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

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Модифікований алгоритм штучної бджолиної колонії для розв'язання задач оптимізації»

Виконала: студентка 2-ого курсу групи 1ІСТ-21м
спеціальності 126 – Інформаційні системи та технології
Ярська В. І.

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

«____» _____ 2022 р.

Опонент: _____

_____

«____» _____ 2022 р.

Допущено до захисту
в.о. зав. кафедри АІІТ
д.т.н., проф. Бісікало О. В.
«____» _____ 2022 р.

Вінниця ВНТУ – 2022 рік

Вінницький національний технічний університет

Факультет інтелектуальних інформаційних технологій та автоматизації

Кафедра автоматизації та інтелектуальних інформаційних технологій

Рівень вищої освіти II-ий (магістерський)

Галузь знань – 12 – Інформаційні технології

Спеціальність – 126 – Інформаційні системи та технології

Освітньо-професійна програма – Інформаційні технології аналізу даних та зображень

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Ярській Вікторії Ігорівні

1. Тема роботи: Модифікований алгоритм штучної бджолиної колонії для розв'язання задач оптимізації.

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

   Затверджені наказом ВНТУ від «14» 09 2022 року № 203.
2. Строк подання роботи студентом: до «до 23» 12 2022 р.
3. Вихідні дані до роботи: розмірність простору пошуку *D*; параметри алгоритму (популяція бджіл *N*; кількість ітерацій, на яких забувається рішення *K*; кількість ітерацій алгоритму *limit*); кількість запусків для збору статистики *m*; набір тестових функцій; середовище програмування *MatLab*.
4. Зміст розрахунково-пояснювальної записки: вступ; аналіз предметної області роботи; розробка математичної моделі модифікованого алгоритму штучної бджолиної колонії; розробка програмного забезпечення та експериментальні дослідження; економічний розділ; висновки; список використаних джерел.
5. Перелік графічного матеріалу: приклад випадкового блукання; залежність кількості публікацій на тему ройового інтелекту від року; принцип роботи ройової оптимізації; приклад поведінки бджіл; приклад роботи алгоритму штучної бджолиної колонії; схема програми.

6. Консультанти розділів магістерської кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Іванов Ю. Ю., к.т.н., доцент каф. АІІТ	19.09.2022	07.12.2022
4	Лесько О. Й. к.е.н., проф. каф. ЕПтаВМ		

7. Дата видачі завдання: «19»_____09_____ 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області роботи	10.10.22	
2	Розробка математичної моделі модифікованого алгоритму штучної бджолиної колонії	25.10.22	
3	Розробка програмного забезпечення та експериментальні дослідження	10.11.22	
4	Підготовка економічного розділу	20.11.22	
5	Оформлення пояснювальної записки і графічного матеріалу	30.11.22	
6	Попередній захист роботи	до 07.12.22	
7	Остаточний захист роботи	до 23.12.22	

Студентка _____ (підпис) Ярська В. І. (прізвище та ініціали)

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

АНОТАЦІЯ

УДК 519.85 + 004.8

Ярська В. І. Модифікований алгоритм штучної бджолиної колонії для розв'язання задач оптимізації. Магістерська кваліфікаційна робота зі спеціальності 126 – Інформаційні системи та технології, освітньо-професійна програма – Інформаційні технології аналізу даних та зображень. Вінниця: ВНТУ, 2022. 91 с.

На укр. мові. Бібліогр.: 33 назви; рис.: 18; табл.: 18.

У роботі проаналізовано, теоретично обгрунтовано та модифіковано алгоритм ройового інтелекту, який інспіровано колективною поведінкою бджол у колонії. Розроблено відповідне алгоритмічне та програмне забезпечення, яке дозволяє виконати експериментальні дослідження у ході розв'язання задач оптимізації функції.

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

ABSTRACT

Iarska V. I. A modified artificial bee colony algorithm for solving optimization problems. Master's thesis in specialty 126 – Information systems and technologies, educational and professional program – Information technologies of data and image analysis. Vinnitsa: VNTU, 2022. 91 p.

In Ukrainian language. Bibliography: 33 titles; fig.: 18; tabl.: 18.

In this work has been analyzed, theoretically substantiated and modified an algorithm of swarm intelligence, which is inspired by the collective behavior of bees in colony. The algorithms and software, allows us to perform some experimental researches for solving function optimization tasks, have been developed.

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

ЗМІСТ

ВСТУП

Актуальність. У ході розв'язання задач оптимізації складних нелінійних багатовимірних функцій необхідно враховувати їх властивості, що суттєво впливає на вибір відповідного алгоритму. Оптимальне рішення задачі оптимізації може бути отримано з використанням класичних градієнтних (Хук-Дживс, Флетчер-Рівс, Ньютон) алгоритмів прямого пошуку, які використовують значення похідних. Однак недоліком є необхідність визначення характеру цільової функції (неперервність, унімодальність, диференційовність функції тощо), залежність від початкового наближення і, як наслідок, велика ймовірність потрапляння в пошукову пастку в локальному екстремумі [1].

У зв'язку з цим постає проблема розробки та використання таких алгоритмів оптимізації, які здатні знаходити рішення задачі в умовах невизначеності, коли про характер функції мало що відомо.

У сучасному світі алгоритми обчислювального інтелекту (computational intelligence) або інтелектуальні алгоритми (intelligence algorithms) набули широкого поширення для розв'язання низки комплексних оптимізаційних задач у науці та техніці (обробка контенту, пошук маршрутів, налаштування експертних систем, тестування технічних засобів тощо). Серед таких алгоритмів оптимізації окремий клас утворює ройовий інтелект (swarm intelligence), принципи роботи якого інспіровані живою або неживою природою [2].

У світовій науковій літературі описано багато оригінальних ройових алгоритмів оптимізації. Наприклад, можна виділити наукові роботи Д. Карабоги [3-6], Дж.-Л. Лю [7], В.Ф. Гао [8], С.В. Устенко [9] та

інших. Вони виділяють алгоритм оптимізації, інспірований поведінкою медоносних бджіл у колонії, як один із найбільш ефективних та зручних алгоритмів, який можна використовувати для розв'язання різноманітних оптимізаційних задач. Тому актуальною задачею є аналіз особливостей роботи даного представника ройового інтелекту та розробка модифікацій, які підвищать ефективність пошуку розв'язку.

Мета і задачі дослідження. Метою роботи є підвищення ефективності розв'язання задачі неперервної оптимізації функцій з використанням алгоритму штучної бджолиної колонії за рахунок застосування нової моделі поведінки бджіл-спостерігачів.

Для досягнення мети необхідно розв'язати наступні задачі:

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

– модифікувати математичну модель алгоритму штучної бджолиної колонії;

– розробити програмні засоби та оцінити ефективність роботи запропонованої модифікації.

Об'єктом дослідження є оптимізація функцій від багатьох змінних.

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

Методи дослідження. У роботі використано поняття теорії оптимізації та ройового інтелекту під час дослідження роботи алгоритму штучної бджолиної колонії у ході розв'язання задачі оптимізації функцій. Для аналізу та перевірки достовірності отриманих теоретичних положень застосовано експериментальне дослідження. Оброблення даних виконано

за допомогою середовища комп'ютерного моделювання MatLab від корпорації MathWorks.

Наукова новизна отриманих результатів полягає у наступному:

1. Запропоновано модифікацію алгоритму штучної бджолиної колонії, особливістю якої є математична модель поведінки бджіл-спостерігачів, що дозволяє підвищити точність розв'язання задачі неперервної оптимізації функцій.

Практичне значення результатів роботи:

1. Розроблене математичне, алгоритмічне та програмне забезпечення дозволяє дослідити вплив параметрів алгоритму штучної бджолиної колонії на ефективність його роботи у ході розв'язання задач оптимізації функцій.

Апробація результатів та публікації. За результатами даної роботи опубліковано 2 тези доповіді на L науково-технічній конференції підрозділів ВНТУ (м. Вінниця, 2021) [10] та науково-практичній конференції з міжнародною участю "Актуальні задачі медичної, біологічної фізики та інформатики" (м. Вінниця, 2022) [11]. Основні результати можна використати на практиці, що підтверджено довідкою про можливість впровадження розробки.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ РОБОТИ

1.1 Постановка задачі оптимізації

Під час проектування різних систем досить часто виникає необхідність досягнути мети не як-небудь, а деяким оптимальним способом, де під оптимальним розуміють такий спосіб, при якому досягається мінімальне або максимальне значення деякого критерію якості.

Однією із найважливіших ознак у алгоритмі оптимізації є наявність стохастичності. З її урахуванням такі методи можна поділити на три категорії [12]:

– детерміновані (аналітичні);

– стохастичні (метаевристичні, наявна ймовірнісна природа);

– комбіновані (гібридні).

Слід зазначити, що у ході розв'язання інженером задач проектування та управління технічними системами проявляються їх властивості:

1. Висока розмірність простору. Наявна велика кількість керованих змінних n, кожна з яких може приймати m значень, тобто загальна кількість можливих рішень становить m^n.

2. Нелінійність. Як правило, критерій оптимальності і обмеження задачі є нелінійними функціями в силу нелінійності фізичних законів, які визначають залежності в системі.

3. Мультимодальність. Нелінійні залежності і велика кількість альтернатив призводить до великої кількості локальних екстремумів. Тобто існує множина рішень, локальна зміна яких не дає можливості покращити

результат, тому процес оптимізації може зупинитися в локальному оптимумі.

4. Недиференційовність. Багато актуальних задач (комбінаторних, алгоритмічних) не мають аналітичного виразу для функції критерію оптимальності, який можна було б досліджувати за допомогою методів диференціального числення.

5. Багатокритеріальність. Технічні системи можуть виконувати декілька функцій, кожна з яких може мати свої критерії ефективності (різні цільові функції та обмеження). Крім того, важливо враховувати показники, які пов'язані зі споживанням матеріальних ресурсів, витратами часу тощо.

6. Багатофакторність. Для досягнення максимальної ефективності роботи системи потрібно вирішувати задачу не тільки параметричної, але й структурної оптимізації, розглядати велику кількість різних керованих змінних.

7. Робота в режимі "онлайн". Дана властивість важлива для задач керування об'єктами. Швидкість, з якою система реагує на зміни навколишнього середовища, важлива для безпеки і економічності.

