кластерного аналізу та відкрити нові можливості для використання аналізу даних у практичних застосуваннях. Враховуючи швидке зростання обсягів та складності даних у сучасному світі, розвиток нових ефективних методів кластеризації стає актуальною та важливою задачею.

2 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

2.1 Огляд існуючих методів кластеризації

У сфері аналізу даних існує безліч методів та алгоритмів кластеризації, які використовуються для групування схожих об'єктів у наборі даних. Один з найпоширеніших методів – k-середніх (k-means), який базується на припущенні, що кожен кластер представляється своїм центром, а об'єкти в кластері подібні за відстанню до центру. Цей метод добре працює з великими наборами даних, але потребує попереднього визначення кількості кластерів [4].

Інший популярний метод – ієрархічна кластеризація, яка групує об'єкти у вигляді дерева подібності [5]. Цей підхід не вимагає визначення кількості кластерів перед виконанням алгоритму, але може бути вимогливим до обчислювальних ресурсів для великих наборів даних.

Також варто зазначити алгоритм DBSCAN, який здатний автоматично визначати кількість кластерів у даних, а також розпізнавати шумові точки [6]. Цей метод особливо ефективний для даних з неоднорідною густотою кластерів.

Окрім цього, існують інші методи, такі як агломеративна кластеризація, спектральна кластеризація, методи основані на вибіркових процедурах та багато інших. Кожен з цих методів має свої переваги та недоліки, і вибір конкретного методу залежить від специфіки даних та поставленої задачі кластеризації.

Для задачі правдоподібної кластеризації на основі еволюційного алгоритму варто ретельно вивчити і порівняти існуючі методи, щоб обрати найбільш підходящий для вирішення поставленої проблеми. Дані методи можуть включати не лише класичні алгоритми, а й новаторські підходи, які можуть бути адаптовані для використання в гібридному методі.

У сфері аналізу даних постійно з'являються нові підходи до

кластеризації, оскільки завдання та вимоги до обробки даних постійно змінюються. Однак, не всі методи показують однакову ефективність в усіх сценаріях. Наприклад, метод k-середніх часто має проблеми з визначенням правильної кількості кластерів та може давати неправильні результати у випадках, коли кластери мають складну форму або різну густину. Також, методи ієрархічної кластеризації можуть бути обмежені у використанні для великих наборів даних через високу обчислювальну складність. Тому потребується пошук нових підходів, які б ураховували ці обмеження та забезпечували найкращу ефективність у різних умовах застосування.

Однією з передових областей у сфері кластеризації є використання генетичних алгоритмів та інших еволюційних стратегій. Генетичні алгоритми імітують процеси природного відбору та генетичної мутації для вирішення оптимізаційних задач [7]. Вони дозволяють знаходити оптимальні або наближені до оптимальних рішення для складних задач, де традиційні методи можуть бути неефективними. У контексті кластеризації, еволюційні алгоритми можуть адаптуватися до різноманітних структур даних, оптимізувати кількість кластерів та їх внутрішню структуру без попереднього знання про дані. Це робить їх особливо цінними для задач, де традиційні методи не виявляють ефективності або потребують додаткових даних для налаштування параметрів.

Іншим новаторським напрямком є використання методів машинного навчання з вчителем для підвищення ефективності кластеризації [8]. Наприклад, методи передачі навчання можуть бути застосовані для адаптації існуючих кластеризованих моделей до нових наборів даних без необхідності повного перенавчання. Такий підхід дозволяє значно зменшити обчислювальні витрати та підвищити точність кластеризації за рахунок використання апріорного знання. Комбінація еволюційних алгоритмів з техніками машинного навчання відкриває нові можливості для розробки гібридних методів, які можуть ефективно адаптуватися до складних та динамічно змінних даних.

2.2. Аналіз особливостей та викликів використання еволюційних алгоритмів для кластеризації даних

Еволюційні алгоритми представляють собою унікальний і гнучкий підхід до розв’язання задачі кластеризації, виходячи з принципів еволюції та природного відбору [9]. Ці алгоритми, які включають генетичні алгоритми, еволюційні стратегії, та інші методи, застосовують популяційний підхід, де кожний індивідуум представляє потенційне рішення задачі, а його пристосованість визначається згідно з обраною метрикою якості кластеризації. Однак, попри свою адаптивність та гнучкість, еволюційні алгоритми стикаються з низкою особливостей та викликів, які необхідно враховувати при їх застосуванні для кластеризації даних.

Однією з ключових особливостей еволюційних алгоритмів є їх здатність до глобального пошуку у просторі рішень, що робить їх особливо цінними для виявлення оптимальних або близьких до оптимальних рішень у складних ландшафтах, де традиційні методи можуть легко застрягти в локальних оптимумах. Такий підхід дозволяє ефективно впоратися з задачами, які характеризуються великою кількістю параметрів та високою складністю, як-от кластеризація даних з нерівномірною густотою або складною структурою.

Водночас, використання еволюційних алгоритмів для кластеризації супроводжується певними викликами. По-перше, налаштування параметрів алгоритму, таких як розмір популяції, ймовірності мутації та кросоверу, може суттєво впливати на його ефективність та здатність знаходити оптимальні рішення [10]. Неправильно обрані параметри можуть призвести до передчасної конвергенції або, навпаки, до надмірної експлорації без досягнення задовільних результатів. По-друге, обчислювальні витрати еволюційних алгоритмів можуть бути високими, особливо для великих наборів даних та складних моделей, через необхідність обчислення

пристосованості для великої кількості індивідуумів протягом багатьох поколінь.

Ще одним важливим аспектом є забезпечення різноманітності популяції, що є критично необхідним для запобігання ранній конвергенції та забезпечення глибокої експлорації простору рішень. Методи, такі як нішування або катастрофічний відбір, можуть бути використані для збереження різноманітності [11] та стимулювання пошуку нових, потенційно цікавих регіонів простору рішень.

Інші виклики впровадження еволюційних алгоритмів:

- передчасна конвергенція;
- вибір функції пристосованості;
- адаптація до специфіки даних;
- інтерпретація результатів.

На закінчення, попри виклики, еволюційні алгоритми пропонують потужний інструментарій для вирішення задач кластеризації, здатний адаптуватися до різноманітних умов та вимог. Їхня гнучкість та здатність до глобального пошуку роблять їх незамінними для складних задач аналізу даних, де необхідно знайти баланс між експлорацією та експлуатацією, щоб досягти оптимальних результатів. Подальші дослідження та розробка в цій області можуть відкрити нові можливості для покращення ефективності кластеризації та знаходження інноваційних рішень для аналізу даних.

3 ПОСТАНОВКА ЗАДАЧІ

3.1 Класифікація обраної задачі

Задача дослідження гібридного методу правдоподібної кластеризації на основі еволюційного алгоритму може бути класифікована як комплексна задача аналізу даних, спрямована на групування великих масивів даних, що надходять на опрацювання в різних режимах (пакетному та онлайн), з використанням правдоподібного підходу. Метою є виявлення глобальних екстремумів цільової функції правдоподібної нечіткої кластеризації за допомогою модифікації еволюційного алгоритму, який інтегрує в собі переваги еволюційних алгоритмів та глобального випадкового пошуку.

Ця задача включає в себе кілька ключових аспектів, які визначають її класифікацію в науковій сфері [12]. Розглянемо її особливості.

По перше, задача акцентується на правдоподібній нечіткій кластеризації, використовуючи теорію довіри як основу для групування даних, що вимагає спеціального підходу до оцінки рівнів належності та визначення центроїдів-прототипів класів.

Обрана методологія включає застосування еволюційних алгоритмів для оптимізації цільової функції кластеризації, що дозволяє ефективно обходити проблеми локальних оптимумів та знаходити глобально оптимальні рішення в умовах багатоекстремальності.

Використання гібридного підходу, який інтегрує ідеї теорії довіри, еволюційних алгоритмів та глобального випадкового пошуку, вказує на комплексний та інноваційний характер дослідження, спрямований на підвищення ефективності та точності процесу кластеризації.

Особлива увага приділяється адаптивності методу до різних умов опрацювання даних (пакетний та онлайн режими) та його високій швидкодії, що забезпечує високу адаптивність методу до різноманітних умов обробки даних. Важливою особливістю є також використання

концепції нечіткої належності, яка розраховується за допомогою функції сусідства, що забезпечує гнучкість у визначенні рівнів приналежності векторів-спостережень до кластерів.

Ключовим елементом у вирішенні задачі правдоподібної нечіткої кластеризації є застосування модифікованого алгоритму «сірих вовків», який є одним із швидкодіючих еволюційних алгоритмів. Цей алгоритм, вдосконалений за допомогою введення додаткової випадкової збуреності для забезпечення глобального пошуку, показує високу ефективність у відшуканні глобальних екстремумів цільової функції. Такий підхід збільшує ймовірність знаходження оптимальних рішень у складних умовах багатоекстремальної оптимізації, що підтверджено результатами експериментальних досліджень [13].

Експериментальні дослідження методу на основі відомих багатоекстремальних функцій, таких як функція Аклі, демонструють його високу точність і швидкість пошуку глобального максимуму. Порівняльний аналіз з іншими алгоритмами ройового інтелекту, такими як PSO (Particle Swarm Optimization) та GWO (Grey Wolf Optimizer), підтверджує переваги запропонованого методу з точки зору швидкості збіжності та точності пошуку.

У таблиці 3.1 два алгоритми ройового інтелекту (алгоритм PSO та алгоритм GWO) вибрано для проведення симуляційних контрастних експериментів із запропонованим CGWO, щоб перевірити його перевагу щодо швидкості конвергенції та точності пошуку.

Таблиця 3.1 – порівняльна статистика алгоритмів [14]

Accuracy	CGWO	GWO	PSO
Best	1.5099e − 017	7.5495e − 013	0.0035
Ave	1.2204e − 016	1.0048e − 012	0.0055
Worst	2.2204e − 014	1.4655e − 013	0.0082

Такий комплексний підхід, що об'єднує правдоподібну нечітку кластеризацію з еволюційними алгоритмами та глобальним випадковим пошуком, відкриває нові перспективи для вирішення задач аналізу даних в умовах великої розмірності та складності .

3.2 Гібридні підходи до кластеризації

Гібридні підходи до кластеризації даних об'єднують в собі різноманітні методи та алгоритми з метою покращення результатів кластеризації, адаптації до специфічних умов задачі та збільшення ефективності обробки даних. Перечислимо основні методи.

Метод кластеризації, що використовує алгоритм «Fish schools», базується на імітації поведінки риб, які діють як окремі індивіди та в групах для оптимізації пошуку їжі, що відповідає знаходженню локальних максимумів у функції розподілу густини даних. Алгоритм включає індивідуальні, інстинктивно-колективні та вольові колективні рухи, які сприяють ефективному знаходженню глобальних та локальних екстремумів, поліпшуючи процес кластеризації за рахунок зменшення часу пошуку та покращення якості кластеризації [15].

Метод кластеризації на основі «Скажених вовків» використовує модифікований алгоритм сірих вовків, що інтегрує еволюційні алгоритми та глобальний випадковий пошук [14] для визначення глобальних екстремумів цільової функції кластеризації. Вовки в алгоритмі імітують поведінку реальних тварин під час полювання, де вони спочатку відстежують, переслідують, оточують, а потім атакують здобич, що дозволяє ефективно досліджувати простір пошуку та знаходити оптимальні рішення у складних многовершинних функціях [13].

Метод «Зграї котів» (рисунок 3.1) інтегрує ідеї випадкового пошуку з еволюційними стратегіями [16], використовуючи поведінку котів, які ведуть себе активно (полювання) та пасивно (спостереження).

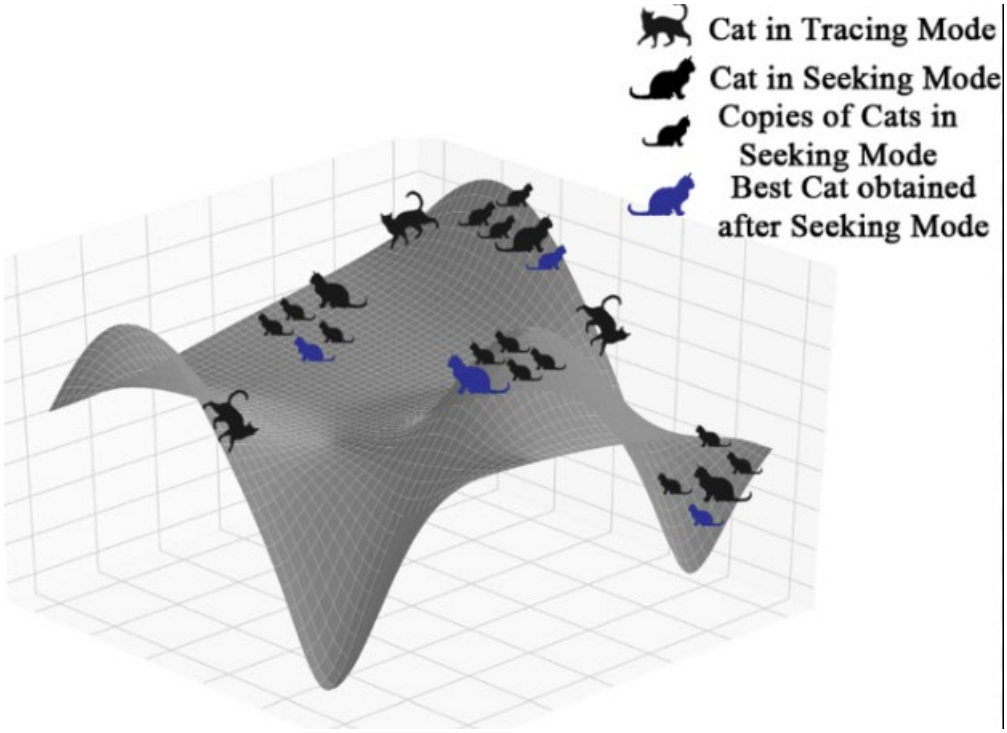

Рисунок 3.1 – Ілюстрація методу cat swarm

Цей підхід дозволяє алгоритму ефективно використовувати обидва режими для балансування між дослідженням та експлуатацією пошукового простору, значно покращуючи здатність знаходити оптимальні рішення у складних умовах багатоекстремальної оптимізації. Метод особливо ефективний при роботі з даними, які мають велику кількість локальних оптимумів, забезпечуючи високу точність [17].

Спільними рисами багатьох гібридних підходів є їх здатність ефективно впоратися з великими обсягами даних, забезпечення гнучкості в налаштуванні параметрів та інтеграція переваг як традиційних алгоритмів, так і сучасних методів машинного навчання.

Основною метою гібридного методу правдоподібної кластеризації, який розглядається в дослідженні, є пошук глобальних екстремумів цільової функції кластеризації, використовуючи модифікацію алгоритму сірих вовків, що комбінує переваги еволюційних алгоритмів та глобального

випадкового пошуку . Цей підхід дозволяє реалізувати високу швидкість та надійність в задачах багатоекстремальної фазової кластеризації.

У випадку використання гібридного методу, що базується на комбінованому еволюційному методі рибних косяків для кластеризації векторних та матричних масивів даних, основою є оптимізація функцій розподілу густини даних у цих масивах [18]. Це дозволяє знизити кількість запусків процедури оптимізації, знаходити екстремуми складних функцій з багатьма екстремумами та є простим у чисельному втіленні. Одним із ключових елементів гібридного підходу є інтеграція методів рандомізованого пошуку та еволюційної оптимізації. Такий підхід дозволяє забезпечити високу точність та швидкість обробки даних [19], що є особливо важливим у випадках, коли дані характеризуються великою розмірністю та складністю.

3.3 Визначення вимог до функціоналу

Для реалізації гібридного методу правдоподібної кластеризації, система повинна бути спроектована так, щоб задовольняти низку ключових вимог, які забезпечать її ефективність, гнучкість та коректність в роботі. Ці вимоги визначають функціональність системи і сприяють досягненню оптимальних результатів в процесі кластеризації вибірок даних.

Перш за все, система має забезпечувати підтримку обробки великих масивів даних у режимі реального часу [20]. Це означає, що алгоритм має бути оптимізований для швидкої обробки даних, що надходять, з мінімальною затримкою, що дозволяє виконувати кластеризацію в пакетному та онлайн режимах. Іншою важливою вимогою є забезпечення високої точності виявлення кластерів за допомогою реалізації гібридного методу, який включає правдоподібну нечітку кластеризацію та еволюційні алгоритми. Система повинна здатна адаптуватися до різноманітних структур даних і ефективно виявляти кластери навіть у складних умовах

багатоекстремальності цільової функції.

Система має надавати користувачу можливість налаштовувати ключові параметри, такі як кількість кластерів, метрику відстані, параметри фазифікації та інші [21], що дозволить користувачу оптимізувати процес кластеризації під конкретні задачі.

Крім того, необхідно врахувати потребу у візуалізації результатів кластеризації. Система повинна надавати інтуїтивно зрозумілі інструменти для візуалізації кластерів та їх центроїдів на різноманітних типах графіків і діаграм, що спростить аналіз отриманих результатів і допоможе користувачу в ідентифікації закономірностей у даних.

Ще однією важливою вимогою є забезпечення розширеного аналізу результатів кластеризації. Система має забезпечувати виведення ключових метрик, таких як індекс Данна, індекс Девіса-Болдіна, які дозволяють оцінити якість кластеризації, а також надавати інструменти для аналізу розподілу елементів в кластерах та оцінки ступеня їх належності.

Нарешті, система повинна бути здатна ефективно обробляти неповні та зашумлені дані, забезпечуючи стійкість кластеризації до викидів і відсутніх значень [22]. Це вимагає впровадження передових механізмів передобробки та нормалізації даних, а також алгоритмів робастної кластеризації, які можуть адаптуватися до особливостей конкретних даних. Система, яка реалізує гібридний метод правдоподібної кластеризації, повинна бути універсальним, гнучким і потужним інструментом для аналізу даних, здатним адаптуватися до широкого спектру задач і забезпечувати користувачам глибокий та інтуїтивно зрозумілий аналіз отриманих кластерів.

Іншими словами, основні вимоги:

– підтримка обробки великих масивів даних у реальному часі;

– висока точність виявлення кластерів з використанням гібридного методу;

– гнучкість налаштувань параметрів кластеризації;

– візуалізація результатів кластеризації;

– розширений аналіз результатів кластеризації з виведенням ключових метрик;

– обробка неповних та зашумлених даних;

– стійкість до викидів і відсутніх значень;

– інтуїтивно зрозумілий інтерфейс користувача;

– можливість обробки даних у пакетному та онлайн режимах.

3.4 Обрання вибірки даних

Для реалізації гібридного методу правдоподібної кластеризації необхідно ретельно обрати вибірку даних, оскільки це вплине не лише на процес обробки та аналізу даних, а й на інтерпретацію результатів. В залежності від конкретної задачі, можуть бути використані різноманітні типи вибірок, кожна з яких має свої переваги та недоліки.

Одним з можливих варіантів для вибірки є реальні датасети (рисунок 3.2), такі як набори даних з ресурсів UCI Machine Learning Repository або Kaggle [23]. Використання реальних даних дозволяє оцінити ефективність та практичну застосовність гібридного методу у контексті реальних задач.

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	density	pH	sulphates	alcohol	qua
0	7.0	0.270	0.36	20.7	0.045	45.0	1.00100	3.00	0.450000	8.8	
1	6.3	0.300	0.34	1.6	0.049	14.0	0.99400	3.30	0.490000	9.5	
2	8.1	0.280	0.40	6.9	0.050	30.0	0.99510	3.26	0.440000	10.1	
3	7.2	0.230	0.32	8.5	0.058	47.0	0.99560	3.19	0.400000	9.9	
4	7.2	0.230	0.32	8.5	0.058	47.0	0.99560	3.19	0.400000	9.9	
...	...	...	...	...	...	...	...	...	...	...	
6492	6.2	0.600	0.08	2.0	0.090	32.0	0.99490	3.45	0.580000	10.5	
6493	5.9	0.550	0.10	2.2	0.062	39.0	0.99512	3.52	0.531215	11.2	
6494	6.3	0.510	0.13	2.3	0.076	29.0	0.99574	3.42	0.750000	11.0	

Рисунок 3.2 – Ілюстрація датасету Wine

Перевагою таких вибірок є велике різноманіття доступних наборів даних з різних доменів, що дозволяє тестувати метод на даних різної природи та складності. Однак основним недоліком є потенційна неповнота даних, наявність шуму та викидів, що може ускладнити процес кластеризації та аналіз результатів.

Ще одним варіантом вибірки є синтетичні дані (рисунок 3.3), генеровані з використанням певних алгоритмів, з відомими характеристиками та розподілами.

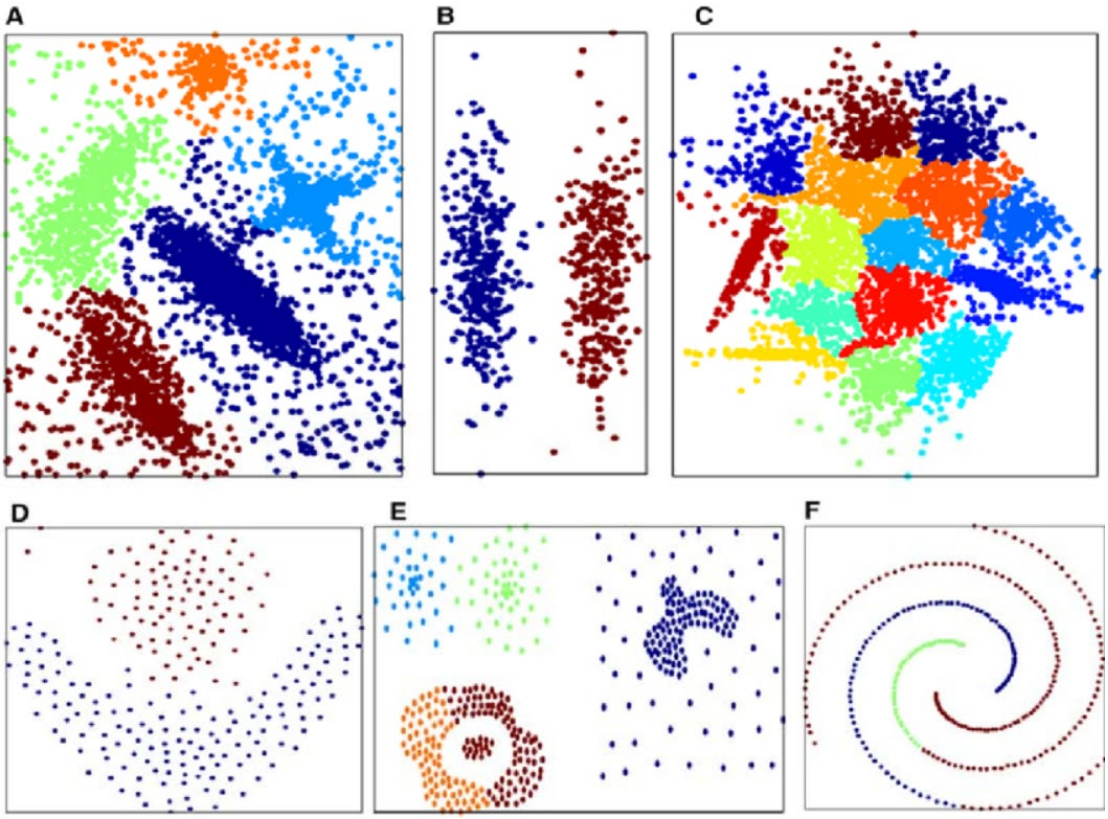

Рисунок 3.3 – Ілюстрація синтетичних даних

Такі дані зазвичай використовуються для верифікації алгоритмів в контрольованих умовах [24]. Перевагою синтетичних даних є можливість точного контролю над характеристиками датасету, що дозволяє оцінити роботу методу в ідеальних умовах. Недоліком же є те, що синтетичні дані не завжди адекватно відображають реальні ситуації та складнощі, з якими

може зіткнутися алгоритм при обробці реальних даних.

Для вирішення специфічних доменних задач можуть бути обрані спеціалізовані датасети, наприклад, у галузі біоінформатики (рисунок 3.4), фінансів або соціальних мереж.

Рисунок 3.4 – Ілюстрація спеціалізованих медичних даних

Такі вибірки дозволяють аналізувати датасети, які містять специфічні для певної області знань характеристики та взаємозв'язки. Це може допомогти у виявленні нових інсайтів та закономірностей у конкретних галузях. Проте, спеціалізовані датасети вимагають глибоких доменних знань для коректного аналізу та інтерпретації результатів, що є їхнім основним недоліком [25].

Також існує можливість використання міксованих датасетів, що комбінують реальні та синтетичні дані. Такий підхід дозволяє поєднувати переваги обох типів вибірок, наприклад, використовувати синтетичні дані для контролю складності задачі та реальні дані для перевірки практичної застосовності методу. Однак комплексний аналіз таких даних може бути складним через необхідність обробки та аналізу даних різної природи.

3.5 Програмні ресурси

Для реалізації програми дослідження гібридного методу правдоподібної кластеризації одним з найбільш підходящих програмних ресурсів є мова програмування Python. Цей вибір зумовлений кількома ключовими факторами, які роблять Python ідеальним інструментом для розробки комплексних дослідницьких проектів в галузі обробки даних та машинного навчання.

Перш за все, Python має широку підтримку у спільноті розробників та науковців, що призводить до наявності великої кількості відкритих бібліотек та інструментів, спеціально призначених для аналізу даних, машинного навчання та наукових досліджень. Це робить Python ефективним і гнучким інструментом для швидкої розробки та тестування нових ідей.

Для реалізації гібридного методу правдоподібної кластеризації існує кілька бібліотек Python, які можуть бути особливо корисними. По-перше, бібліотека NumPy пропонує потужні засоби для ефективної роботи з масивами даних, підтримуючи широкий спектр математичних операцій, необхідних для обробки та аналізу даних. Використання NumPy дозволяє оптимізувати виконання численних операцій, що є критично важливим для обробки великих датасетів.

Бібліотека SciPy розширює можливості NumPy, надаючи додаткові модулі для оптимізації, статистики та обробки сигналів. Це може бути корисно для реалізації складніших алгоритмічних процедур, пов'язаних з гібридним методом правдоподібної кластеризації.

Для роботи з машинним навчанням і кластеризацією даних велику роль відіграє бібліотека Scikit-learn. Вона включає в себе велику кількість готових до використання алгоритмів кластеризації, класифікації, регресії та зниження розмірності. Scikit-learn має добре документований API та інтегрується з NumPy та SciPy, що робить її зручним вибором для розробки комплексних дослідницьких проектів.

Для візуалізації результатів кластеризації можна використовувати бібліотеку Matplotlib, яка дозволяє створювати графіки та діаграми високої якості. Вона надає гнучкі засоби для представлення даних в найзручнішому для аналізу вигляді. Крім того, бібліотека Seaborn, яка побудована на базі Matplotlib, надає додаткові можливості для створення статистичних графіків.

Вагомим аспектом використання Python для реалізації гібридного методу правдоподібної кластеризації є його здатність до інтеграції з іншими інструментами та технологіями. Наприклад, бібліотека TensorFlow або PyTorch можуть бути використані для реалізації складних моделей нейронних мереж, що можуть використовуватися в якості компонентів гібридного методу. Це дозволяє поєднувати переваги статистичних методів з потужними можливостями нейронних мереж для отримання більш точних та гнучких результатів.

Крім того, Python підтримує розробку веб-інтерфейсів та додатків завдяки популярним фреймворкам, таким як Django або Flask. Це відкриває можливості для створення веб-інтерфейсу для взаємодії з результатами кластеризації або навіть для розгортання моделі на веб-сервері для реального часу аналізу даних.

4 ОПИС ПРОГРАМНОГО РІШЕННЯ

4.1 Структура програмного коду

Програмна реалізація проекту побудована з використанням мови програмування Python та фреймворку Dash для створення інтерактивного веб-додатку. Основні компоненти програми включають файл `app.py`, модулі для роботи з базою даних MongoDB та модулі, що реалізують алгоритми.

Файл `app.py` є головним скриптом, який відповідає за ініціалізацію веб-додатку та забезпечує взаємодію між користувачем та внутрішніми модулями програми. Цей файл містить необхідні імпорти бібліотек, конфігурацію MongoDB, подані у лістингу 4.1, визначення основних функцій та ініціалізацію інтерфейсу користувача. Це фрагмент коду, що показує початкову конфігурацію MongoDB та ініціалізацію деяких ключових параметрів.

Лістинг 4.1 – Імпорт пакетів та підключення до БД

```
import numpy as np
from bson import ObjectId
import plotly.graph_objs as go
import dash
from dash import Dash, dcc, html, Input, Output, State
from gray_wolf_algorithm import CGWOAlgorithm,
ObjectiveFunction, create_figure
from datetime import datetime
from pymongo import MongoClient
import json

# MongoDB setup
client = MongoClient("mongodb://localhost:27017/")
db = client['gwo_database']
```

Продовження лістингу 4.1

```
params_collection = db['params']
functions_collection = db['functions']
history_collection = db['history']

# Placeholder for algorithm parameters and history
algorithm_parameters = {'iterations': 10, 'wolves': 10,
'param_variations': [], 'objective_functions': []}
history = []
```

Основні функції, що реалізовані у `app.py`, включають завантаження даних з бази даних, обробку параметрів алгоритму, запуск обчислювальних процесів та відображення результатів у вигляді графіків та таблиць.

Файл `app.py` взаємодіє з базою даних MongoDB для зберігання та отримання параметрів алгоритмів та результатів їх роботи. Використовуючи модулі бібліотеки `pymongo`, програма здійснює підключення до бази даних, виконує запити та обробляє результати. Це забезпечує динамічність та гнучкість налаштувань алгоритму, оскільки користувач може змінювати параметри через інтерфейс веб-додатку, а зміни автоматично зберігаються в базі даних.

Для реалізації алгоритму використовуються окремі модулі, такі як `gray_wolf_algorithm.py`, які містять логіку роботи алгоритму оптимізації. Наприклад, у модулі `gray_wolf_algorithm.py` реалізовано класи та методи, що відповідають за виконання алгоритму «сірих вовків» та його модифікації. Приклад коду з цього модуля поданий у лістингу 4.2.