Зазвичай задачі оптимізації мають загальну форму і класифікуються як задачі мінімізації дійснозначної функції $f(x)$ на певній множині Ω n-вимірного векторного аргументу $x=(x'_1,x'_2,...,x'_n)^T$. Множина Ω задається обмеженнями на компоненти вектора x, які задовольняють систему з K рівнянь $h_k(x)=0$ та J нерівностей $g_j(x)\geq 0$, а також обмежені зверху та знизу ($x_i^{upper}\geq x'_i\geq x_i^{lower}$). Тоді математичну постановку можна задати таким чином [13-16]

$$\begin{cases} f(x) \to extremum\ (\max, \min); \\ h_k(x) = 0,\ k = 1,...,K; \\ g_j(x) \geq 0,\ j = 1,...,J; \\ x_i^{'upper} \geq x'_i \geq x_i^{'lower},\ i = 1,...,n \end{cases} \tag{1.1}$$

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

$$f(x) \to extremum\ (\max, \min),\ x \in \Omega;\ \Omega = \{-\infty; +\infty\}. \tag{1.2}$$

У випадку, коли функція $f(x)$ має в точці x^* мінімум, то для функції $-f(x)$ в цій же точці – максимум. Тому для пошуку максимуму застосовують ті ж методи, що і для пошуку мінімуму.

У функції може бути багато локальних мінімумів (багатоекстремальна функція). Для знаходження абсолютного (глобального) мінімуму необхідно знайти всі локальні мінімуми, порівняти їх і вибрати найменше значення $f(x^*)$.

Відомо, що кількість локальних оптимумів збільшується експоненціально збільшенню розмірності вектора рішення, якщо цільова функція $f(x)$ є мультимодальною функцією.

1.2 Огляд алгоритмів оптимізації

Повний перебір є найбільш простим підходом розв'язання оптимізаційних задач, шляхом безпосереднього перебору всіх можливих рішень задачі. Складність методу сильно залежить від розмірності простору пошуку. Наприклад, якщо потрібно знайти оптимальну комбінацію n змінних, кожна з яких може приймати m значень, то необхідно проаналізувати m^n комбінацій. Для простої, на перший погляд, задачі з $n = 25$ і $m = 10$ кількість можливих комбінацій буде складати 10^{25}. Тоді, щоб перебрати всі варіанти, для пристрою, який обробляє 10^{12} варіантів за секунду, потрібно більше 317 тисяч років [16].

Ще одним потужним алгоритмом є пошук з поверненням або перебір з поверненням, який представлено у 1950 році американським математиком Лемером. Суть пошуку з поверненням очевидна: розвиток (розширення) раніше сформованого часткового розв'язку. Якщо це неможливо виконати, то алгоритм повертається до одного з минулих станів (розв'язків) та намагається розвивати його. Даний алгоритм має прямий зв'язок з методом гілок і меж, оскільки останній є його модифікацією [2].

Інші удосконалення бектрекінгу можна зустріти під такими назвами: пошук в глибину, метод проб і помилок, алгоритм Девіса-Патнема тощо. Пошук з поверненням може бути реалізований стохастично, тобто процедура вибору нового рішення базується на ймовірнісному правилі. Однак виникає проблема налаштування параметрів вибору. Так, якщо максимальний крок переходу буде великим, то алгоритм може "застрягнути" в локальному екстремумі і не вийти з нього. Якщо ж крок малий, то алгоритм буде працювати дуже довго [1].

Удосконалення алгоритмів повного перебору і бектрекінгу є відсікання підмножин, які завідомо не містять оптимальний результат (неперспективні варіанти). Такий метод запропонували у 1960 році Ленд і Дойг. Він був названий метод гілок та меж. Для його роботи необхідні дві процедури: рекурсивне бінарне розгалуження і знаходження оцінок. Процедура розгалуження полягає в розбитті певної підмножини на менші підмножини, що у результаті дозволяє отримати пошукове дерево.

Процедура знаходження оцінок полягає в пошуку верхніх і нижніх меж для вирішення задачі на певній підмножині рішень. В основі методу гілок і меж лежить наступна ідея: якщо нижня межа значень функції на підмножині A пошукового дерева більша, ніж верхня межа на будь-якій раніше розглянутій підмножині B, то A виключається з подальших обчислень, а гілка дерева відрізається (правило відсіву). У результаті можна отримати певне рішення, яке називають рекордом R. Для пошуку глобального оптимуму необхідно перевірити всі вузли, в яких межа менша за поточний рекорд. Час розв'язання задачі оптимізації у загальному випадку експоненціально зростає зі збільшенням розмірності задачі, що робить застосування даного методу обмеженим. Приклад роботи методу оптимізації наведено на рисунку 1.1 [14].

Термін "динамічне програмування" був вперше використаний Р.Е. Беллманом у 1940 році для опису процесу розв'язання задачі, в якій процес прийняття рішення може бути розділений на окремі етапи (кроки), причому кожний крок визначається лише поточним станом системи. Динамічне програмування ґрунтується на принципах Беллмана.

Оптимальний розв'язок на кожному кроці визначається станом системи на початку цього кроку. Яким би не був стан системи в результаті

певної кількості кроків, на найближчому кроці потрібно вибирати стан таким чином, щоб він в сукупності з оптимальним вибором на всіх наступних кроках призводив до оптимального виграшу на всіх кроках, що залишилися, включаючи даний.

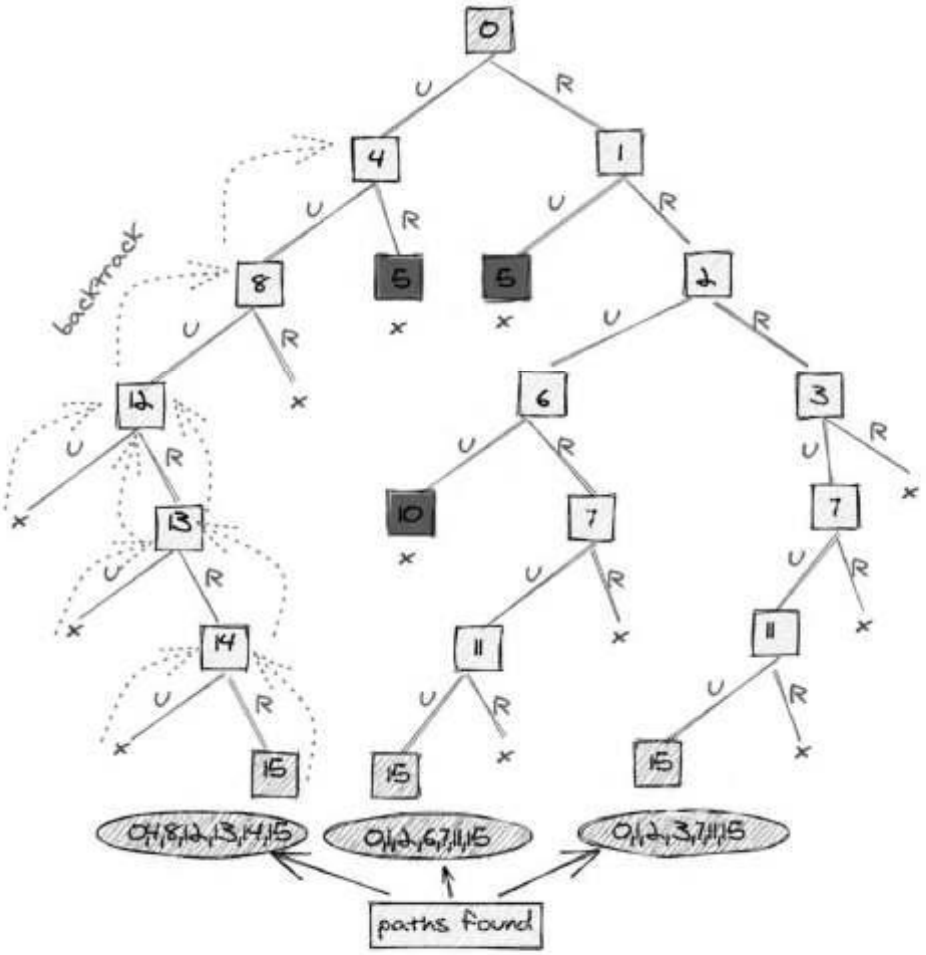

Рисунок 1.1 – Приклад роботи методу гілок та меж

Форма завдання, яке розв'язується методом динамічного програмування, не змінюється при зміні кількості кроків n. У цьому сенсі всякий конкретний процес із заданим числом кроків виявляється як би зануреним в сімейство подібних йому процесів і може розглядатися з позиції більш широкого класу задач [15].

Фактично динамічне програмування представляє собою математичний апарат, який дозволяє здійснювати оптимальне планування багатокрокових керованих процесів, розв'язуючи набір підзадач. До типових задач відносяться задачі про розрахунок траєкторії, рюкзак [1].

Однією з поширених на практиці задач є задача лінійного програмування, яка подається у вигляді: знайти екстремум функції [13]

$$Z = c_1 x_1 + c_2 x_2 + ... + c_n x_n \to ext(\max, \min), \tag{1.3}$$

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n \{\leq, \geq, =\} b_1; \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n \{\leq, \geq, =\} b_2; \\ ..... \\ a_{m1}x_1 + a_{m2}x_2 + ... + a_{mn}x_n \{\leq, \geq, =\} b_m, \end{cases} \tag{1.4}$$

$$x_1 \geq 0, x_2 \geq 0, ..., x_n \geq 0.$$

Отже, потрібно знайти значення змінних x_1, x_2, …, x_n, які задовольняють дані нерівності, тоді як цільова функція набуває екстремального (максимального чи мінімального) значення. Дану задачу можна звести до стандартної форми, коли в системі обмежень b_i невід'ємні, а всі обмеження є рівностями. Якщо задача лінійного програмування має оптимальний план, то цільова функція набуває екстремуму в одній із вершин багатокутника розв'язків.

Для розв'язання подібних задач існує симплекс-метод, ідея якого полягає у переборі вершин опуклого багатогранника в заданому багатовимірному просторі (рисунок 1.2). Алгоритм отримав широкого поширення завдяки своїй універсальності, тобто застосування до

різноманітних задач, включаючи багатокритеріальні, і невисокий рівень вимог до обчислювальної потужності технічних засобів.

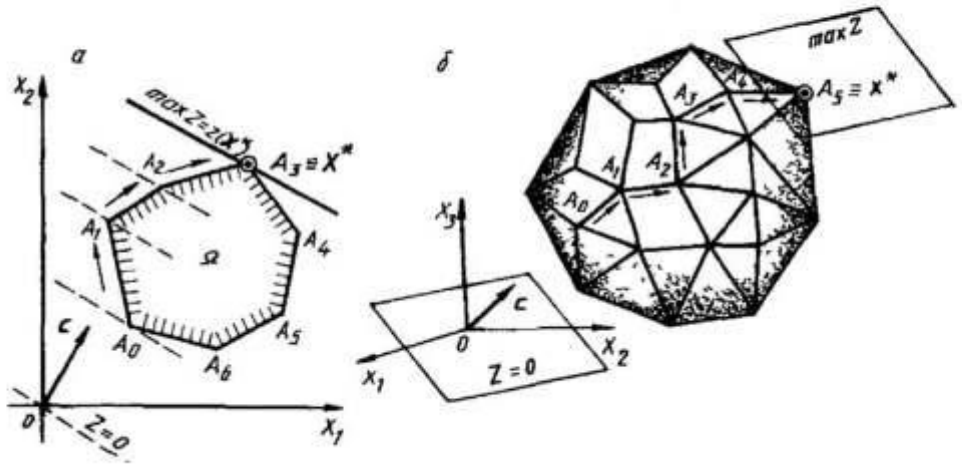

Рисунок 1.2 – Приклад роботи симплекс-методу

При використанні симплекс-методу задачу необхідно привести до канонічної форми, тобто відповідним чином записати цільову функцію та систему обмежень. Далі сформувати симплекс-таблицю та на кожній ітерації замінювати одну вільну змінну на базисну, при цьому коефіцієнти у таблиці перераховуються відповідно до алгоритму їх перетворення.

Клас чисельних методів, які використовують градієнти цільової функції для пошуку її оптимуму, дозволяє сформувати ефективні ітеративні обчислювальні схеми, які є досить простими для реалізації і часто застосовуються для розв'язання ряду нелінійних задач. Але градієнтні алгоритми також мають істотний недолік: одного разу досягнувши локального екстремуму, вони вже неспроможні вибратися з нього. Яскравими представниками даного класу є методи градієнтного спуску, Ньютона, Левенберга-Марквардта, квазіньютонівські методи тощо (рисунок 1.3) [13-16].

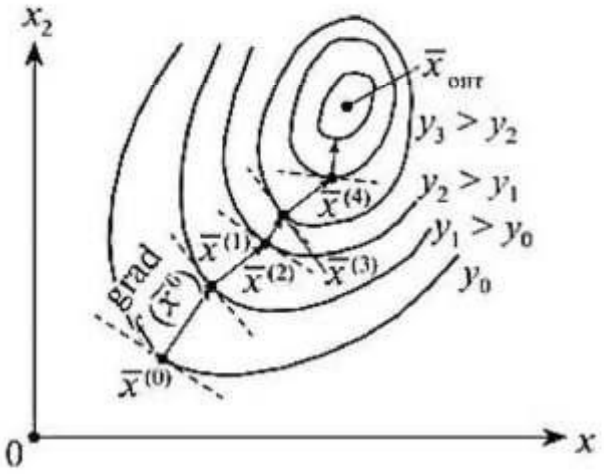

Рисунок 1.3 – Приклад роботи градієнтних алгоритмів

На початку XVIII століття Л. Лагранж розробив потужний алгоритм оптимізації, який пізніше отримав його ім'я – множники Лагранжа. Пізніше даний алгоритм був узагальнений у роботі Г. Куна і А. Такера, в якій були сформульовані необхідні умови розв'язання задачі нелінійного програмування. Оскільки метод Лагранжа можна застосовувати для задач лінійного та нелінійного програмування, то він використовується для розв'язання різних задач економіки, теорії управління, енергетики та кодування даних (задача оптимізації якості кодування аудіо- та відеоданих при заданому середньому бітрейті). Приклад наведено на рисунку 1.4 [1].

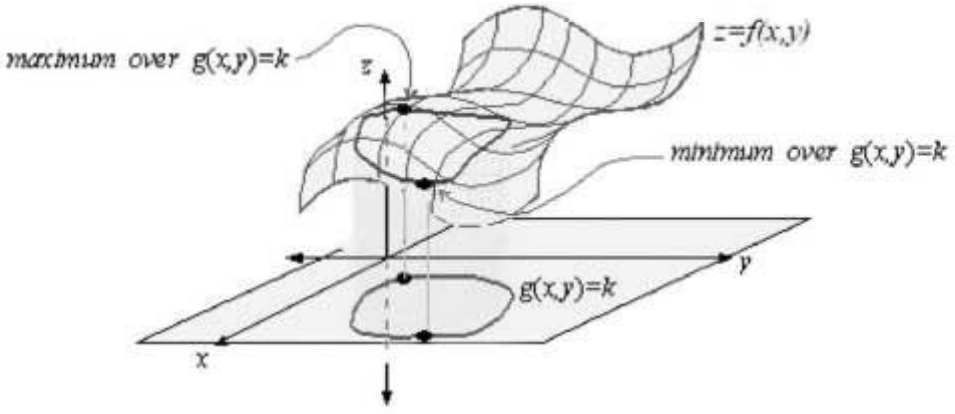

Рисунок 1.4 – Приклад роботи множників Лагранжа

Жадібні алгоритми дуже схожі на динамічне програмування, але їх відмінність в тому, що вони працюють з однією локальною задачею, ніколи не повертаючись назад, завжди застосовуючи локально оптимальний вибір. Наприклад, алгоритм Е.В. Дейкстри для пошуку найкоротшого шляху на графі є жадібним, оскільки на кожному кроці шукають нерозглянуту вершину з найменшою вагою, після чого оновлюють значення інших вершин [14].

Жадібні алгоритми можуть мати і стохастичні властивості. Для деяких із них, незалежно від початкових умов, можна довести асимптотичну збіжність до глобального оптимуму задачі. Але необхідно розуміти, що не існує загального підходу до формування жадібного алгоритму, тобто для кожної конкретної задачі його потрібно налаштовувати. Однак існує досить універсальний спосіб визначення доцільності використання подібного алгоритму для певної задачі. Необхідно з'ясувати чи є підмножини елементів даної задачі матроїдом. Якщо відповідь на це питання позитивна, то відповідно до теореми Радо-Едмондса до задачі можна застосувати жадібний алгоритм, який дозволить отримати оптимум. Приклад роботи жадібного алгоритму наведено на рисунку 1.5.

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

Алгоритми Монте-Карло (статистичних випробувань) або алгоритм стохастичного моделювання сформувався в рамках Манхеттенського проекту (створення атомної бомби), в якому фізики з Лос-Аламоської наукової лабораторії досліджували радіаційний захист, відстань, яку

нейтрони проходять у речовині до зіткнення з атомним ядром, та кількість енергії, яка виділяється. Зважаючи на велику кількість необхідних даних, задача не могла бути розв'язана за допомогою аналітичних розрахунків [1].

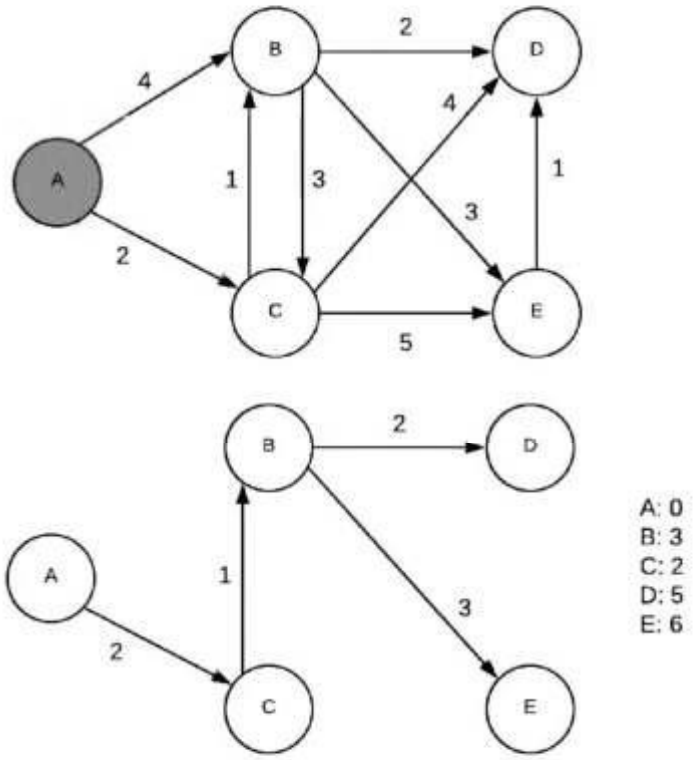

Рисунок 1.5 – Приклад роботи жадібного алгоритму

Дж. фон Нейман, Н.К. Метрополіс і С.М Улам в 1949 році запропонували розв'язати її на основі моделювання експерименту на комп'ютері з використанням “випадковості”. Будучи засекреченою, їхня робота вимагала кодове ім'я. Тому така оригінальна назва запозичена у міста Монте-Карло в князівстві Монако, відомому казино, в яких використовується один із самих відомих генераторів випадкових чисел – рулетка.

Алгоритми Монте-Карло представляють собою непараметричний підхід (не розглядається оцінка параметрів, наприклад, математичне

очікування або стандартне відхилення), який ґрунтується на моделюванні випадкових процесів із заданими характеристиками. Результат кожного окремого випробування залежить від деякої випадкової величини, розподіленої за деяким законом, тобто має випадковий характер.

Найбільш простий стохастичний алгоритм оптимізації – це алгоритм сходження на вершину, який представляє собою випадкову генерацію альтернатив (без будь-якого врахування якості раніше знайдених рішень) з подальшим вибором серед них найкращої (рисунок 1.6). Подібну процедуру можна легко розпаралелити, сформувавши "променевий пошук". Головним недоліком даного методу є відсутність пам'яті про попередні рішення [10].