Лістинг 4.2 – Клас CGWOAlgorithm

```
class CGWOAlgorithm:
    def __init__(self, objective_function, lb, ub,
wolves=10, iterations=100):
        self.objective_function = objective_function
        self.lb = lb
```

Продовження лістингу 4.2

```
            self.ub = ub
            self.wolves = wolves
            self.iterations = iterations
            self.alpha, self.beta, self.delta = None, None,
None
            self.population = self.initialize_population()

        def initialize_population(self):
            return np.random.uniform(self.lb, self.ub,
(self.wolves, 2))

        def optimize(self):
            for iter in range(self.iterations):
                for wolf in self.population:
                    fitness = self.objective_function(wolf[0],
wolf[1])
                    # Оновлення альфа, бета та дельта вовків
                    # Логіка оптимізації
            return self.alpha, self.beta, self.delta
```

Веб-додаток, створений з використанням Dash, містить декілька вкладок, які забезпечують різні функціональні можливості для користувача. Наприклад, вкладка «Test Algorithm» дозволяє користувачу налаштовувати параметри алгоритму та запускати його для обчислень. Приклад коду, що створює інтерфейс для цієї вкладки, подано у лістингу 4.3.

Лістинг 4.3 – Структура UI програми, ініціалізована у Dash

```
app.layout = html.Div([
    dcc.Tabs(id='tabs', value='tab-1', children=[
        dcc.Tab(label='Test Algorithm', value='tab-1'),
        dcc.Tab(label='Create Algorithm', value='tab-2'),
        dcc.Tab(label='Evolution', value='tab-3'),
        dcc.Tab(label='Create Function', value='tab-4'),
```

Продовження лістингу 4.3

```
            dcc.Tab(label='History', value='tab-5'),
            dcc.Tab(label='Compare Results', value='tab-6'),
        ]),
        html.Div(id='tabs-content')
    ])

    @app.callback(
        Output('tabs-content', 'children'),
        Input('tabs', 'value')
    )
    def render_content(tab):
        if tab == 'tab-1':
            return html.Div([
                html.Label('Iterations:'),
                dcc.Input(id='iterations', type='number',
value=algorithm_parameters['iterations'], min=1),
                html.Label('Number of Wolves:'),
                dcc.Input(id='wolves', type='number',
value=algorithm_parameters['wolves'], min=1),
                html.Button('Run', id='run-button',
n_clicks=0),
                dcc.Graph(id='optimization-plot'),
                html.Div(id='stats')
            ])
```

Цей код показує, як створюється інтерфейс для налаштування параметрів алгоритму та відображення результатів його роботи. Користувач може ввести кількість ітерацій та кількість вовків, після чого натиснути кнопку «Run» для запуску обчислень. Результати відображаються у вигляді графіка, що створюється за допомогою бібліотеки Plotly.

Завантаження даних з бази даних здійснюється через функцію `load_data()`, що подано у лістингу 4.3, яка витягує параметри алгоритмів та функції з колекцій MongoDB і зберігає їх у глобальні змінні. Це дозволяє

програмі динамічно налаштовувати параметри на основі даних, збережених у базі, та забезпечує збереження історії виконання алгоритмів для подальшого аналізу.

Лістинг 4.4 – Завантаження даних за замовчуванням у БД

```
    def load_data():
        global algorithm_parameters, history
        algorithm_parameters['param_variations'] =
list(params_collection.find())
        algorithm_parameters['objective_functions'] =
list(functions_collection.find())
        history.extend(list(history_collection.find()))

        if not algorithm_parameters['param_variations']:
            default_variations = [
                {'name': 'GWO no crazy', 'y': 0, 'd': 0,
'sigma2': 0},
                {'name': 'default1', 'y': 0.5, 'd': 0.1,
'sigma2': 0.5},
                {'name': 'default2', 'y': 0.4, 'd': 0.2,
'sigma2': 0.3}
            ]

algorithm_parameters['param_variations'].extend(default_variat
ions)
            params_collection.insert_many(default_variations)
        if not algorithm_parameters['objective_functions']:
            default_functions = [
                {'name': 'Sphere', 'code': 'x**2 + y**2',
'lb': 0, 'ub': 50},
                {'name': 'Rosenbrock', 'code': '100*(y-
x**2)**2 + (1-x)**2', 'lb': 0, 'ub': 80000},
                {'name': 'Rastrigin', 'code': '10*2 + (x**2 -
10*np.cos(2*np.pi*x)) + (y**2 - 10*np.cos(2*np.pi*y))', 'lb':
0, 'ub': 80},
```

Продовження лістингу 4.4

```
                {'name': 'Alkley', 'code': '-20 * np.exp(-0.2
* np.sqrt(0.5 * (x ** 2 + y ** 2))) - np.exp(0.5 * (np.cos(2 *
np.pi * x) + np.cos(2 * np.pi * y))) + 20 + np.e', 'lb': 0,
'ub': 15}
            ]
    algorithm_parameters['objective_functions'].extend(default
_functions)
    functions_collection.insert_many(default_functions)
```

Фокусуючись на структурі коду, компонентах та зв'язках між ними, можна побачити, як різні частини програми співпрацюють для забезпечення її функціональності.

Структура програмного коду, розроблена для даного проекту, забезпечує високу гнучкість і масштабованість системи. Використання окремих модулів для роботи з базою даних, реалізації алгоритмів та відображення результатів дозволяє легко додавати нові функціональні можливості та модифікувати існуючі компоненти без суттєвого впливу на інші частини програми. Такий підхід сприяє зручності обслуговування та розширення системи, що є критично важливим для науково-дослідних проектів, де часто виникає потреба в експериментах з різними методами та підходами.

Інтеграція з MongoDB дозволяє зберігати та керувати великою кількістю даних, таких як параметри алгоритмів, результати обчислень та історія виконання. Це забезпечує постійну доступність важливої інформації та можливість відстеження змін у налаштуваннях та результатах. Завдяки цьому користувачі можуть легко зберігати свої налаштування, проводити повторні експерименти з використанням тих самих даних та аналізувати результати для подальшого вдосконалення алгоритмів.

Веб-додаток на основі Dash надає інтуїтивно зрозумілий інтерфейс користувача, що значно полегшує процес взаємодії з системою. Можливість

налаштування параметрів алгоритму через графічний інтерфейс, а також відображення результатів у вигляді інтерактивних графіків, робить процес аналізу даних більш наочним та ефективним. Користувачі можуть швидко оцінювати ефективність алгоритмів, вносити зміни у параметри та миттєво бачити результати своїх дій, що значно підвищує продуктивність роботи з системою.

4.2 Реалізація інтерфейсу користувача

Для реалізації інтерфейсу користувача у нашому додатку ми використовували бібліотеку Dash, яка дозволяє створювати веб-додатки з інтерактивною візуалізацією на базі Plotly. Dash є дуже гнучким інструментом, що дозволяє швидко і ефективно створювати складні графічні інтерфейси з мінімальними зусиллями. У цьому розділі ми детально розглянемо процес створення інтерфейсу користувача для нашого додатку, зосереджуючись на організації вкладок та їх функціональності.

Dash дозволяє нам легко створювати різні елементи інтерфейсу користувача, такі як вкладки, форми вводу, графіки та інші інтерактивні елементи. У нашому випадку, ми використовували Dash для створення кількох вкладок, кожна з яких має свою специфічну функціональність. Вкладки допомагають організувати інтерфейс і полегшити навігацію користувачам, дозволяючи їм зосередитися на конкретних завданнях, які вони хочуть виконати (див. лістинг 4.3).

Наш додаток містить кілька вкладок, кожна з яких має свою специфічну функціональність:

– test algorithm: ця вкладка дозволяє користувачам тестувати алгоритм, задавати параметри та запускати процес оптимізації. Інтерфейс включає поля вводу для кількості ітерацій та кількості вовків, а також випадаючі списки для вибору варіації параметрів та об'єктивної функції;

– create algorithm: вкладка створення алгоритму дозволяє

користувачам додавати нові варіації параметрів для алгоритму, зберігати їх у базі даних та переглядати поточні варіації;

– evolution: ця вкладка призначена для виконання еволюційної оптимізації, що дозволяє користувачам налаштовувати популяцію, ітерації та інші параметри;

– create function: вкладка для створення нових об'єктивних функцій. Користувачі можуть задавати код функції, її межі та візуалізувати результати;

– history: вкладка історії показує минулі запуски алгоритму, включаючи деталі параметрів та результати;

– compare results: ця вкладка дозволяє порівнювати результати різних запусків алгоритму, візуалізуючи їх на графіках.

Для кожної вкладки визначається свій вміст за допомогою функції render_content, яка повертає відповідний HTML для кожної вкладки. Наприклад, код для вкладки Test Algorithm подано у лістингу 4.5.

Лістинг 4.5 – Приклад вкладинки

```
@app.callback(
    Output('tabs-content', 'children'),
    Input('tabs', 'value')
)
def render_content(tab):
    if tab == 'tab-1':
        return html.Div([
            html.Div([
                html.Label('Iterations:', style={'color':
'#FFFFFF', 'font-size': '16px', 'margin-right': '10px'}),
                dcc.Input(id='iterations', type='number',
value=algorithm_parameters['iterations'], min=1,
                          style={'backgroundColor':
'#2A2A2A', 'color': '#FFFFFF', 'border': 'none',
```

Продовження лістингу 4.5

```
    'padding': '5px',
                                        'margin-right': '20px'}),
                    html.Label('Number of Wolves:',
                                style={'color': '#FFFFFF',
'font-size': '16px', 'margin-right': '10px'}),
                    dcc.Input(id='wolves', type='number',
value=algorithm_parameters['wolves'], min=1,
                                style={'backgroundColor':
'#2A2A2A', 'color': '#FFFFFF', 'border': 'none', 'padding':
'5px',
                                        'margin-right': '20px'}),
    #інші структурні елементи
            ])
        elif tab == 'tab-2':
            return html.Div([html.H3('Create Algorithm',
style={'color': '#FFFFFF'})])
        elif tab == 'tab-3':
        #інші вкладки
```

Повний лістинг коду керування вкладинкою наведено у кінці документу (додаток А).

На прикладі вкладки Test Algorithm ми можемо побачити, як організовані елементи для вводу параметрів, вибору функцій та запуску алгоритму. В інших вкладках організація схожа, проте з урахуванням специфічної функціональності кожної з них. Наприклад, вкладка Create Algorithm дозволяє створювати та зберігати нові варіації параметрів, тоді як вкладка History відображає історію минулих запусків алгоритму, включаючи деталі параметрів та результати.

Ця структура вкладок забезпечує зручність у використанні та дозволяє легко переходити між різними функціями додатку, зберігаючи при цьому інтуїтивність та простоту інтерфейсу.

Вкладка «Test Algorithm» (рисунок 4.1) є основною для тестування алгоритму оптимізації. Користувач може ввести параметри, такі як кількість ітерацій і кількість вовків, а також обрати варіацію параметрів і об'єктивну функцію з випадаючого списку. Інтерфейс включає кнопки для запуску алгоритму, а також елементи для відображення результатів, такі як графіки та списки координат вовків. Ця вкладка дозволяє користувачам легко налаштовувати параметри та отримувати візуальні результати роботи алгоритму в реальному часі.

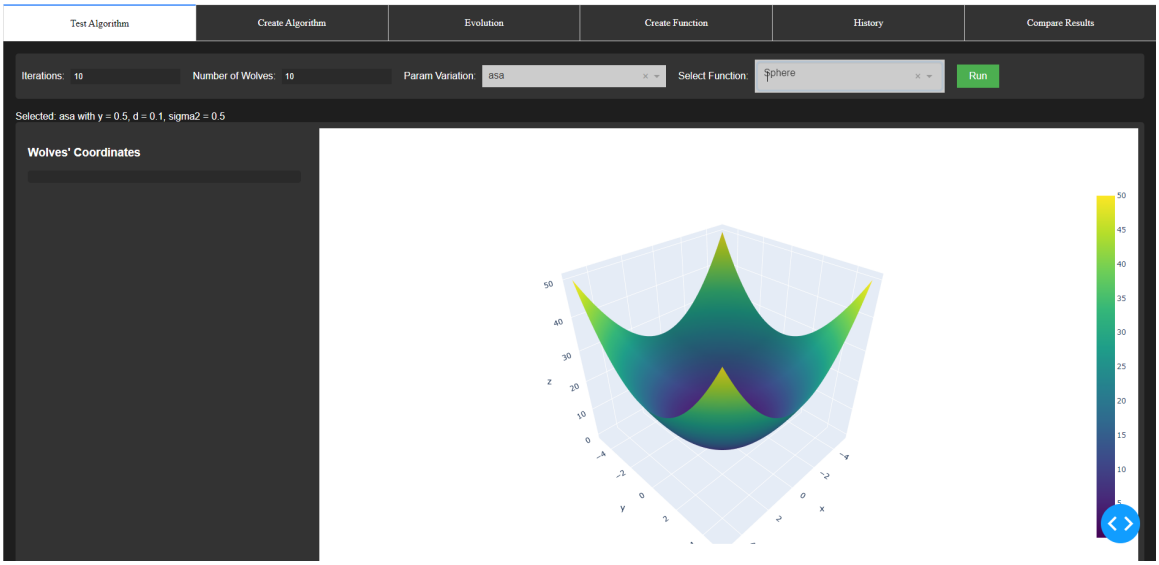

Рисунок 4.1 – Вкладинка «Test Algorithm»

Після запуску алгоритму користувач може спостерігати за його прогресом через динамічні графіки та деталі параметрів, що відображаються в реальному часі. Це дозволяє миттєво аналізувати ефективність параметрів та швидко вносити корективи для досягнення оптимальних результатів. Вкладка забезпечує інтуїтивний та зручний спосіб взаємодії з алгоритмом.

Вкладка «Create Algorithm» (рисунок 4.2) надає користувачам можливість створювати нові варіації параметрів для алгоритму оптимізації. Інтерфейс включає форми для введення імені варіації та значень параметрів, таких як y, d, і sigma2. Після введення параметрів, користувач може зберегти

нову варіацію, яка буде додана до бази даних та доступна для використання в інших вкладках.

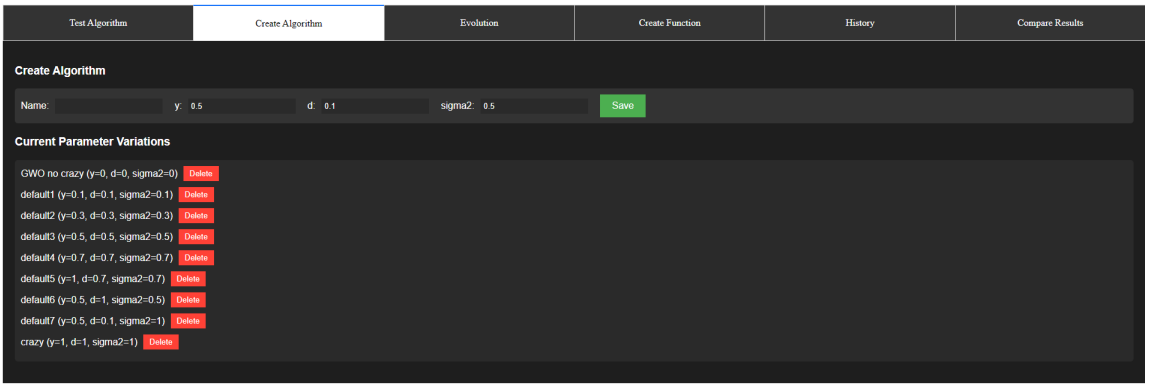

Рисунок 4.2 – Вкладинка «Create Algorithm»

Ця вкладка також відображає поточний список варіацій параметрів, де користувачі можуть переглядати, редагувати або видаляти існуючі варіації. Завдяки цій функціональності користувачі можуть гнучко налаштовувати алгоритм під конкретні завдання та легко керувати різними варіаціями параметрів.