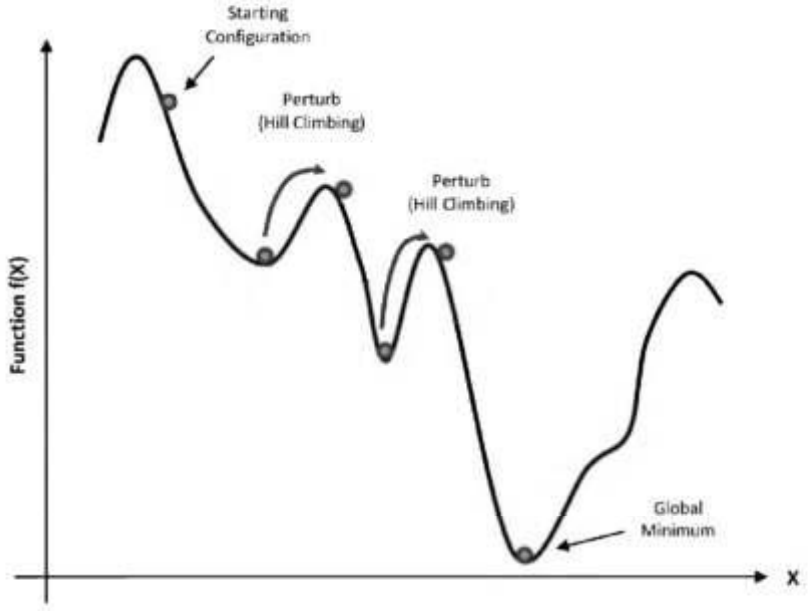

Рисунок 1.6 – Принцип роботи випадкового пошуку

У 1983 році вчені С. Кіркпатрік, Д. Желатт, М. Веччі та у 1985 році В. Черні, використовуючи ідеї цієї праці, вперше застосовують для розв'язання задачі комівояжера метод на основі відпалу, який названо

алгоритмом імітацією відпалу. Він заснований на використанні аналогій з процесом кристалізації речовини, в ході якого вона охолоджується і твердне, а швидкість руху молекул падає. Таким чином, поки штучна температура буде великою, алгоритм може робити великі кроки (аналогічні випадковим флуктуаціям в гарячій речовині) навіть у напрямках, які збільшують значення цільової функції. У багатьох випадках ця особливість дозволяє потрапити в окіл глобального мінімуму (природний стан речовини) [11].

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

В основі еволюційних алгоритмів лежить аналогія природного відбору (селекція, схрещування, мутація), але відбираються рішення оптимізаційної задачі, тобто виконується ітераційна процедура локального пошуку. Рішення представляється у вигляді вектора значень, який називається хромосомою або особиною, а їх набір на кожному кроці алгоритму називається популяцією. Кожна особина оцінюється на основі значення критерію оптимальності (фітнес-функції). Кращі результати (дочірні популяції або потомство) відбираються до наступної ітерації і впливають на поточні рішення – схрещуються, мутують (рисунок 1.7).

Принципові моменти механізму еволюції можуть бути формалізовані по-різному. Внаслідок цього з'являється велика кількість еволюційних алгоритмів. Перевагою еволюційних алгоритмів є простота застосування, а

недоліками – невисока ефективність для задач великої розмірності і складної структури. Критики справедливо вказують на низьку швидкість їх роботи, оскільки еволюція і в природі протікає повільно [2].

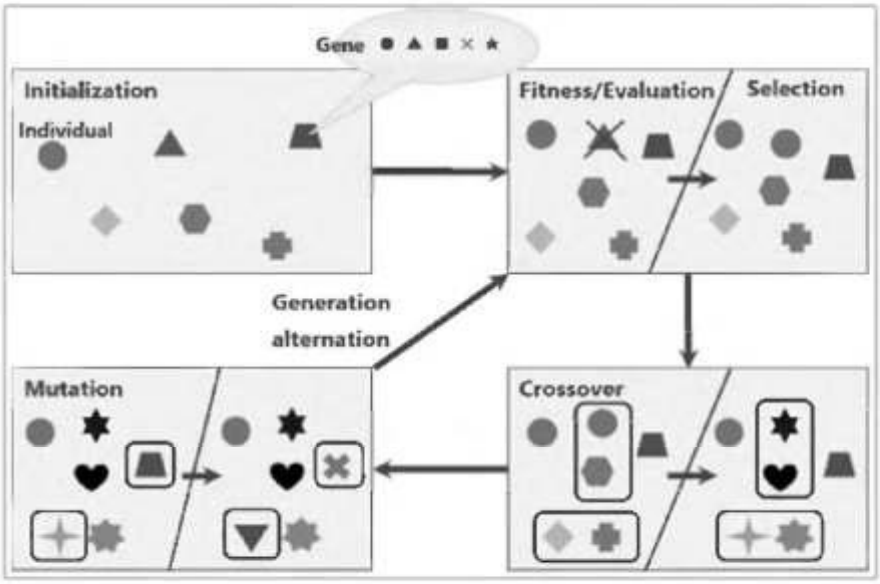

Рисунок 1.7 – Принцип роботи еволюції

У 1989 році з'являється праця Х. Бені і В. Цзіна, присвячена клітинним автоматам, в якій вперше фігурує термін "ройовий інтелект". Формально, рій може бути визначений як група мобільних програмних-агентів, які взаємодіють один з одним (прямо або опосередковано), впливаючи на своє оточення та навколишнє середовище. Така колективна узгодженість дій агентів призводить до розподілених колективних стратегій оптимізації. Отже, вчені звернули увагу на моделі колективної поведінки в природі і змогли застосувати їх для створення оптимізаційних алгоритмів, що показано у зростанні кількості праць (рисунок 1.8) [17].

Потужним алгоритмом ройового інтелекту є алгоритм наслідування поведінки мурашиної колонії, розроблений бельгійським вченим М. Доріго у 1992 році для розв'язання комбінаторних задач, наприклад, задачі

комівояжера. Він моделює дії колонії мурах у ході пошуку оптимального маршруту від мурашника до їжі у навколишньому середовищі. Концепція алгоритму активно вдосконалюється як для дискретних, так і для неперервних задач оптимізації [12].

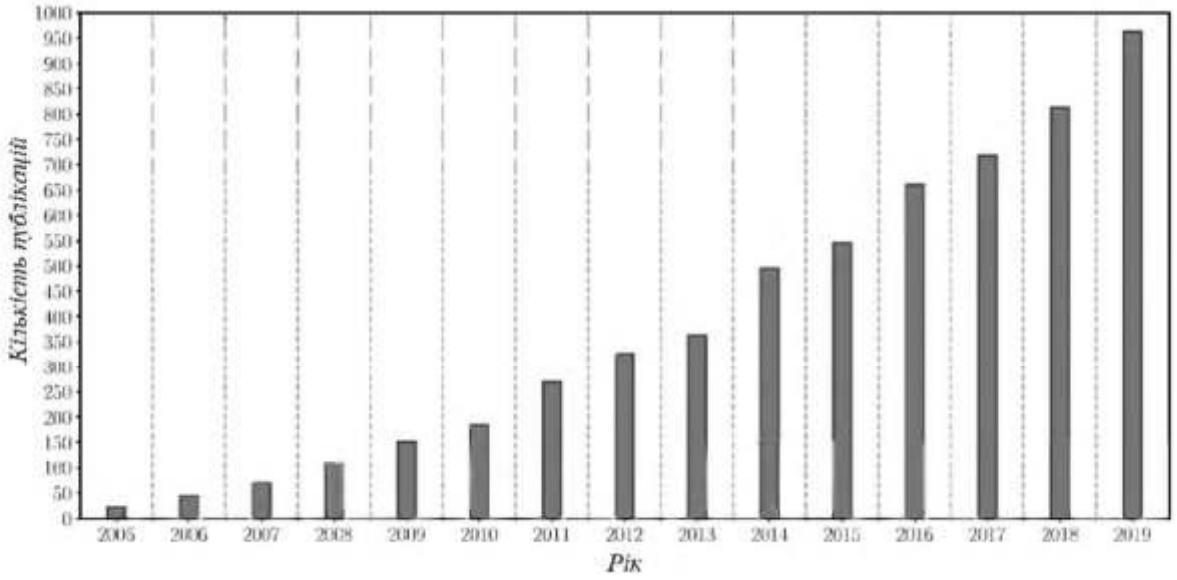

Рисунок 1.8 – Залежність кількості публікацій від року

Однією з перших моделей, яка реалізувала колективну поведінку, стала модель переміщення птахів у зграї, запропонована К. Рейнольдсом у 1986 році. Незважаючи на її простоту, вона давала правдоподібну візуалізацію колективної поведінки зграї. На базі даної моделі Дж. Кеннеді і Р. Еберхарт у 1995 році розробили алгоритм оптимізації неперервних нелінійних функцій, який назвали алгоритмом рою часток.

Основу алгоритму становить факт, що частки при формуванні рою рухаються до "центру тяжіння" (оптимального рішення), а простір пошуку заповнюється популяцією часток, кожна з яких у конкретний момент часу має низку параметрів. У часток є своя пам'ять, вони можуть обмінюватися

інформацією між собою. Для кожного положення частки обчислюється відповідне значення цільової функції, на основі якого за певними правилами, обчислюються нові координати і швидкість даної частки. Описаний алгоритм має безліч модифікацій (рисунок 1.9) [12].

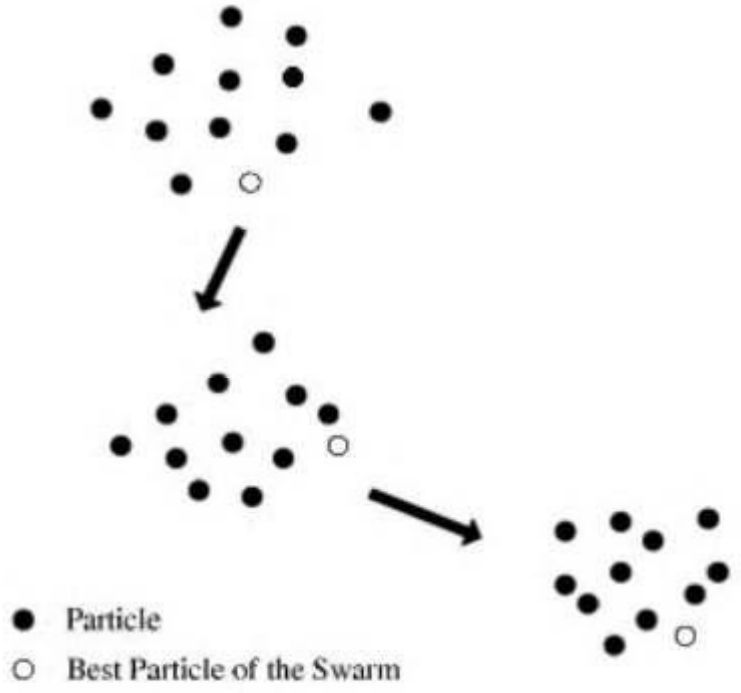

Рисунок 1.9 – Приклад роботи алгоритму рою часток

Одним з відносно нових алгоритмів ройового пошуку є алгоритм, інспірований поведінкою кажанів, запропонований К.-Ш. Янгом у 2010 році. Всі кажани використовують ехолокацію (певний тип ультразвуку), завдяки якому вони можуть літати і полювати. Даний алгоритм потенційно більш потужний, ніж алгоритм рою часток, генетичний алгоритм, а також гармонійний пошук [2].

У 2013 році С. Мірджалілі розробив алгоритм зграї сірих вовків, який імітує полювання та ієрархічний розподіл зграї. Для того, щоб математично змоделювати соціальну ієрархію вовків використовують найбільш

пристосоване рішення – альфа (α), друге та третє найкращі рішення, які називають бета (β) та дельта (δ) відповідно, а решта кандидатів-рішень вважаються омегою (ω) [13].

Перспективним є дослідження гібридних алгоритмів, які комбінують детерміновані та стохастичні підходи, наприклад, меметичний алгоритм, який використовує концепцію мемів (довільний алгоритм локальної оптимізації), імітуючи культурну еволюцію.

1.3 Висновки

У даному розділі визначено, що після того, як комп'ютерні системи стали досить швидкодіючими та відносно недорогими, метаевристики поступово використовують для розв'язання різних задач оптимізації, машинного навчання, розпізнавання образів тощо. Якщо задачу не можна розв'язати, використовуючи детермінований підхід, ройові алгоритми часто дозволяють знайти оптимальні або близькі до них рішення. Актуальною є задача дослідження та розробки ефективних ройових алгоритмів.

2 РОЗРОБКА МАТЕМАТИЧНОЇ МОДЕЛІ МОДИФІКОВАНОГО АЛГОРИТМУ ШТУЧНОЇ БДЖОЛИНОЇ КОЛОНІЇ

2.1 Загальна структура ройового алгоритму

Загальна схема ройового алгоритму включає наступні етапи [10, 12]:

1. Ініціалізація популяції. В області пошуку формуємо певну кількість початкових наближень до шуканого рішення задачі, тобто ініціалізуємо популяцію агентів. У ході цього процесу можна застосувати детерміновані та стохастичні методи. Зазвичай агентів розподіляють випадковим чином рівномірно по всій області пошуку екстремуму.

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

В якості базової послідовності випадкових чисел для імітації випадкових факторів різної природи необхідно вибирати таку послідовність, яка може бути отримана з найменшими витратами машинного часу і, крім того, забезпечує простоту і зручність подальших перетворень.

Практика показує, що найкраще даним вимогам задовольняє послідовність випадкових чисел з рівномірним розподілом на інтервалі:

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

У MatLab для генерації випадкового числа, розподіленого рівномірно на відрізку $[a, b] = [0, 1]$, використовується функція *unifrnd*(a, b).

Ще одним популярним випадковим блуканням є гаусівське блукання, яке використовує нормальний розподіл з математичним очікуванням μ і середнім квадратичним відхиленням σ [18]:

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

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

$$N(\mu,\sigma) = \mu + \sigma \cdot N(0,1). \qquad (2.3)$$

У MatLab для генерації випадкового числа, розподіленого нормально $N(\mu,\sigma)$, використовується функція $normrnd(\mu,\sigma)$.

Ще одним видом випадкових блукань є польоти Леві, які характеризуються серією стрибків, обумовлених функцією щільності ймовірності з важкими хвостами, за рахунок яких ймовірність значних відхилень від середнього більша, ніж у нормального розподілу. Розподіл із важким хвостом – це розподіл, хвіст якого не можна "відрізати", тобто не можна знехтувати великими, але рідкісними подіями.

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

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

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

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

Оскільки ітераційний процес не може тривати нескінченно, потрібно вибрати відповідну умову його закінчення – критерій збіжності ітерацій, при виконанні якого, останнє знайдене ітераційне наближення буде прийняте за шукане рішення [10, 11].

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

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

б) досягнуто граничних значень

1) ітерацій

$$iter > iter\_limit; \tag{2.5}$$

2) звернень до цільової функції

$$func\_call > func\_limit; \tag{2.6}$$

2) часу

$$time > time\_limit. \tag{2.7}$$

в) стагнація пошуку:

1) найкраще рішення не змінюється досить довго;

2) всі агенти знаходяться у своїх нішах, тобто немає покращень;

3) всі агенти знаходяться на дуже близькій відстані *eps* від найкращого рішення (вихід на плато);

4) середня якість популяції не змінюється досить довго.

Слід зазначити, що ні один метод або клас методів не відрізняється високою ефективністю при вирішенні оптимізаційних задач різних типів.

2.2 Алгоритм штучної бджолиної колонії

Алгоритм штучної бджолиної колонії (artificial bee colony algorithm), запропонований Д. Карабогою у 2005 році [3-6], моделює особливості поведінки медоносних бджіл під час пошуку джерел їжі (нектару на квітці). Серед усіх бджіл виділяють особини, які виконують пошук нектару, а саме: робочих бджіл або фуражирів (employed bees), бджіл-спостерігачів (onlooker bees) і бджіл-розвідників (scout bees). У початковий момент часу існує апріорна інформація про місцезнаходження джерел їжі. Вона зберігається таким чином, що до неї мають доступ усі особини.

Робочі бджоли направляються до джерел нектару (існуючих рішень) і проводять пошук в їх околі. Кожному джерелу відповідає одна робоча бджола. Джерело нектару характеризується значущістю. Якщо нове джерело їжі (нова точка на множині допустимих рішень) за певними параметрами (величина фітнес-функції) краще, ніж попереднє джерело, яке збережене в пам'яті бджоли, то вона запам’ятовує саме його. Потім робочі бджоли повертаються у вулик і передають бджолам-спостерігачам інформацію про нові, більш привабливі джерела, вербуючи їх. Кожний вулик має майданчик для танцю, на якому бджоли, які виявили джерела нектару, виконують виляючий танець (waggle dance), намагаючись залучити інших незайнятих бджіл летіти за ними, ”рекламуючи” джерело

їжі. Ті, в свою чергу, оцінюють всі знайдені джерела їжі й певним імовірнісним чином вибирають, з якого джерела починати свій пошук.

Після цього бджоли-спостерігачі проводять процес пошуку аналогічно тому, як це робили робочі бджоли. Нові рішення також проходять відбір і зберігаються тільки в тому випадку, якщо вони дозволяють покращити якість нектару (величину фітнес-функції). Якщо в процесі пошуку певне рішення не оновлюється протягом заданої кількості ітерацій, то воно вважається "забутим" (abandoned); бджола, до якої приписане дане джерело нектару, звільняється від роботи і стає бджолою-розвідником, тобто вибирає довільну точку в якості нового джерела нектару (рисунок 2.1) [19].

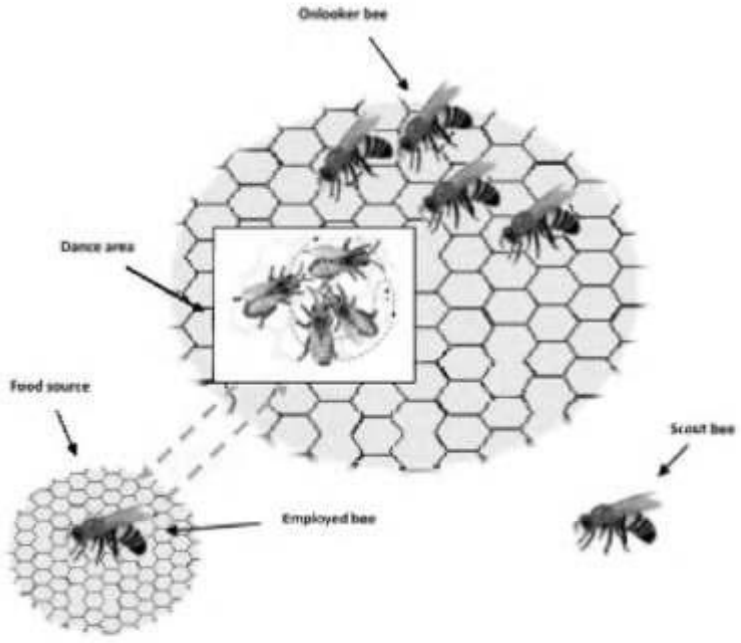

Рисунок 2.1 – Поведінка бджіл у природі

Розглядаємо задачу мінімізації функції. Алгоритм штучної бджолиної колонії включає в себе наступні кроки [3-6, 20-23]:

1. Ініціалізація параметрів алгоритму: LB, UB – нижня і верхня межі області допустимих значень D розмірності n, $UB > LB$; $X_i = (x_{i1},...,x_{in})^T$ – позиції джерел нектару, до яких летять $EBees$ робочих бджіл; $OBees \geq EBees$, $SBees << EBees$ – кількість бджіл-спостерігачів та бджіл-розвідників; $limit$ – рішення вважається забутим, якщо воно не змінювалося впродовж такої кількості ітерацій; K – максимальна кількість ітерацій алгоритму.

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

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

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

Подібну перевірку можна реалізувати програмно без циклу:

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

2. Для кожної робочої бджоли знаходяться координати її положення:

$$X_i = LB + rand \otimes (UB - LB), i = \overline{1, EBees}, \tag{2.10}$$

де *rand* – $(n\times 1)$-вектор незалежних дійсних випадкових чисел, рівномірно розподілених на інтервалі [0, 1]; *LB*, *UB* – дійсні числа, межі інтервалу пошуку для кожної координати, $UB > LB$; $\otimes$ – поелементне множення.

3. Для всіх отриманих рішень (робочих бджіл *EBees*) оцінюються джерела їжі – значення цільової функції $f(X_i)$, $i=\overline{1,EBees}$.