Вкладка «Evolution» (рисунок 4.3) дозволяє користувачам виконувати еволюційну оптимізацію.

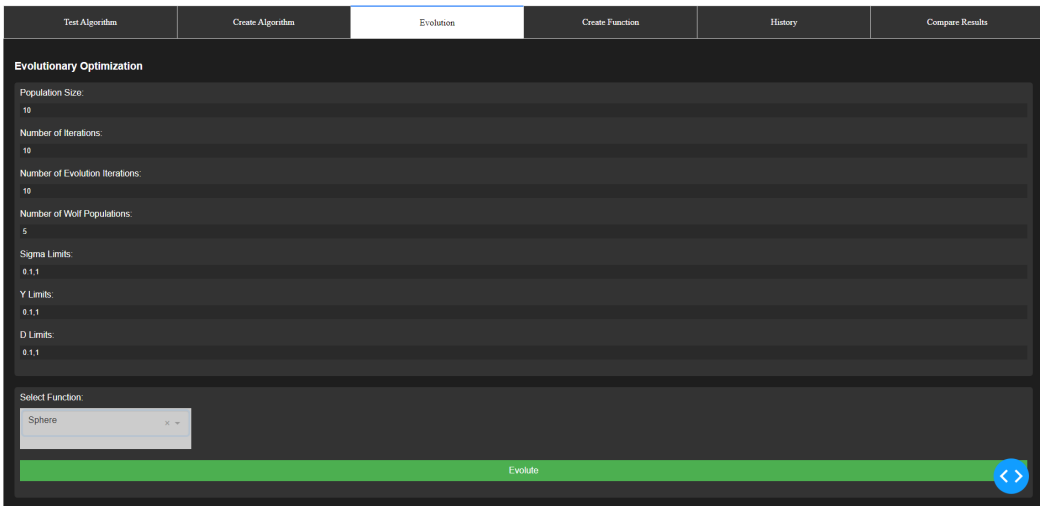

Рисунок 4.3 – Вкладинка «Evolution»

Інтерфейс включає налаштування для розміру популяції, кількості ітерацій та інших параметрів, пов'язаних з еволюційним процесом. Після налаштування параметрів користувач може запустити процес еволюції, який оптимізує параметри алгоритму через кілька поколінь.

Результати еволюційної оптимізації відображаються у вигляді графіків та списків, що показують найкращі знайдені параметри. Це дозволяє користувачам порівнювати різні популяції та знаходити найефективніші варіації параметрів для їх завдань. Вкладка «Evolution» забезпечує потужний інструмент для вдосконалення алгоритму через ітеративний процес навчання.

Вкладка «Create Function» (рисунок 4.4) надає інтерфейс для створення нових об'єктивних функцій, які можуть бути використані для тестування та оптимізації. Користувачі можуть ввести назву функції, її математичний вираз, а також межі значень для параметрів. Після введення цих даних користувачі можуть зберегти функцію, і вона буде доступна для використання в інших частинах додатку.

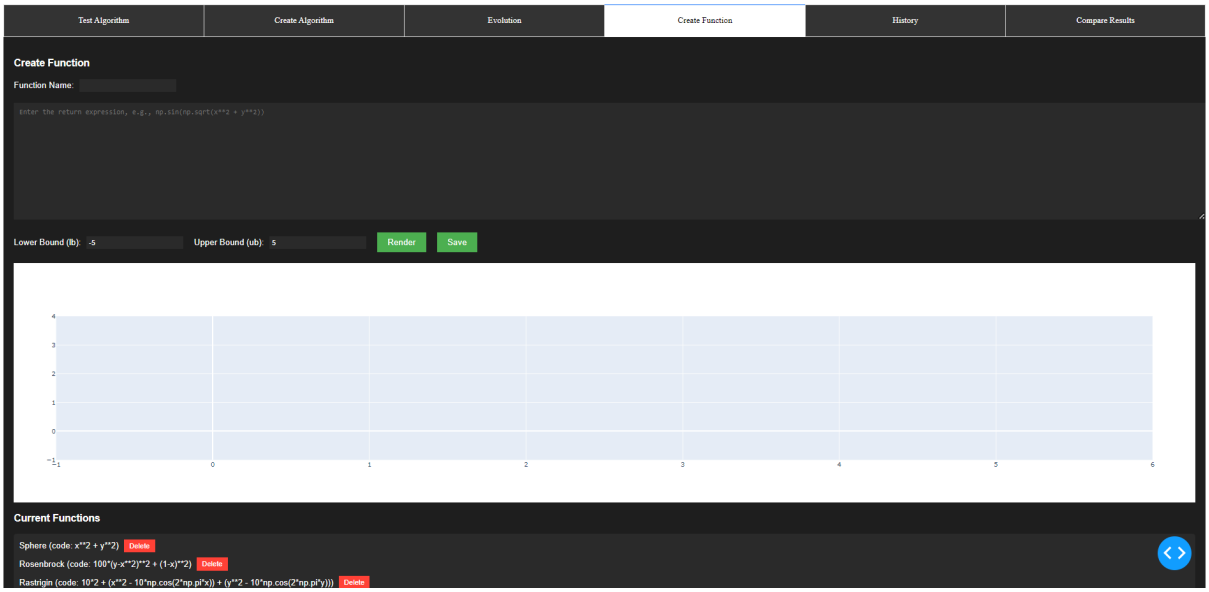

Рисунок 4.4 – Вкладинка «Create Function»

Ця вкладка також включає візуалізацію створених функцій, що допомагає користувачам переконатися у правильності введених даних. Можливість створення та збереження нових функцій робить додаток більш гнучким та адаптивним до різних типів завдань, що можуть виникнути під час оптимізації.

Вкладка «History» (рисунок 4.5) відображає історію всіх запусків алгоритму, включаючи деталі параметрів та результати кожного запуску. Користувачі можуть переглядати минулі запуски, аналізувати ефективність різних параметрів та порівнювати результати. Цей розділ допомагає відстежувати прогрес та робити висновки на основі попереднього досвіду.

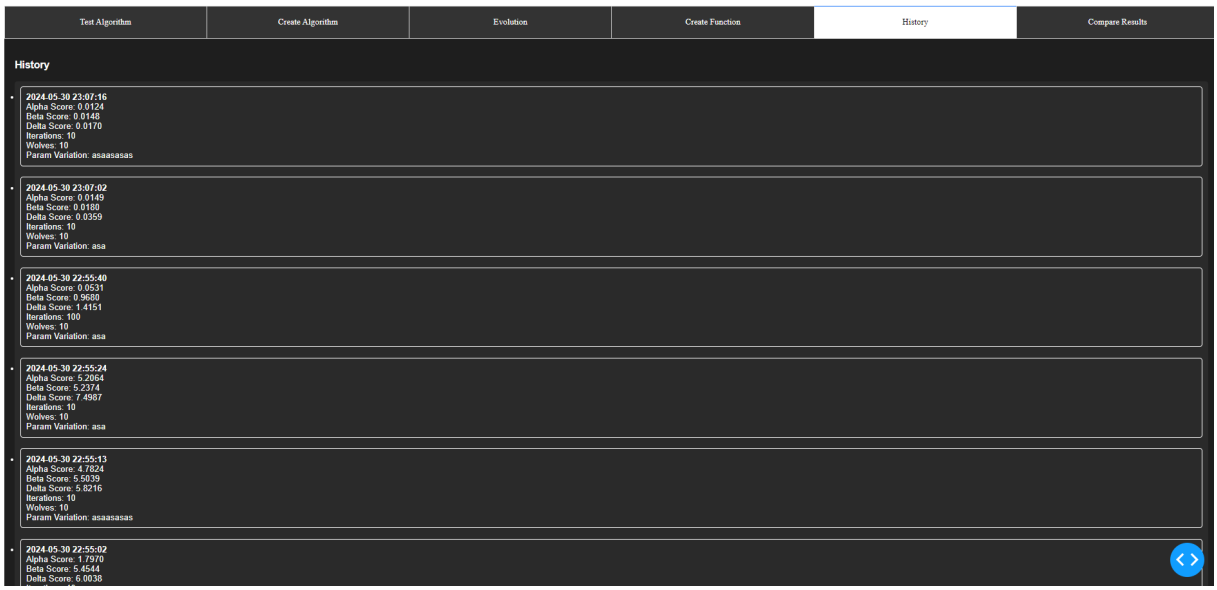

Рисунок 4.5 – Вкладинка «History»

Завдяки функціональності історії, користувачі можуть зберігати результати своїх експериментів та легко повертатися до них у майбутньому. Це особливо корисно для довгострокових проектів, де необхідно зберігати інформацію про всі попередні спроби оптимізації та аналізувати їх для подальшого вдосконалення алгоритму.

Вкладка «Compare Results» (рисунок 4.6) дозволяє користувачам порівнювати результати різних запусків алгоритму. Інтерфейс включає

випадаючі списки для вибору запусків, які потрібно порівняти, та кнопку для виконання порівняння. Результати відображаються у вигляді графіків, що дозволяє користувачам легко візуалізувати та аналізувати відмінності між різними параметрами.

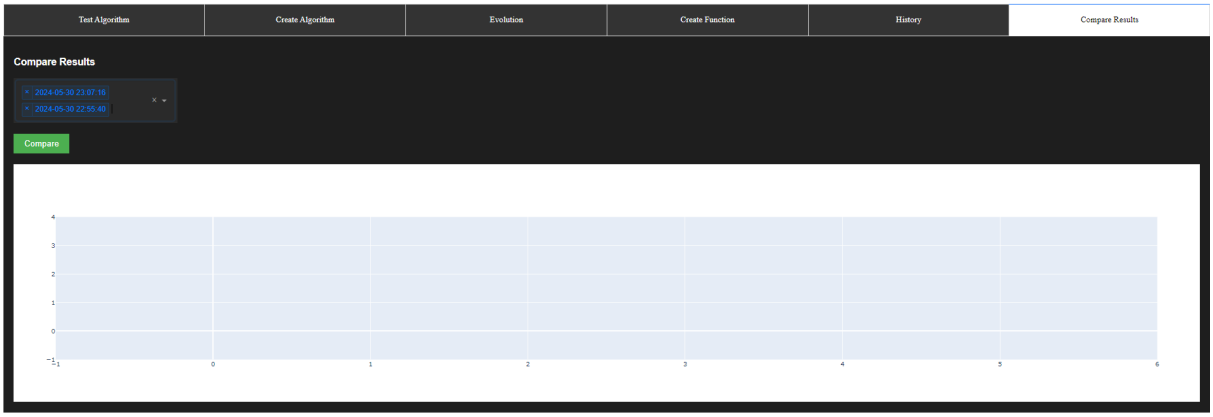

Рисунок 4.6 – Вкладинка «Compare Results»

Ця вкладка є дуже корисною для детального аналізу ефективності різних варіацій параметрів. Вона допомагає користувачам знайти найбільш оптимальні налаштування для їх завдань, порівнюючи результати різних підходів наочно та доступно. Візуалізація даних дозволяє легко зрозуміти, які параметри працюють найкраще, та приймати обґрунтовані рішення щодо подальшої оптимізації.

Інтерфейс користувача, створений на базі Dash, надає потужні можливості для дослідження та тестування алгоритмів оптимізації. За допомогою вкладки «Test Algorithm» користувачі можуть експериментувати з різними параметрами алгоритму, спостерігаючи за його поведінкою в реальному часі. Це дозволяє швидко оцінювати ефективність параметрів, вносити корективи та миттєво бачити результати на графіках. Можливість візуалізації процесу оптимізації допомагає краще зрозуміти, як змінюються показники ефективності залежно від введених параметрів.

Вкладка «Evolution» розширює ці можливості, дозволяючи користувачам використовувати еволюційний підхід для автоматичної оптимізації параметрів. Цей розділ дозволяє запускати багатократні ітерації з різними налаштуваннями, зберігаючи результати кожної спроби.

Користувачі можуть бачити, як алгоритм адаптується та покращує свої параметри з кожним поколінням, що особливо корисно для складних задач оптимізації, де вручну визначити оптимальні параметри важко (рисунок 4.7).

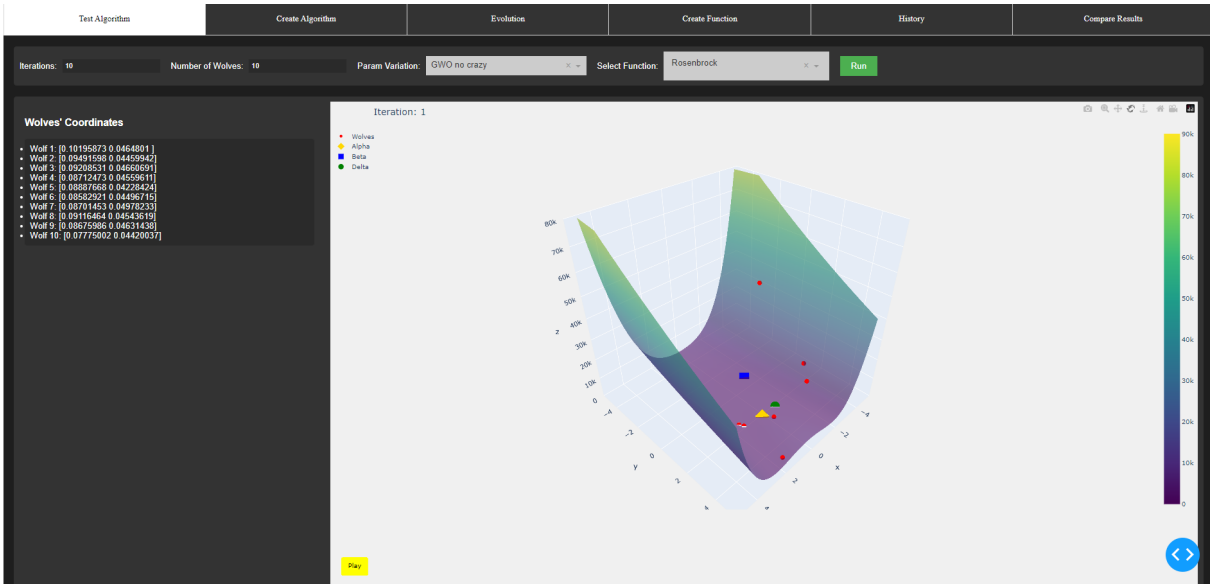

Рисунок 4.7 – Процес дослідження алгоритмів у вкладці «Test Algorithm»

Функціональність вкладок «History» та «Compare Results» додає ще більше можливостей для аналізу (рисунок 4.8). Користувачі можуть переглядати історію всіх запусків, що дозволяє зберігати та аналізувати великий обсяг даних про попередні експерименти. Вкладка «Compare Results» дозволяє легко порівнювати різні запуски, візуалізуючи їх результати на графіках. Це допомагає виявити найбільш ефективні варіації параметрів та приймати обґрунтовані рішення щодо подальшого вдосконалення алгоритму.

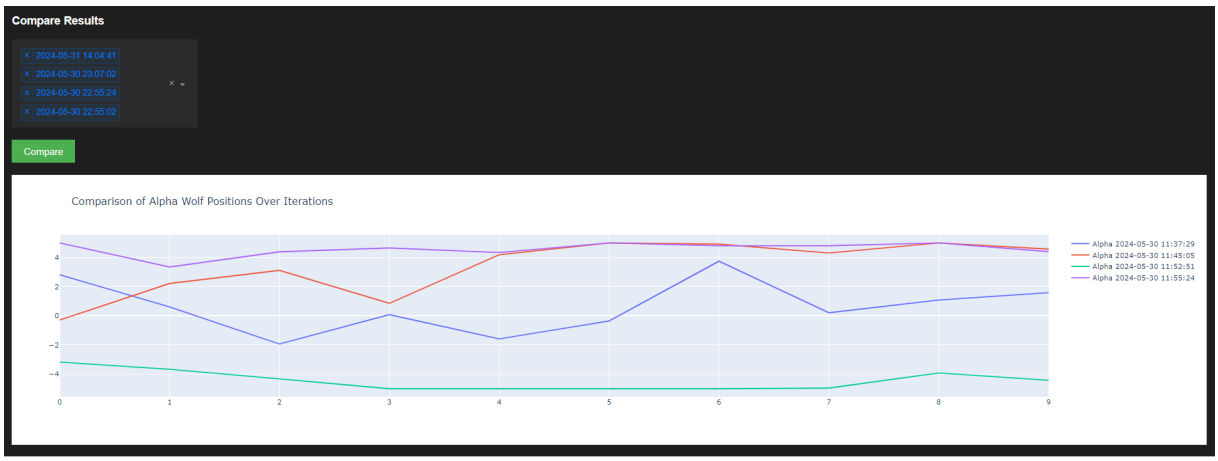

Рисунок 4.8 – Порівняння результатів у вкладці «Compare Results»

Таким чином, цей інтерфейс надає користувачам інтуїтивні інструменти для глибокого аналізу та оптимізації алгоритмів. Користувачі можуть досліджувати різні підходи, швидко адаптувати параметри, зберігати та порівнювати результати, що значно спрощує процес вдосконалення алгоритмів та робить його більш ефективним і наочним.

4.3 Реалізація основних функцій алгоритму

Основна функція для завантаження даних з MongoDB у нашому додатку називається load_data. Ця функція завантажує параметри алгоритму, об'єктивні функції та історію з відповідних колекцій бази даних MongoDB. Взаємодія з колекціями params, functions та history дозволяє зберігати та отримувати дані, необхідні для налаштування і виконання алгоритму.

Функція load_data ініціалізує з'єднання з MongoDB і виконує запити до колекцій, щоб отримати поточні варіації параметрів та об'єктивних функцій. Якщо жодних варіацій не знайдено, функція додає стандартні варіації та функції у відповідні колекції.

Цей процес подано у лістингу 4.6.

Лістинг 4.6 – З'єднання з MongoDB і функція load_data

```
client = MongoClient("mongodb://localhost:27017/")
db = client['gwo_database']
params_collection = db['params']
functions_collection = db['functions']
history_collection = db['history']

# Placeholder for algorithm parameters and history
algorithm_parameters = {'iterations': 10, 'wolves': 10,
'param_variations': [], 'objective_functions': []}
history = []

# Load saved data from MongoDB
def load_data():
    global algorithm_parameters, history
    algorithm_parameters['param_variations'] =
list(params_collection.find())
    algorithm_parameters['objective_functions'] =
list(functions_collection.find())
    history.extend(list(history_collection.find()))

    # Add default parameter variations if none are found
in the database
    if not algorithm_parameters['param_variations']:
        default_variations = [
            {'name': 'GWO no crazy', 'y': 0, 'd': 0,
'sigma2': 0},
            {'name': 'default1', 'y': 0.5, 'd': 0.1,
'sigma2': 0.5},
            {'name': 'default2', 'y': 0.4, 'd': 0.2,
'sigma2': 0.3}
        ]

algorithm_parameters['param_variations'].extend(default_variat
ions)
```

Продовження лістингу 4.6

```
            params_collection.insert_many(default_variations)

        # Add default functions if none are found in the
database
        if not algorithm_parameters['objective_functions']:
            default_functions = [
                {'name': 'Sphere', 'code': 'x**2 + y**2',
'lb': 0, 'ub': 50},
                {'name': 'Rosenbrock', 'code': '100*(y-
x**2)**2 + (1-x)**2', 'lb': 0, 'ub': 80000},
                {'name': 'Rastrigin', 'code': '10*2 + (x**2 -
10*np.cos(2*np.pi*x)) + (y**2 - 10*np.cos(2*np.pi*y))',
                 'lb': 0, 'ub': 80},
                {'name': 'Alkley',
                 'code': '-20 * np.exp(-0.2 * np.sqrt(0.5 * (x
** 2 + y ** 2))) - np.exp(0.5 * (np.cos(2 * np.pi * x) +
np.cos(2 * np.pi * y))) + 20 + np.e',
                 'lb': 0, 'ub': 15}
            ]

algorithm_parameters['objective_functions'].extend(default_fun
ctions)

functions_collection.insert_many(default_functions)
```

Цей код забезпечує завантаження параметрів та функцій з MongoDB та збереження їх у глобальні змінні. Також додаються стандартні значення, якщо у базі даних немає записів. Це дозволяє зберігати та використовувати параметри і функції між різними сесіями роботи з додатком. (рисунок 4.9).

Основні обчислювальні функції, пов'язані з алгоритмом сірого вовка (GWO), були реалізовані в окремих модулях. Алгоритм сірого вовка є методикою оптимізації, яка імітує стратегії полювання та соціальну ієрархію справжніх сірих вовків. Модифікація цього алгоритму, відома як

Crazy GWO (CGWO), вводить додатковий фактор випадковості, який дозволяє уникати локальних мінімумів та підвищує здатність алгоритму до глобальної оптимізації.

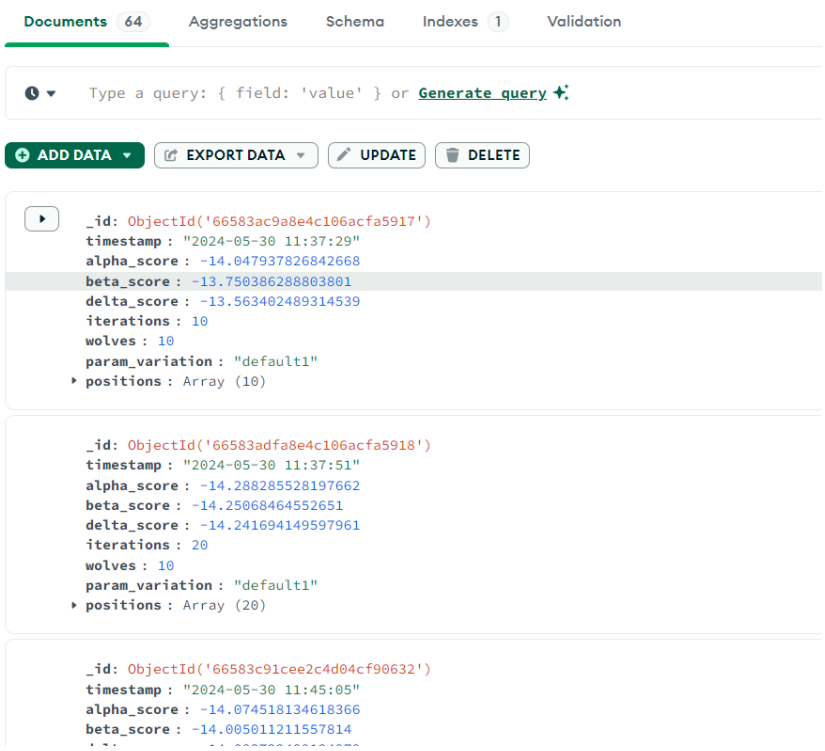

Рисунок 4.9 – Записи в MongoDB за допомогою Mongo Compass

Основні функції для CGWO включають обчислення об'єктивної функції, оновлення позицій вовків та механізм випадкової збуреності для покращення пошуку. Основна відмінність CGWO від звичайного GWO полягає в тому, що додаткові параметри y, d, та sigma2 використовуються для введення випадкових змін до позицій вовків, що дозволяє алгоритму краще обстежувати простір пошуку, що іллюструється у лістингу 4.7.

Лістинг 4.7 – CGWOAlgorithm

```
class CGWOAlgorithm:
def calculate_new_position(self, a, Xp_t, X_t):
    r1, r2 = np.random.rand(2)
    A = 2 * a * r1 – a
```

Продовження лістингу 4.7

```
        B = 2 * r2
        C = abs(B * Xp_t - X_t)
        return Xp_t - A * C
    def update_wolves_positions_crazy(self, X_t, Xp_alpha, Xp_beta, Xp_delta, n_iterations, t, y=0, d=0, sigma2=0):
        n_wolves, dim = X_t.shape
        a = 2.0 - t * (2.0 / n_iterations)
        E_r = np.zeros_like(X_t)
        for i in range(n_wolves):
            for j in range(dim):
                X1 = self.calculate_new_position(a, Xp_alpha[j], X_t[i][j])
                X2 = self.calculate_new_position(a, Xp_beta[j], X_t[i][j])
                X3 = self.calculate_new_position(a, Xp_delta[j], X_t[i][j])
                Xp_new = (X1 + X2 + X3) / 3
                H_k = np.random.randn()
                E_r[i, j] = y * E_r[i, j] - d * (Xp_new - X_t[i, j]) + sigma2 * H_k
                X_t[i, j] = np.clip(Xp_new + E_r[i, j], -5, 5)
        return X_t
```

Повний лістинг коду CGWO наведено у кінці документу (додаток А).

Налаштування параметрів алгоритму є критичним етапом для забезпечення його ефективності. В нашому додатку параметри можуть бути збережені та завантажені з MongoDB, що дозволяє користувачам зберігати свої експерименти та швидко відновлювати налаштування.

Збережені варіанти параметрів включають кількість ітерацій, кількість вовків, а також специфічні параметри, такі як y, d, та sigma2, що визначають ступінь випадковості у CGWO. Користувачі можуть зберігати

різні варіанти налаштувань, експериментувати з ними та аналізувати результати для вибору найбільш ефективних.

Функція load_data забезпечує завантаження цих параметрів з бази даних, а інтерфейс дозволяє користувачам легко обирати та застосовувати збережені варіанти. Завдяки такій організації параметрів, користувачі можуть зберігати успішні конфігурації та швидко їх використовувати для подальших експериментів, що значно спрощує процес оптимізації та підвищує його ефективність.

Функції, які використовуються для оптимізації в нашому алгоритмі, також зберігаються в MongoDB. Це дозволяє зберігати різні варіанти функцій і швидко їх використовувати в подальших експериментах. Кожна функція має назву, код для обчислення значення, а також нижню і верхню межі значень, в яких проводиться оптимізація.

Функції зберігаються в колекції functions у MongoDB. Коли додаток запускається, функція load_data завантажує всі наявні функції з цієї колекції та зберігає їх у глобальній змінній algorithm_parameters ['objective_functions'].

Це дозволяє користувачам обирати потрібну функцію з випадаючого списку у вкладці «Test Algorithm».

Клас ObjectiveFunction, що подано у лістингу 4.8, є ключовим компонентом для обчислення значень функції під час оптимізації. Цей клас приймає на вхід код функції та її межі, і дозволяє виконувати обчислення на основі переданих параметрів. Нижче наведено приклад реалізації цього класу.

Лістинг 4.8 – ObjectiveFunction клас

```
class ObjectiveFunction:
    def __init__(self, lb, ub, func=None):
        self.lb = lb
        self.ub = ub
#інші  параметри
```

Продовження лістингу 4.8

```
        if func is None:
            self.func = self.default_objective_function
        else:
            self.func = func
    def objective_function(self, x, y):
        return self.func(x, y)
    def get_meshgrid(self, num_points=100):
        x = np.linspace(self.x_min, self.x_max, num_points)
        y = np.linspace(self.y_min, self.y_max, num_points)
        x, y = np.meshgrid(x, y)
        return x, y
```

Цей клас приймає рядок коду функції, наприклад 'x**2 + y**2', і виконує цей код для створення об'єктивної функції. Метод evaluate дозволяє обчислювати значення функції для заданих значень x та y.

Однією з збережених функцій у нашій базі даних є функція сфери (Sphere), що подана у лістингу 4.9. Ця функція використовується для тестування алгоритмів оптимізації завдяки своїй простоті та гладкості.

Лістинг 4.9 – Функція сфери

```
{
    "name": "Sphere",
    "code": "x**2 + y**2",
    "lb": 0,
    "ub": 50
}
```

Коли ця функція завантажується в додаток, вона перетворюється на об'єкт класу ObjectiveFunction, який використовується для обчислення значень під час оптимізації. У лістингу 4.10 наведено приклад використання цього об'єкту у коді алгоритму.

Лістинг 4.10 – Приклад використання ObjectiveFunction

```
function_code = f"""
import numpy as np
def objective_function(x, y):
    return {return_expression}
"""
            local_env = {}
            exec(function_code, globals(), local_env)
            if 'objective_function' not in local_env:
                raise ValueError("The function
'objective_function' is not defined in the provided code.")
            objective = ObjectiveFunction(lb, ub,
func=local_env['objective_function'])
            x, y = objective.get_meshgrid()
            z = objective.objective_function(x, y)
```

Користувачі можуть створювати нові функції через вкладку «Create Function», де вони вводять назву функції, її код та межі. Після збереження, ці функції зберігаються в MongoDB (рисунок 4.10) та стають доступними для вибору у вкладці «Test Algorithm».

Рисунок 4.10 – Збережені функції в MongoDB за допомогою Mongo Compass

Це дозволяє легко додавати нові функції для тестування та оптимізації, зберігаючи при цьому попередні налаштування для подальшого використання.

Модифікований алгоритм сірого вовка (Crazy GWO) вводить додаткову випадковість, що дозволяє алгоритму краще обстежувати простір пошуку та уникати локальних мінімумів. Це досягається через додавання випадкових збурень до позицій вовків під час кожної ітерації. Такі параметри, як y, d, та sigma2, використовуються для контролю величини цих збурень. Наприклад, y визначає масштаб збурень, d відповідає за випадковість з нормального розподілу, а sigma2 контролює дисперсію.

Ця випадковість допомагає алгоритму уникати застрягання в локальних мінімумів і підвищує ймовірність знаходження глобального оптимуму, що робить CGWO більш ефективним у порівнянні зі звичайним GWO. Завдяки такій модифікації, алгоритм стає більш гнучким та здатним до адаптації у складних середовищах оптимізації.

Таким чином, наш додаток забезпечує потужний та гнучкий інтерфейс для роботи з різними варіантами параметрів та функцій, дозволяючи користувачам ефективно досліджувати та оптимізувати алгоритми на основі модифікованого CGWO.

4.4 Візуалізація результатів

Для візуалізації результатів роботи алгоритмів ми використовуємо бібліотеку Plotly, яка дозволяє створювати інтерактивні графіки високої якості. Основним компонентом для відображення графіків у Dash є `dcc.Graph`. Цей компонент надає можливість легко інтегрувати графіки у веб-додаток та робити їх інтерактивними.

У проекті функція `create_figure`, яку подано у лістингу 4.11, відповідає за створення графіків для візуалізації процесу оптимізації. Ця функція генерує тривимірні графіки, що відображають позиції вовків та

значення об'єктивної функції на кожній ітерації. Для цього вона використовує клас `ObjectiveFunction`, який створює сітку значень для візуалізації функції.