4. Отримати нові рішення для робочих бджіл *EBees*:

$$X_{new\_i} = X_i + U(a,b)\otimes(X_i - X_r),\ i=\overline{1,EBees}, \qquad (2.11)$$

де $U(a, b)$ – $(n\times 1)$-вектор незалежних дійсних випадкових чисел, рівномірно розподілених на інтервалі $[a, b]$; $a = -1$; $b = 1$; $b > a$; індекс r вибирається випадково з множини $\{1, ..., EBees\} / \{i\}$, тобто $r \neq i$; $\otimes$ – поелементне множення.

5. Розрахувати нове значення цільової функції $f(X_{new\_i})$.

6. Провести селекцію серед нових X_{new_i} та існуючих X_i рішень ($i=\overline{1,EBees}$):

$$X_i = \begin{cases} X_{new\_i}, \text{ якщо } f(X_{new\_i}) \le f(X_i); \\ X_i, \quad \text{інакше.} \end{cases} \qquad (2.12)$$

7. Розрахувати ймовірність вибору p_i джерел їжі X_i бджолами-спостерігачами ($i=\overline{1,EBees}$). Зазвичай використовують колесо рулетки (roulette wheel selection) [24-26].

Пропорційний відбір (proportional selection). Імовірність вибору p_i бджолою-спостерігачем джерела їжі виявляється пропорційною значенню фітнес-функції, тобто кожному джерелу їжі відповідає сектор рулетки:

$$p_i = \frac{F(X_i)}{\sum_{i=1}^{EBees} F(X_i)}, \tag{2.13}$$

де $F(.)$ – функція, яку підбирають залежно від задачі (максимум, мінімум).

Наприклад, для задачі на мінімум та максимум відповідно можна використати такі функції:

$$p_i = \frac{\exp\left(\frac{-f(X_i)}{f_{mean}}\right)}{\sum_{i=1}^{EBees} \exp\left(\frac{-f(X_i)}{f_{mean}}\right)}, \tag{2.14}$$

$$p_i = \frac{\exp\left(\frac{f(X_i)}{f_{mean}}\right)}{\sum_{i=1}^{EBees} \exp\left(\frac{f(X_i)}{f_{mean}}\right)}, \tag{2.15}$$

де $f_{mean} = \frac{\sum_{i=1}^{EBees} f(X_i)}{EBees}$ – середнє значення якості джерел їжі на даній ітерації.

Слабка сторона цього підходу у тому, що особини з дуже малим значенням функції пристосованості $f(X_i)$ мало досліджуються, що може призвести до передчасної збіжності алгоритму. Тому використовують масштабування оцінок значень цільової функції або ранжування.

Лінійне ранжування (linear ranking), в якому використовується тільки номер джерела їжі в упорядкованому списку, тоді ймовірність вибору можна описати формулою [12]

$$p_i = \frac{1}{EBees} \cdot \left(a - (a-b) \cdot \frac{i-1}{EBees-1} \right), \tag{2.16}$$

де $1 \le a \le 2$ – вибирається випадково; $b = 2 - a$; i – номер джерела.

Також можна застосовувати рівномірне ранжування (uniform ranking), яке задається наступною формулою [12]

$$p_i = \begin{cases} \dfrac{1}{\mu}, & якщо\ 1 \le i \le \mu; \\ 0, & якщо\ \mu < i \le EBees, \end{cases} \tag{2.17}$$

де $\mu \le EBees$ — параметр (фактично кількість найкращих агентів).

Слід зазначити, що попередньо необхідно відсортувати бджол за значенням цільової функції (за зростанням для задачі на мінімум або за спаданням для задачі на максимум), зберігши порядкові номера бджол до сортування. При цьому у відсортованому списку найбільшу ймовірність відбору отримає бджола з номером 1.

Сігма-відсікання (sigma truncation) виконує масштабування цільової функції за формулою [2]:

$$p_i = \frac{F_\sigma(X_i)}{\sum_{i=1}^{EBees} F_\sigma(X_i)}, \tag{2.18}$$

$$F_\sigma(X_i) = 1 + \frac{f(X_i) - f_{mean}}{2 \cdot RMSE}, \tag{2.19}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{EBees} (f(X_i) - f_{mean})^2}{EBees}}, \tag{2.20}$$

де $f_{mean} = \frac{\sum_{i=1}^{EBees} f(X_i)}{EBees}$ – середнє значення якості джерел їжі на даній ітерації.

Вибір джерела їжі можна ототожнити з вибором числа з інтервалу [*A*; *B*], де *A* та *B* позначають відповідно початок і кінець фрагмента кола. Отже, у цьому випадку вибір зводиться до вибору числа з інтервалу [0, 1] (відповідного сектору колеса рулетки), яке відповідає конкретній точці на колі (рисунок 2.2). Слід зазначити, що $\sum_{i=1}^{EBees} p_i = 1$.

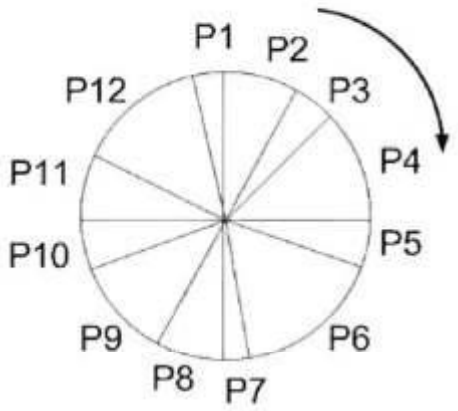

Рисунок 2.2 – Принцип роботи колеса рулетки

8. Отримати нові рішення для бджіл-спостерігачів *OBees* аналогічно рішенням робочих бджіл на кроці 4. Але індекс *r* вибирається на основі ймовірностей *p* з кроку 7. Для цього генерується випадкове число *rand* і перевіряється, в який сектор рулетки *r* воно потрапляє.

9. Повторити кроки 5 (оцінка джерел їжі) та 6 (селекція рішень).

10. Генерація нових джерел нектару замість "забутих". Для таких джерел випадково генерують нові рішення на множині допустимих рішень D аналогічно до кроку 2.

11. Видалити *SBees* найгірших рішень і згенерувати нові аналогічно до кроку 2.

12. Перевіряємо критерій зупинки пошуку (час, кількість ітерацій, стагнація пошуку і т.д.).

13. Якщо умова на кроці 12 не виконана, то визначити найкращого агента s^{best} та повернутися до кроку 4; інакше – перейти на крок 14.

14. Визначити та вивести глобально найкраще рішення $f^{best}(X^{best})$ для бджоли s^{best}.

Однією з основних проблем розробки метаевристик є забезпечення балансу між інтенсивністю і широтою пошуку. Інтенсифікація пошуку вимагає швидкої збіжності алгоритму, що означає швидке зменшення різноманітності популяції. А диверсифікація пошуку навпаки дозволяє забезпечити більш широкий огляд простору пошуку і більш високу ймовірність локалізувати глобальний екстремум задачі, тобто вона вимагає збереження різноманітності популяції якомога більшу кількість поколінь.

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

2.3 Модифікація алгоритму

Щоб підвищити ефективність у багатовимірному випадку запропоновано нову модель поведінки бджіл-спостерігачів. У канонічному варіанті алгоритму за допомогою колеса рулетки вибираються випадково бджоли, які здійснюють локальний пошук в околі певного джерела їжі. У даному випадку використовується лінійне ранжування, яке дозволяє більш детально досліджувати перспективні області пошуку, оскільки такі ділянки зацікавлять більше бджіл-спостерігачів. Приклад ранжування наведено у таблиці 2.1, а відповідна гістограма представлена на рисунку 2.3.

Таблиця 2.1 – Приклад лінійного ранжування

№	1	2	3	4	5	6	7	8	9	10	11
$f(X)$	2	1.8	1.6	1.4	1.2	1.0	0.8	0.7	0.3	0.2	0.0
p_i	0.155	0.142	0.129	0.116	0.104	0.091	0.078	0.065	0.053	0,04	0,027

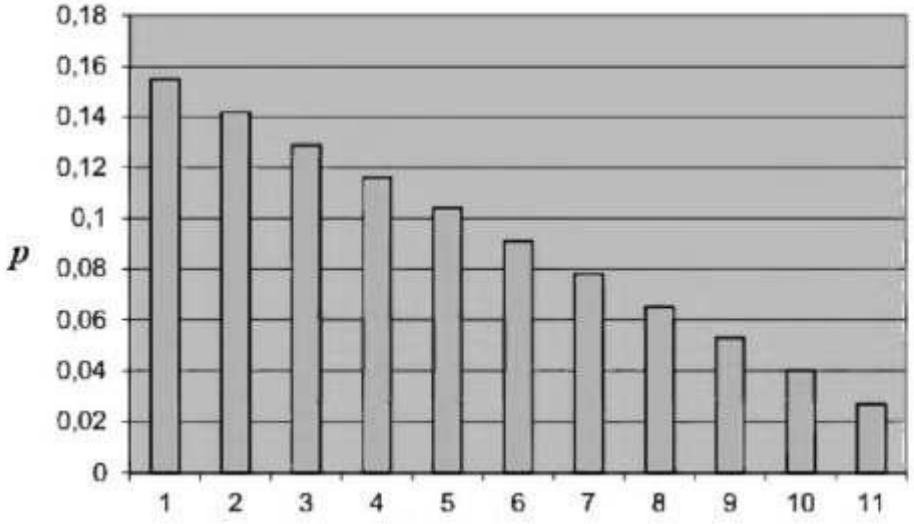

Рисунок 2.3 – Гістограма для лінійного ранжування

Також застосовано нову формулу для пошуку кращих джерел їжі бджолами-спостерігачами. Для активного пошуку використано ймовірнісну формулу вибору просторових координат для зміни [10, 11].

Нехай задано цільову функцію $f(x)=f(x_1,...,x_m)$, яку потрібно оптимізувати (знайти мінімум). Відповідно на будь-якій ітерації i існує m проектних змінних ($j=\overline{1,m}$), які обмежені нижньою LB і верхньою UB межею області допустимих значень D ($UB > LB$), при цьому розмір популяції n алгоритму представляє кількість варіантів розв'язків ($k=\overline{1,n}$), тобто вектор $X(k)=(x_1(k),...,x_m(k))^T$. Найкращий агент популяції *best* отримує найкраще поточне значення $f(X)$, а найгірший агент *worst* визначає найгірше поточне значення $f(X)$. Тоді для пошуку значень проектних змінних x_j для k-ого агента популяції на $(i + 1)$ ітерації алгоритму оптимізації можна застосувати наступну міграційну формулу:

$$\begin{aligned} x_j^{(i+1)}(k) &= x_j^{(i)}(k) + R_1 + R_2 = \\ &= x_j^{(i)}(k) + l_j \cdot (rand_{1j}^{(i)} \cdot (x_j^{(i)}(best) - |x_j^{(i)}(k)|) \\ &\quad - rand_{2j}^{(i)} \cdot (x_j^{(i)}(worst) - |x_j^{(i)}(k)|)), \end{aligned} \tag{2.21}$$

де $rand_1$, $rand_2$ – незалежні дійсні випадкові числа, рівномірно розподілені на інтервалі [0, 1]; R_1 , R_2 – параметри, які задають тенденцію наближення до найкращого та віддалення від найгіршого рішення; l_j – логічна змінна, яка визначає зміну заданої координати.