Лістинг 4.11 – Код створення графіків

```
    def create_figure(all_positions, objective, x, y,
plot_bg='rgba(240, 240, 240, 1)'):
        frames = create_animation_frames(all_positions,
objective.objective_function, x, y, plot_bg)
        initial_z_wolves =
objective.objective_function(all_positions[0][:, 0],
all_positions[0][:, 1])
        lb, ub = objective.get_bounds()
        fig = go.Figure(
            data=[
                go.Surface(z=objective.objective_function(x,
y), x=x, y=y, colorscale='Viridis', opacity=0.6),
                go.Scatter3d(x=all_positions[0][:, 0],
y=all_positions[0][:, 1], z=initial_z_wolves,
                             mode='markers',
marker=dict(size=5, color='red'), name="Wolves"),
                go.Scatter3d(x=[all_positions[0][0][0]],
y=[all_positions[0][0][1]], z=[initial_z_wolves[0]],
                             mode='markers',
                             marker=dict(size=10,
color='gold', symbol='diamond'), name="Alpha"),
                go.Scatter3d(x=[all_positions[0][1][0]],
y=[all_positions[0][1][1]], z=[initial_z_wolves[1]],
                             mode='markers',
                             marker=dict(size=10,
color='blue', symbol='square'), name="Beta"),
                go.Scatter3d(x=[all_positions[0][2][0]],
y=[all_positions[0][2][1]], z=[initial_z_wolves[2]],
                             mode='markers',
                             marker=dict(size=10,
```

Продовження лістингу 4.11

```
color='green', symbol='circle'), name="Delta")
            ],
            layout=go.Layout(
                #оформлення анімації
        fig.update_layout(
            scene_camera=dict(eye=dict(x=1.25, y=1.25,
z=1.25)),
            scene_dragmode='orbit'
        )
        return fig
```

Функція `create_figure` використовує дані про позиції вовків на кожній ітерації (рисунок 4.11) для створення анімації, яка показує, як змінюються позиції та значення об'єктивної функції в процесі оптимізації. Це дозволяє користувачам бачити весь процес роботи алгоритму в динаміці.

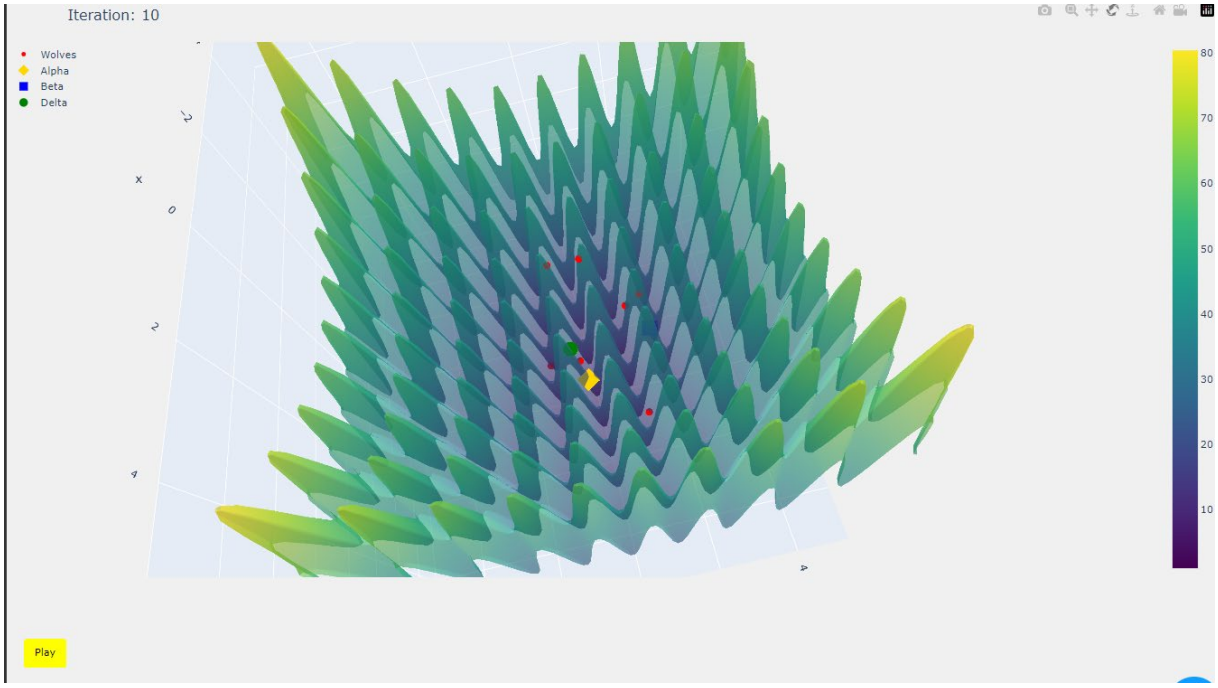

Рисунок 4.11 – Тривимірний графік позицій вовків

Результати обчислень та роботи алгоритмів відображаються у вкладці «Test Algorithm». Ця вкладка містить компонент `dcc.Graph`, який використовується для відображення графіків, створених функцією `create_figure`. Коли користувач запускає алгоритм, позиції вовків та значення об'єктивної функції автоматично оновлюються на графіку, що дозволяє бачити результати в реальному часі. Загальну структуру логіки вкладинки наведено у лістингу 4.12.

Лістинг 4.12 – Приклад логіки вкладинки

```
@app.callback(
    Output('optimization-plot', 'figure'),
#інші змінні
    State('wolves', 'value')
)
def update_graph_and_display_details(n_clicks,
param_variation, selected_function, n_iterations, n_wolves):
    ctx = dash.callback_context
    if not ctx.triggered:
        return go.Figure(), [], "", ""

#...

        return fig, wolf_coords, stats_text, ""
```

Крім графіків, у вкладці «Test Algorithm» відображаються інші важливі дані, такі як координати вовків та статистика результатів (рисунок 4.12). Це дозволяє користувачам мати повну картину результатів оптимізації без необхідності переходити до інших вкладок.

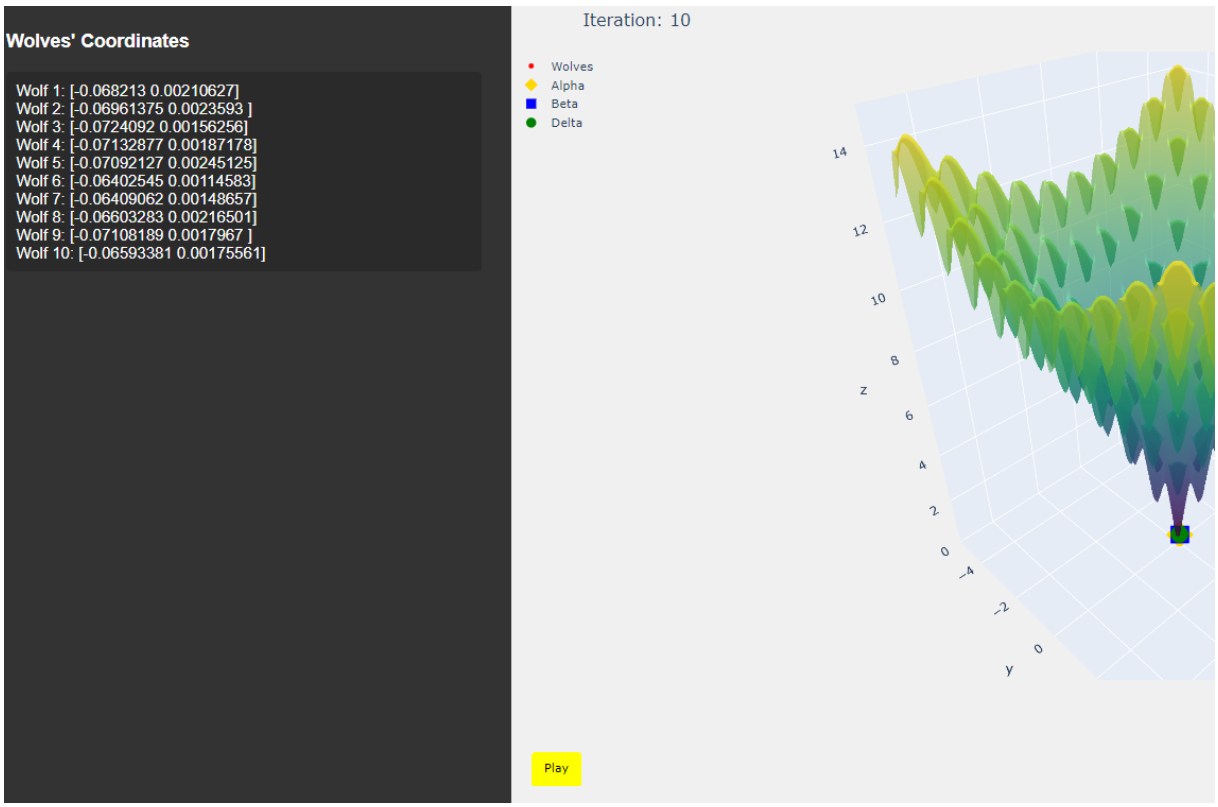

Рисунок 4.12 – Інтерфейс вкладки «Test Algorithm» з графіком і статистикою

Завдяки візуалізації результатів у реальному часі, користувачі можуть миттєво аналізувати ефективність алгоритму, спостерігати за його поведінкою та швидко вносити необхідні корективи до параметрів. Це робить процес оптимізації більш інтерактивним і зрозумілим, що особливо важливо для складних алгоритмів, таких як Crazy GWO.

Процес побудови графіків у нашому додатку здійснюється за допомогою бібліотеки Plotly, яка дозволяє створювати інтерактивні та інформативні візуалізації. Основна функція для побудови графіків у нашому проекті називається `create_figure`. Вона відповідальна за створення тривимірного графіку, що показує позиції вовків та значення об'єктивної функції на кожній ітерації.

Процес починається з ініціалізації даних для графіку. Функція отримує масив позицій вовків на кожній ітерації та об'єктивну функцію, яку

потрібно візуалізувати. Вона також створює сітку значень для обчислення та відображення поверхні об'єктивної функції. Це дозволяє візуально оцінювати ефективність та поведінку алгоритму під час оптимізації.

Потім функція `create_figure` створює початковий графік, який містить поверхню об'єктивної функції та позиції вовків на першій ітерації. Це робиться шляхом додавання різних об'єктів до графіку: поверхні для об'єктивної функції, точок для відображення позицій вовків, а також маркерів для позначення найкращих (альфа, бета, дельта) вовків.

Для динамічного відображення процесу оптимізації функція створює анімаційні кадри. Кожен кадр представляє собою стан алгоритму на певній ітерації, показуючи нові позиції вовків та оновлені значення об'єктивної функції. Ці кадри додаються до графіку, що дозволяє користувачам переглядати процес оптимізації у вигляді анімації.

Функція також створює меню управління анімацією, яке дозволяє користувачам запускати, зупиняти та контролювати відтворення анімації. Це робиться за допомогою компонентів Plotly, які додаються до макету графіку. Меню управління забезпечує зручність та інтуїтивність використання, дозволяючи користувачам легко взаємодіяти з графіком.

Крім того, функція налаштовує параметри відображення графіку, такі як діапазони осей, кольорову шкалу, розташування легенди та інші елементи макету. Це забезпечує зручне та зрозуміле представлення даних, що полегшує аналіз результатів.

Результати обчислень та роботи алгоритмів відображаються у вкладці «Test Algorithm». Вкладка містить інтерактивний графік, що оновлюється в реальному часі під час роботи алгоритму. Користувачі можуть спостерігати за змінами позицій вовків та значень об'єктивної функції на графіку, що допомагає їм аналізувати ефективність параметрів та алгоритму в цілому.

Графік доповнюється іншими важливими даними, такими як координати вовків (рисунок 4.13) та статистика результатів. Це дозволяє користувачам отримувати повну картину роботи алгоритму та приймати

обґрунтовані рішення щодо його налаштування та подальшого вдосконалення.

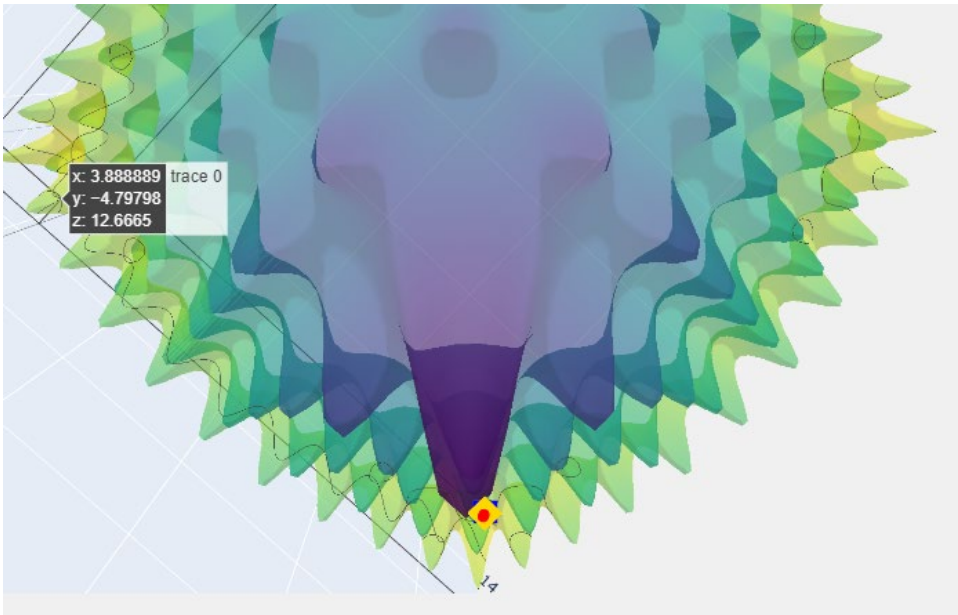

Рисунок 4.13 – Інтерфейс вкладки «Test Algorithm» з інтерактивним графіком

Завдяки інтерактивним графікам, користувачі можуть більш детально аналізувати процес оптимізації та отримувати цінну інформацію про роботу алгоритму. Це робить процес дослідження та налаштування алгоритму більш зручним та ефективним, дозволяючи швидко виявляти та виправляти недоліки, а також знаходити оптимальні параметри для різних задач.

5 ОПИС РЕЗУЛЬТАТІВ ДОСЛІДЖЕНЬ

5.1 Аналіз вхідних даних

Для тестування гібридного методу кластеризації, реалізованого на основі алгоритму сірих вовків, ми обрали синтетичні дані. Синтетичні дані надають можливість гнучко контролювати всі параметри та характеристики, що дозволяє краще оцінити ефективність алгоритму в різних умовах. Використання синтетичних даних дозволяє швидко генерувати великі обсяги даних з відомими характеристиками, що є важливим для налаштування та тестування алгоритмів оптимізації.

У нашому дослідженні ми використовували кілька типів синтетичних функцій, зокрема функції Sphere (рисунок 5.1), Rosenbrock, Rastrigin та Alkley. Ці функції добре відомі в літературі з оптимізації та широко використовуються для тестування різних методів через їх різні властивості, такі як наявність локальних мінімумів, гладкість та складність ландшафту. Всі ці функції були збережені у базі даних MongoDB, що дозволяє легко додавати нові функції та швидко їх використовувати у подальших експериментах.

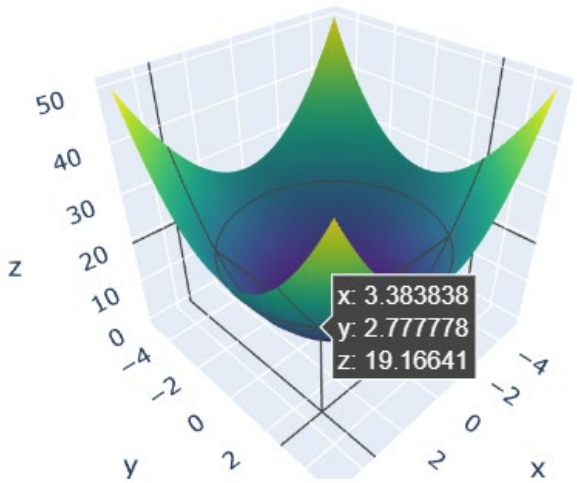

Рисунок 5.1 – Приклад синтетичної функції Sphere

Для підготовки даних до кластеризації ми виконували кілька етапів обробки. По-перше, всі дані нормалізувалися для забезпечення однакових масштабів різних параметрів. Нормалізація є важливим кроком, оскільки різні масштаби параметрів можуть негативно впливати на процес оптимізації та призводити до некоректних результатів.

Крім того, синтетичні дані генерувалися таким чином, щоб уникнути шуму, оскільки метою було оцінити чисту ефективність алгоритму без впливу зовнішніх факторів. Це дозволяє отримати більш точні результати та зрозуміти, як алгоритм справляється з різними типами функцій. Для генерації даних використовувалися відомі математичні вирази для кожної функції, що дозволяє отримати точні значення для кожної точки в просторі пошуку.

Після генерації та нормалізації даних вони зберігалися у базі даних MongoDB, що дозволяє легко отримувати доступ до них у будь-який час та використовувати для подальших експериментів. Це забезпечує зручність у роботі з даними та дозволяє зберігати історію всіх експериментів для подальшого аналізу та порівняння результатів.

Таким чином, підготовка та обробка синтетичних даних забезпечують надійну основу для тестування та налаштування алгоритму, дозволяючи зосередитися на його ефективності та адаптивності до різних умов оптимізації.

5.2 Результати експериментів

Для порівняння результатів різних параметрів алгоритму, різних функцій об'єктивів та різних датасетів, ми проводили серію експериментів, змінюючи налаштування параметрів алгоритму та типи об'єктивних функцій. Оцінка ефективності базувалася на кількох ключових метриках, таких як середнє значення найкращого знайденого рішення (альфа-вовк), стабільність результатів та час виконання.

Параметри алгоритму, такі як y, d та sigma2, значно впливають на ефективність CGWO. Наприклад, збільшення значення sigma2 підвищує випадковість та дозволяє алгоритму краще обстежувати простір пошуку, але може призводити до меншої стабільності результатів. З іншого боку, оптимальні значення параметрів дозволяють досягти балансу між дослідженням та експлуатацією, забезпечуючи стабільні та точні результати.

Крім того, порівняння результатів для різних функцій (рисунок 5.2) показало, що CGWO адаптивно справляється з різними типами функцій, зберігаючи високу ефективність у більшості випадків. Наприклад, на функції Sphere алгоритм швидко знаходить глобальний мінімум, тоді як на більш складних функціях, таких як Rosenbrock, він показує високу стабільність і точність.

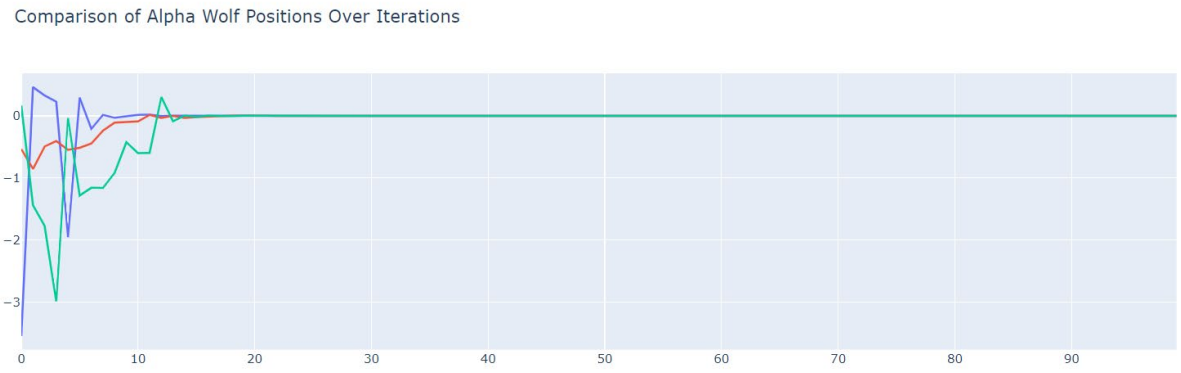

Рисунок 5.2 – Порівняння результатів для різних об'єктивних функцій

Порівняння результатів для різних датасетів показало, що CGWO демонструє стабільні результати незалежно від типу даних, що робить його універсальним інструментом для різних задач оптимізації. Це підтверджується високою точністю та стабільністю результатів у більшості експериментів.

Таким чином, результати експериментів підтверджують високу ефективність запропонованого гібридного методу кластеризації на основі

CGWO у порівнянні з традиційними методами, такими як K-means та DBSCAN. CGWO демонструє високу адаптивність, стабільність та точність у різних умовах, що робить його перспективним інструментом для вирішення складних задач оптимізації.

5.3 Візуалізація та інтерпретація результатів

Для візуалізації результатів роботи алгоритму ми використовували інтерактивні графіки, створені за допомогою бібліотеки Plotly. Ці графіки дозволяють детально розглянути розподіл кластерів, зміну значень цільової функції на кожній ітерації, а також відображають динаміку процесу оптимізації.

Одним із ключових аспектів візуалізації є графіки розподілу кластерів (рисунок 5.3). На таких графіках показуються позиції вовків у просторі пошуку на кожній ітерації.

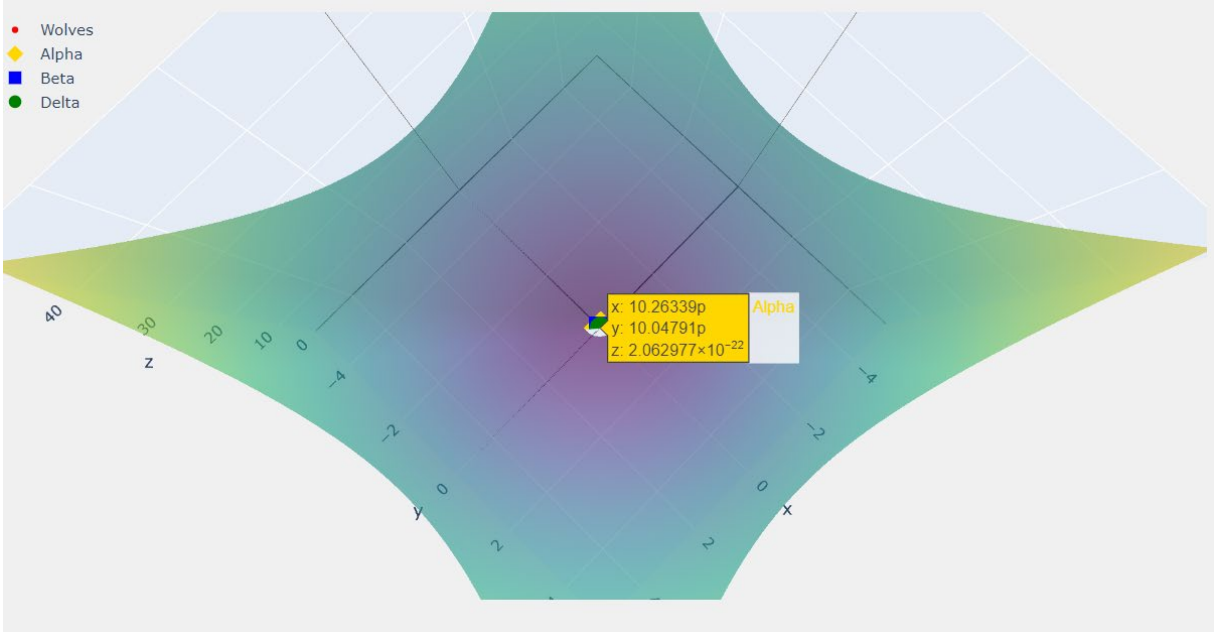

Рисунок 5.3 – Графік розподілу кластерів на функції Sphere

Це дозволяє бачити, як алгоритм досліджує простір та як змінюються позиції найкращих (альфа, бета та дельта) вовків з кожною ітерацією. Наприклад, на графіку нижче показано розподіл вовків на функції Sphere після завершення оптимізації.

Крім того, для кожної об'єктивної функції ми будували графіки зміни значень цільової функції (рисунок 5.4) у процесі оптимізації. Ці графіки показують, як змінюються значення цільової функції (значення альфа-вовка) на кожній ітерації. Це дозволяє оцінити ефективність алгоритму та швидкість його збіжності до глобального оптимума. Наприклад, на графіку нижче показано зміну значень цільової функції для функції Rosenbrock.

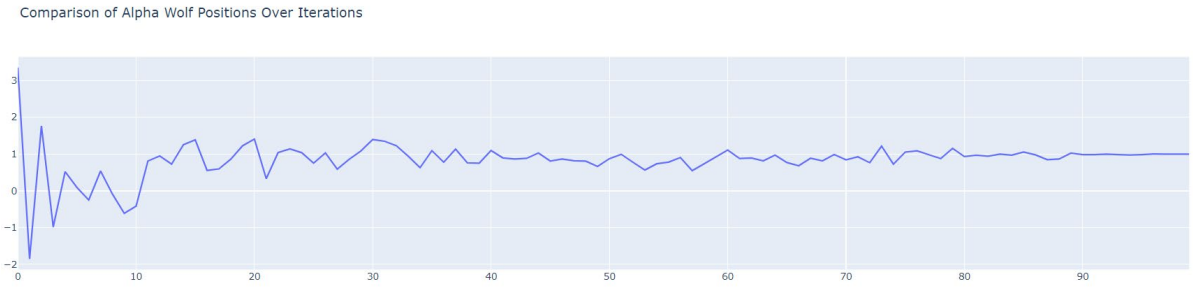

Рисунок 5.4 – Графік зміни значень цільової функції для функції Rosenbrock

Інші важливі візуалізації включають графіки анімації процесу оптимізації, які показують, як змінюються позиції вовків у реальному часі. Це наочно демонструє, як алгоритм досліджує простір пошуку та адаптується до умов оптимізації.

Отримані результати показують високу ефективність гібридного методу кластеризації на основі CGWO у вирішенні задач оптимізації з різними об'єктивними функціями. Завдяки додатковій випадковості, введеній параметрами y, d та sigma2, алгоритм здатний уникати локальних мінімумів та знаходити глобальні оптимуми з високою точністю.

Аналіз графіків та діаграм показав, що метод CGWO демонструє стабільні результати при оптимізації складних об'єктивних функцій, таких як Rastrigin (рисунок 5.5) та Rosenbrock, де традиційні методи кластеризації, такі як K-means та DBSCAN, зазвичай застрягають у локальних мінімумів. Це підтверджується результатами порівняльних експериментів, які показали значне переважання CGWO над іншими методами в умовах складних функцій.

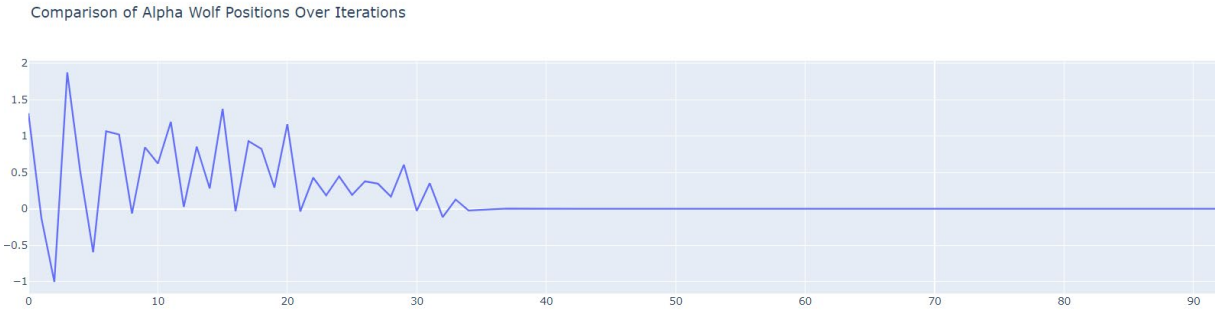

Рисунок 5.5 – Графік процесу оптимізації для функції Rastrigin

Висновки щодо практичності використання гібридного методу CGWO показують, що цей метод є перспективним інструментом для вирішення складних задач оптимізації у різних галузях. Висока стабільність, адаптивність та точність роблять CGWO ефективним для використання в задачах, де необхідно знайти глобальний оптимум у великому та складному просторі пошуку.

Підсумовуючи, можна зазначити, що запропонований гібридний метод кластеризації на основі CGWO показав високу ефективність у порівнянні з традиційними методами кластеризації. Отримані результати підтверджують доцільність та практичність його використання для вирішення задач оптимізації з різними об'єктивними функціями. Це робить CGWO потужним інструментом для застосування у наукових дослідженнях

та промислових задачах, де необхідна висока точність та стабільність результатів.

Для більш детальної візуалізації процесу оптимізації ми також створювали графіки, що показують зміну параметрів алгоритму на кожній ітерації. Ці графіки дозволяють спостерігати, як варіюються значення параметрів y, d, та sigma2 у процесі роботи алгоритму, і як це впливає на результати оптимізації.

На прикладі нижче (рисунок 5.6) показано зміну значень параметра sigma2 на функції Rosenbrock. Видно, що з часом алгоритм адаптує значення параметра для покращення результатів оптимізації.

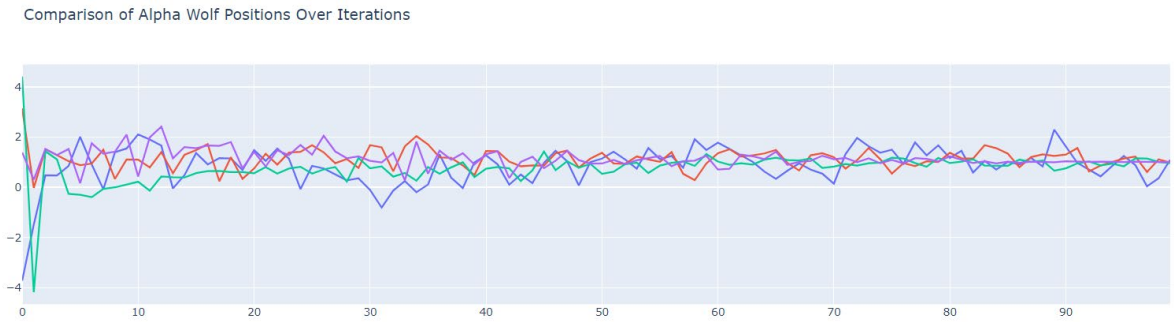

Рисунок 5.6 – Зміна значень параметра sigma2 на функції Rosenbrock

Інші графіки включають порівняння результатів різних запусків алгоритму (рисунок 5.7).