Перевагами алгоритму є простота обчислень, збіжність до оптимуму та можливість розпаралелювання; недоліки – складність теоретичного аналізу; невідомий час збіжності алгоритму. Такі удосконалення

дозволяють виконати ретельний аналіз області пошуку, оскільки алгоритм має стохастичну природу.

2.4 Висновки

У даному розділі описано стратегію пошуку оптимуму з використанням ройового інтелекту, сформульовано основні принципи класичного алгоритму штучної бджолиної колонії. Також представлено його модифіковану математичну модель для розв'язання оптимізаційних задач з використанням нової поведінки бджіл-спостерігачів. У наступному розділі будуть проведені експериментальні дослідження ефективності роботи модифікації з використанням тестових функцій та комплексної задачі з умовами.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

3.1 Вибір мови програмування та опис програми

MatLab представляє собою високорівневу мову інженерних і математичних розрахунків, інтерактивне середовище розробки алгоритмів і сучасний інструмент аналізу даних, який порівняно з класичними мовами програмування типу C++, Java дає можливість зменшити час розв'язання задач та спрощує розробку алгоритмів. Це потужний інструмент для вирішення низки наукових і прикладних задач, в таких галузях як: розробка систем управління, моделювання об'єктів, проектування інформаційно-комунікаційних систем, оброблення сигналів і зображень, обчислювальна біологія, фінансове моделювання тощо. Слово MatLab означає матрична лабораторія (MATrix LABoratory), а це говорить про те, що всі операції виконуються з використанням матричних обчислень. Обширна бібліотека функцій спрощує роботу (зокрема графічне відображення даних). Система MatLab складається з мови M-language; середовища MatLab; керованої графіки; бібліотеки математичних функцій; програмного інтерфейсу (рисунок 3.1) [18].

Тому програмне забезпечення для розв'язання поставленої задачі буде розроблятися саме на цій мові. Робота з програмою починається з вибору необхідних даних в інтерфейсі.

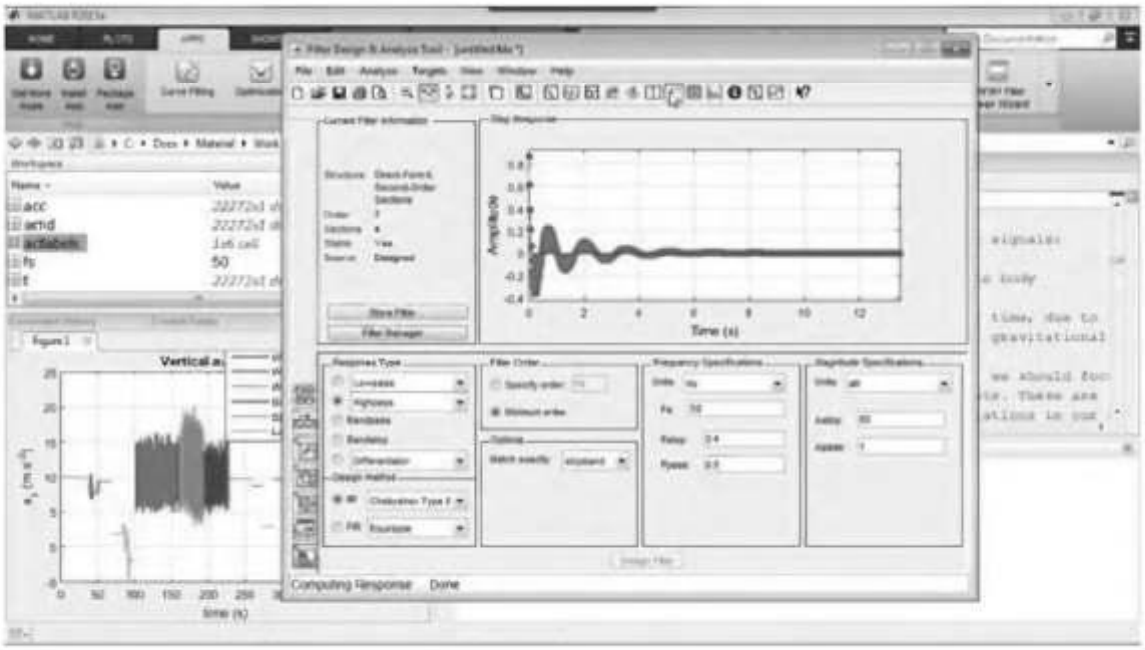

Рисунок 3.1 – Вигляд системи MatLab

Головне вікно програми складається з пунктів:

- вибору тестового прикладу (Select function);
- значень параметрів алгоритму (розміру колонії Bee Num; кількості підходів Abandoned, після яких рішення вважається забутим; кількості ітерацій Max Cycle);
- розмірності Param Num D та меж Lower Bound, Upper Bound простору пошуку D; а також кількості тестових запусків Runtime;
- кнопок управління; панелей виведення.

При натисканні на кнопку "Run" програма проводить розрахунок і зберігає кінцеві результати: вектор аргументів X^* і значення $f(X^*)$, а також буде показано кінцеву популяцію бджіл та виділено найкращу особину колонії. Результати виконання кожної ітерації заносяться в таблицю, яку можна викликати натисканням кнопки " Details". Для "очищення" робочого поля слід натиснути на кнопку "Clear" (рисунок 3.2).

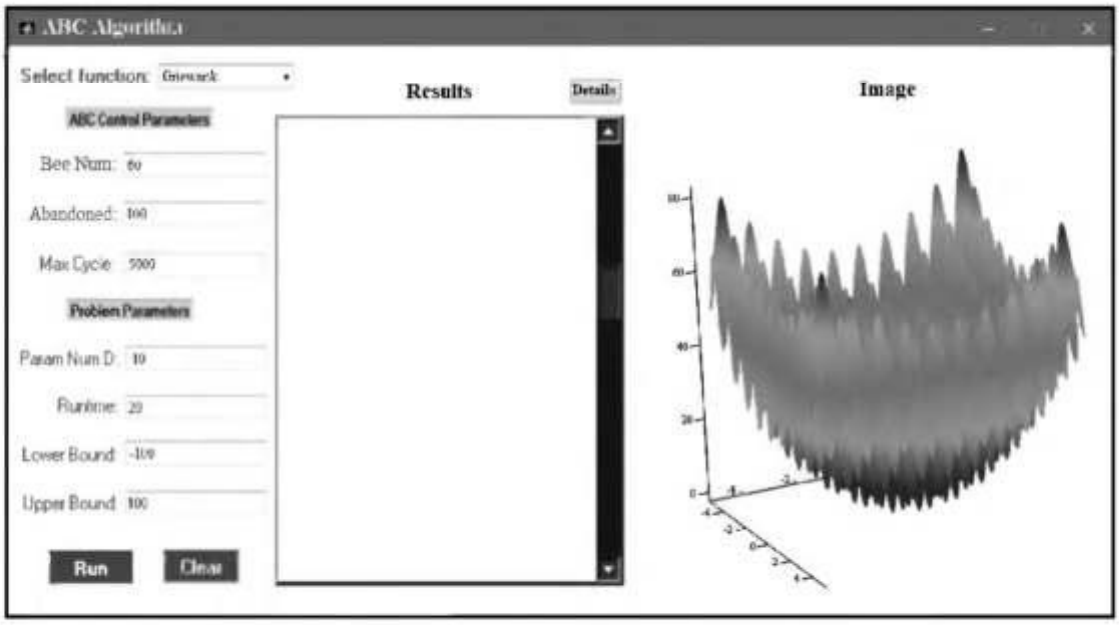

Рисунок 3.2 – Інтерфейс користувача програми

3.2 Результати експериментів

У ході дослідження роботи алгоритму проводились серії із 50 розв'язків однієї і тієї ж задачі при розмірності простору $n = 10$ з одними і тими ж значеннями параметрів [3-6].

1. Функція Растрігіна (рисунок 3.3). Функція зі складним рельєфом. Вона грунтується на функції "еліптичний параболоїд" з додаванням косинусної модуляції для створення багатьох локальних мінімумів, розміщених регулярно. Глобальний мінімум функції $f(X_i^*) = f(0,\ldots,0) = 0$.

$$f(x_1,\ldots,x_n) = 10 \cdot n + \sum_{j=1}^{n} (x_j^2 - 10\cos(2\pi x_j)),\ x_j \in [-5; 5]. \qquad (3.1)$$

У таблиці 3.1 проведено досліди по підбору параметрів модифікованого алгоритму штучної бджолиної колонії для даної функції.