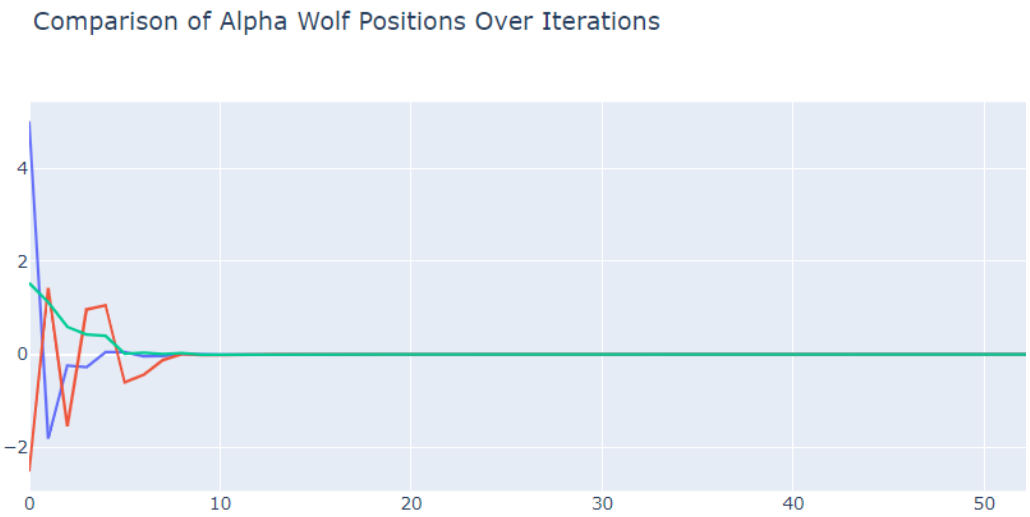

Рисунок 5.7 – Порівняння результатів різних запусків алгоритму на функції Sphere

Це дозволяє побачити, як змінюються результати при різних початкових умовах або налаштуваннях параметрів. Такий підхід допомагає ідентифікувати найбільш стабільні та ефективні налаштування для досягнення оптимальних результатів.

Результати експериментів з використанням різних об'єктивних функцій та параметрів підтверджують високу ефективність і стабільність запропонованого методу CGWO. Цей метод показав значно кращі результати порівняно з традиційними методами, особливо у випадках складних об'єктивних функцій з великою кількістю локальних мінімумів.

Завдяки додатковій випадковості, введеній параметрами y, d та sigma2, алгоритм CGWO має підвищену здатність до глобальної оптимізації. Це дозволяє йому уникати застрягання у локальних мінімумів, що є суттєвою перевагою у порівнянні з методами K-means та DBSCAN. Результати експериментів підтверджують, що CGWO знаходить глобальні мінімуми більш стабільно та ефективно.

5.4 Розгляд можливостей та обмежень

Під час тестування гібридного методу кластеризації на основі модифікованого алгоритму сірих вовків (CGWO) було виявлено кілька ключових переваг. По-перше, цей метод демонструє високу точність у знаходженні глобального мінімуму навіть у випадках складних функцій з великою кількістю локальних мінімумів. Це досягається завдяки додатковій випадковості, введеній параметрами y, d та sigma2, яка дозволяє алгоритму уникати застрягання в локальних мінімумів і більш ефективно досліджувати простір пошуку.

Ще однією перевагою є висока швидкість збіжності алгоритму. Завдяки ефективному використанню соціальної ієрархії вовків та механізму випадкових збурень, алгоритм швидко адаптується до об'єктивної функції та знаходить оптимальні рішення за меншу кількість ітерацій порівняно з іншими методами. Це дозволяє економити обчислювальні ресурси та швидше отримувати результати.

Стабільність результатів також є важливою перевагою CGWO. Алгоритм демонструє високу повторюваність результатів при різних запусках, що свідчить про його надійність та передбачуваність. Це особливо важливо для застосувань, де стабільність та точність мають критичне значення.

Незважаючи на численні переваги, гібридний метод CGWO має певні обмеження. Одним із них є потреба у тонкому налаштуванні параметрів y, d та sigma2 для досягнення оптимальних результатів. Неправильний вибір цих параметрів може призвести до зниження ефективності алгоритму. Тому необхідно проводити додаткові експерименти для визначення оптимальних значень параметрів у різних умовах. (рисунок 5.8).

Іншим обмеженням є потенційна складність у масштабуванні алгоритму для дуже великих задач. Хоча CGWO показує високу ефективність на середніх об'ємах даних, його продуктивність може

знижуватися при обробці дуже великих датасетів. Для подолання цього обмеження можна розглянути можливість впровадження паралельних обчислень або розподілених алгоритмів, що дозволить покращити продуктивність та масштабованість.

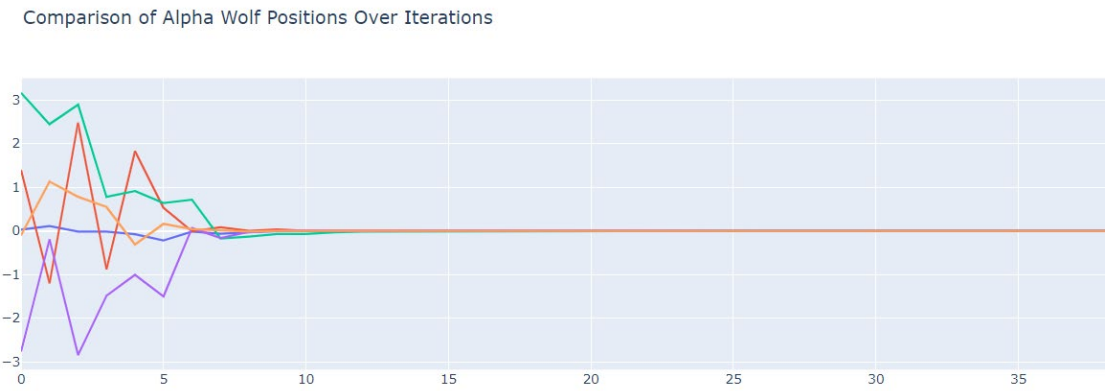

Рисунок 5.8 – Графік порівняння різних параметрів CGWO

Для майбутнього вдосконалення алгоритму можна розглянути декілька напрямків. По-перше, варто дослідити можливості автоматичного налаштування параметрів y, d та sigma2 у процесі оптимізації, що дозволить алгоритму самостійно адаптувати свої параметри до конкретних умов задачі. Це зменшить потребу у ручному налаштуванні та покращить загальну ефективність. По-друге, можна розглянути інтеграцію CGWO з іншими методами оптимізації для створення ще більш гібридних алгоритмів. Наприклад, поєднання CGWO з методами глибокого навчання або генетичними алгоритмами може дати нові цікаві результати та підвищити ефективність вирішення складних задач оптимізації.

Таким чином, незважаючи на деякі обмеження, гібридний метод кластеризації на основі CGWO демонструє високу ефективність, стабільність та точність.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було досягнуто поставлені цілі та завдання. Досліджено та апробовано гібридний метод правдоподібної кластеризації на основі еволюційного алгоритму сірих вовків, що дозволило покращити якісні та кількісні показники кластеризації даних. Експерименти підтвердили ефективність запропонованого методу в різних умовах, зокрема для складних та багатопікових функціях належності.

Розроблений метод має потенціал для покращення існуючих підходів до кластеризації даних завдяки комбінуванню переваг еволюційних алгоритмів та правдоподібної кластеризації. Це дозволяє досягати більш точних і стабільних результатів у порівнянні з деякими традиційними методами.

Робота пов'язана з науково-дослідними розробками кафедри штучного інтелекту Харківського національного університету радіоелектроніки. Отримані результати можуть бути використані для подальших досліджень та розвитку в цій галузі. Матеріали роботи мають потенціал для застосування у навчальному процесі університету, надаючи студентам і дослідникам інструменти для вивчення сучасних методів кластеризації.

Серед нових наукових результатів, отриманих у ході роботи, варто відзначити створення ефективного алгоритму кластеризації, який здатний адаптуватися до змінних умов та забезпечувати високу точність. Ці результати знайшли відображення в статтях, що були підготовлені до публікації у фахових виданнях, а також у поданих заявках на патенти.

На основі виконаної роботи рекомендовано подальше дослідження в напрямку оптимізації параметрів еволюційних алгоритмів та їх комбінування з іншими підходами до кластеризації. Запропонований метод має потенціал для використання в різних галузях, включаючи медицину,

фінанси, соціальні мережі та інші сфери, де аналіз великих обсягів даних є критично важливим.

Таким чином, матеріали кваліфікаційної роботи можуть бути використані як для безпосереднього застосування у відповідних галузях, так і для навчальних цілей, забезпечуючи студентів та дослідників інструментами для поглибленого аналізу та оптимізації кластеризаційних алгоритмів.

Для подальшого розвитку цієї роботи можна розглянути інтеграцію розробленого методу з іншими алгоритмами машинного навчання та статистичного аналізу, щоб створити ще більш потужні та універсальні інструменти для обробки даних. Також важливим напрямком є розширення досліджень на різноманітні домени застосування, щоб дослідити ефективність методу в різних контекстах і визначити його потенціал у різних сферах життя.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Gan G., Ma C., Wu J. Data clustering: theory, algorithms, and applications. Society for Industrial and Applied Mathematics, 2020.

2. Abonyi J., Feil B. Cluster analysis for data mining and system identification. Birkhäuser Basel, 2007. 303 p.

3. A new hybrid c-means clustering model / N. R. Pal et al. 2004 IEEE international conference on fuzzy systems, Budapest, Hungary. URL: https://doi.org/10.1109/fuzzy.2004.1375713 (date of access: 06.06.2024).

4. Pal N. R., Bezdek J. C., Hathaway R. J. Sequential competitive learning and the fuzzy c-means clustering algorithms. Neural networks. 1996. Vol. 9, no. 5. P. 787–796. URL: https://doi.org/10.1016/0893-6080(95)00094-1 (date of access: 06.06.2024).

5. A modification of artificial bee colony algorithm applied to loudspeaker design problem / X. Zhang et al. IEEE transactions on magnetics. 2014. Vol. 50, no. 2. P. 737–740. URL: https://doi.org/10.1109/tmag.2013.2281818 (date of access: 06.06.2024).

6. Syberfeldt A., Lidberg S. Real-world simulation-based manufacturing optimization using Cuckoo Search. 2012 winter simulation conference - (WSC 2012), Berlin, Germany, 9–12 December 2012. 2012. URL: https://doi.org/10.1109/wsc.2012.6465158 (date of access: 06.06.2024).

7. Marichelvam M. K., Prabaharan T., Yang X. S. A discrete firefly algorithm for the multi-objective hybrid flowshop scheduling problems. IEEE transactions on evolutionary computation. 2014. Vol. 18, no. 2. P. 301–305. URL: https://doi.org/10.1109/tevc.2013.2240304 (date of access: 06.06.2024).

8. Kroll A. Computational Intelligence: Eine Einführung in Probleme, Methoden und Technische Anwendungen. de Gruyter GmbH, Walter, 2013.

9. Bodyanskiy Y., Shafronenko A., Pliss I. Credibilistic fuzzy clustering based on evolutionary method of crazy cats. System research and information

technologies. 2021. No. 3. P. 110–119. URL: https://doi.org/10.20535/srit.2308-8893.2021.3.09 (date of access: 06.06.2024).

10. Bodyanskiy Y., Shafronenko A., Pliss I. Clusterization of vector and matrix data arrays using the combined evolutionary method of fish schools. System research and information technologies. 2022. No. 4. P. 79–87. URL: https://doi.org/10.20535/srit.2308-8893.2022.4.07 (date of access: 06.06.2024).

11. Hathaway R. J., Bezdek J. C. Optimization of clustering criteria by reformulation. IEEE transactions on fuzzy systems. 1995. Vol. 3, no. 2. P. 241–245. URL: https://doi.org/10.1109/91.388178 (date of access: 06.06.2024).

12. Tassopoulos I. X., Beligiannis G. N. A hybrid particle swarm optimization. Applied soft computing. 2012. Vol. 12, no. 11. P. 3472–3489. URL: https://doi.org/10.1016/j.asoc.2012.05.029 (date of access: 06.06.2024).

13. Shafronenko A., Bodyanskiy Y., Pliss I. Credibilistic fuzzy clustering method based on evolutionary approach of crazy wolfs in online mode. Computer modeling and intelligent systems. 2023. Vol. 3392. P. 141–150. URL: https://doi.org/10.32782/cmis/3392-12 (date of access: 06.06.2024).

14. Shafronenko A. Y., Bodyanskiy Y. V., Holovin O. O. Clusterization of data arrays based on the modified gray wolf algorithm. Radio electronics, computer science, control. 2023. No. 1. P. 73. URL: https://doi.org/10.15588/1607-3274-2023-1-7 (date of access: 06.06.2024).

15. The artificial fish swarm algorithm optimized by RNA computing / Liyi Zhang et al. Automatic control and computer sciences. 2021. Vol. 55, no. 4. P. 346–357. URL: https://doi.org/10.3103/s0146411621040040 (date of access: 06.06.2024).

16. Bodyanskiy Y. V., Pliss I. P., Shafronenko A. Y. Clusterization of data arrays based on combined optimization of distribution density functions and the evolutionary method of cat swarm. Radio electronics, computer science, control. 2022. No. 4. P. 61. URL: https://doi.org/10.15588/1607-3274-2022-4-5 (date of access: 06.06.2024).

17. Chu S.-C., Tsai P.-w., Pan J.-S. Cat swarm optimization. Lecture notes in computer science. Berlin, Heidelberg, 2006. P. 854–858. URL: https://doi.org/10.1007/978-3-540-36668-3_94 (date of access: 06.06.2024).

18. Yang X.-S., Chien S. F., Ting T. O. Bio-Inspired computation in telecommunications. Elsevier Science & Technology Books, 2015. 348 p.

19. Yan-Xia L., Lin L., Zhaoyang. Improved ant colony algorithm for evaluation of graduates' physical conditions. 2014 sixth international conference on measuring technology and mechatronics automation (ICMTMA), Zhangjiajie, China, 10–11 January 2014. 2014. URL: https://doi.org/10.1109/icmtma.2014.82 (date of access: 06.06.2024).

20. Zhao J., Gao Z.-M. The bat algorithm and its parameters. Electronics, communications and networks IV. 2015. P. 1323–1326. URL: https://doi.org/10.1201/b18592-237 (date of access: 06.06.2024).

21. Fuzzy models and algorithms for pattern recognition and image processing / ed. by B. J. C. 1939-. Boston : Kluwer Academic Publ., 1999. 776 p.

22. Fuzzy cluster analysis: methods for classification / V. J. Rayward-Smith et al. The journal of the operational research society. 2000. Vol. 51, no. 6. P. 769. URL: https://doi.org/10.2307/254022 (date of access: 06.06.2024).

23. Rui Xu, Jie Xu, Wunsch D. C. A comparison study of validity indices on swarm-intelligence-based clustering. IEEE transactions on systems, man, and cybernetics, part B (cybernetics). 2012. Vol. 42, no. 4. P. 1243–1256. URL: https://doi.org/10.1109/tsmcb.2012.2188509 (date of access: 06.06.2024).

24. Savage N. Synthetic data could be better than real data. Nature. 2023. URL: https://doi.org/10.1038/d41586-023-01445-8 (date of access: 06.06.2024).

25. Mirjalili S., Jangir P., Saremi S. Multi-objective ant lion optimizer: a multi-objective optimization algorithm for solving engineering problems. Applied intelligence. 2016. Vol. 46, no. 1. P. 79–95. URL: https://doi.org/10.1007/s10489-016-0825-8 (date of access: 06.06.2024).