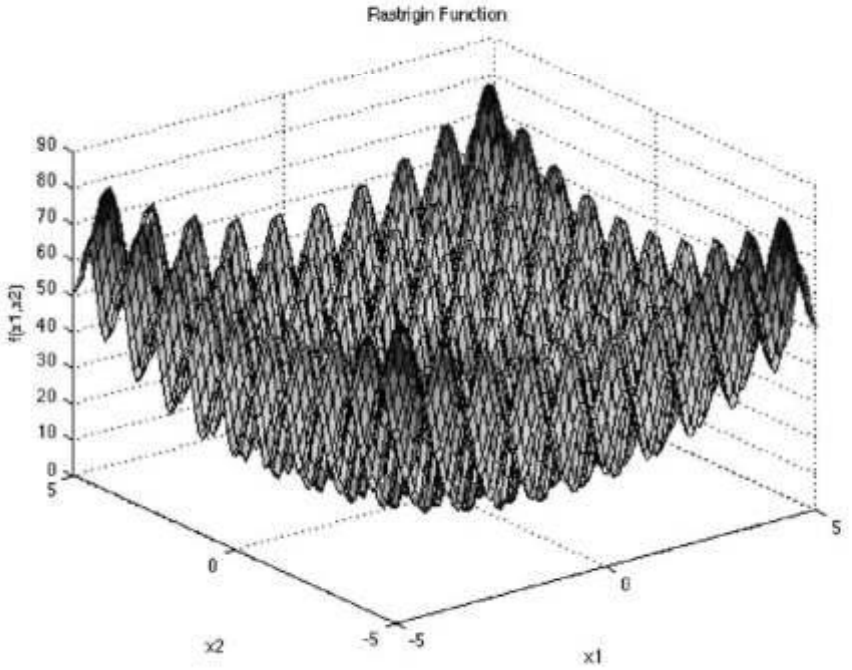

Рисунок 3.3 – Графічне відображення функції 1

Таблиця 3.1 – Результати обчислень для функції 1

Параметри модифікованого алгоритму				$f_{mean} = \frac{1}{50}\sum_{i=1}^{50} f_i$	f_{min}
EBees = OBees	*SBees*	*limit*	*K*		
10	3	5	30	0,774036163	0,000191140
40	3	20	100	0,055502399	0
5	3	10	100	2,865435970	$5{,}284235 \cdot 10^{-6}$
100	3	10	100	$1{,}200195 \cdot 10^{-12}$	$1{,}776357 \cdot 10^{-15}$
50	3	10	500	0	0

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

$$f(x_1,\ldots,x_n) = 20 + e - 20 \cdot \exp\left(\frac{1}{5} \cdot \sqrt{\frac{1}{n}\sum_{j=1}^{n} x_j^2}\right) - \\ - \exp\left(\frac{1}{n} \cdot \sum_{j=1}^{n} \cos(2\pi x_j)\right), x_i \in [-20;20]. \quad (3.2)$$

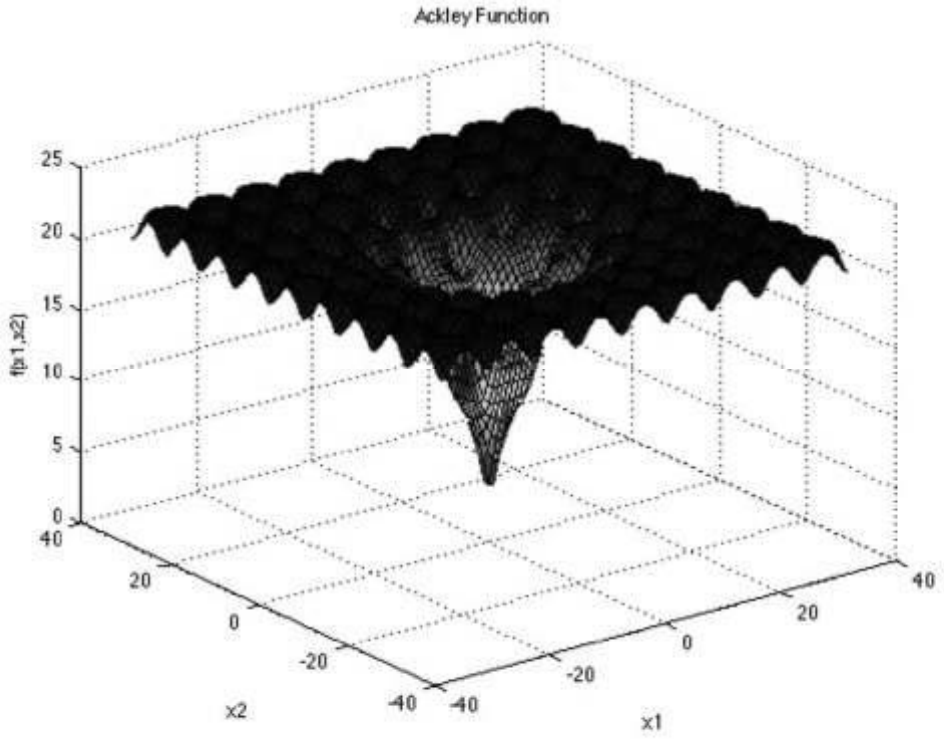

Рисунок 3.4 – Графічне відображення функції 2

У таблиці 3.2 наведено результати підбору параметрів модифікованого алгоритму штучної бджолиної колонії для даної функції.

3. Функція трьохгорбого верблюда (рисунок 3.5). Складна двовимірна функція з декількома локальними та одним глобальним мінімумами $f(X_i^*) = f(0,\ldots,0) = 0$.

$$f(x_1,x_2) = 2 \cdot x_1^2 - 1.05 \cdot x_1^4 + \frac{x_1^6}{6} + x_1 \cdot x_2 + x_2^2, x_i \in [-5;5]. \quad (3.3)$$

Таблиця 3.2 – Результати обчислень для функції 2

Параметри модифікованого алгоритму				$f_{mean} = \frac{1}{50}\sum_{i=1}^{50} f_i$	f_{min}
EBees = OBees	*SBees*	*limit*	*K*		
10	3	5	30	0,191223022	0,000342758
40	3	20	100	$5{,}995204 \cdot 10^{-16}$	0
5	3	10	100	2,532882839	$1{,}942374 \cdot 10^{-6}$
100	3	10	100	$6{,}319690 \cdot 10^{-9}$	$3{,}390865 \cdot 10^{-10}$
50	3	10	500	0	0

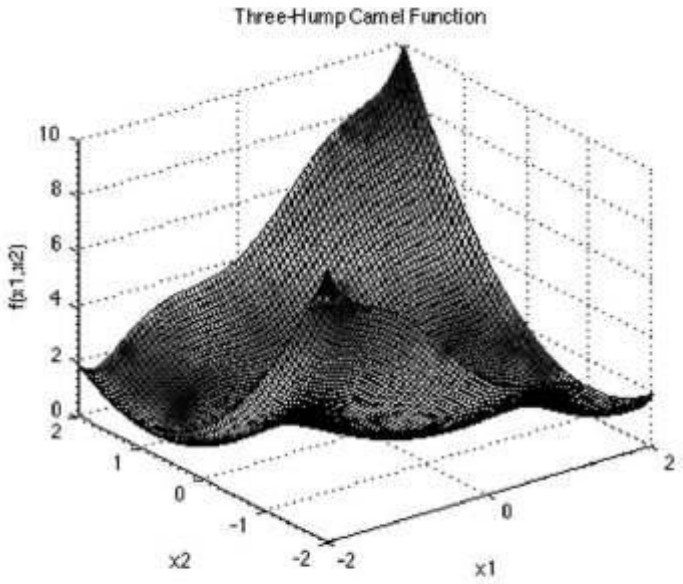

Рисунок 3.5 – Графічне відображення функції 3

У таблиці 3.3 показано результати підбору параметрів модифікованого алгоритму для цієї функції.

Оскільки алгоритм ефективно використовує експлуатаційний і дослідницький процеси для забезпечення достатньої різноманітності в популяції колонії, то не потрібно задавати їй надвеликий розмір.

Таблиця 3.3 – Результати обчислень для функції 3

Параметри модифікованого алгоритму				$f_{mean} = \frac{1}{50}\sum_{i=1}^{50} f_i$	f_{min}
EBees = *OBees*	*SBees*	*limit*	*K*		
10	3	5	30	0,003087826	$2{,}507842 \cdot 10^{-6}$
40	3	20	100	$7{,}103724 \cdot 10^{-24}$	$8{,}199139 \cdot 10^{-29}$
5	3	10	100	0,086802711	$1{,}299004 \cdot 10^{-12}$
100	3	10	100	$1{,}204724 \cdot 10^{-15}$	$3{,}792854 \cdot 10^{-18}$
50	3	10	500	$8{,}497420 \cdot 10^{-33}$	$7{,}806935 \cdot 10^{-41}$

Значення *limit* не повинно бути занадто великим, щоб гарантувати оновлення популяції. Але при малій кількості джерел нектару (малій величині *EBees*) значення *limit* все ж таки доцільно вибирати більшим, щоб не "загубилися" значення, які могли б бути покращені надалі.

Через те, що кількість параметрів у алгоритмі штучної бджолиної колонії досить мала, то до вибору значення максимальної кількості ітерацій *K* потрібно ставитися акуратно.

Використовуючи найкращі отримані параметри {*EBees* = *OBees* = 50, *SBees* = 3, *limit* = 10, *K* = 500} було протестовано на цих же тестових функціональних залежностях на 50 запусках програм модифікований та базовий алгоритми при розмірності простору пошуку n = 10. Результати порівняння відображено у таблиці 3.4.

Для ілюстрації практичного застосування модифікованого алгоритму розглянемо інженерну задачу умовної нелінійної змішано-цілочисленної оптимізації, а саме розподілу надійності-надлишковості системи

(reliability-redundancy allocation problem) у ході управління захистом газової турбіни від розгону [27].

Таблиця 3.4 – Порівняння базового (Б) та модифікованого (М) алгоритмів

Функція	Алгоритм	$f_{mean} = \frac{1}{50}\sum_{i=1}^{50} f_i$	f_{min}
1	Б	0,012417220	0,000021776
	М	0	0
2	Б	$1{,}547554 \cdot 10^{-7}$	$1{,}546805 \cdot 10^{-8}$
	М	0	0
3	Б	$9{,}376635 \cdot 10^{-16}$	$4{,}057161 \cdot 10^{-20}$
	М	$8{,}497420 \cdot 10^{-33}$	$7{,}806935 \cdot 10^{-41}$

Оскільки газова турбіна працює під високою температурою, то за нею необхідний постійний ретельний контроль. Зазвичай аварійне перезавантаження системи розроблено незалежно від контролю над швидкістю. Захист від перевищення швидкості газової турбіни дуже важливий у системах керування, оскільки дозволяє відновити для неї стаціонарний стан. Отже, слід враховувати високу надійність роботи регулюючих клапанів, яких на практиці використовуються від чотирьох.

Ефективність роботи з максимальною надійністю буде головною метою для регулюючих клапанів. Кожен з них дозволяє паливу легко проходити через вузький канал. У нормальному режимі роботи регулюючі клапани відкриваються послідовно. Виявлення перевищення швидкості постійно забезпечується електричними та механічними системами. При

перевищенні швидкості необхідно припинити подачу палива. Для цього регулюючі клапани ($Y_1 - Y_m$) повинні закритися (рисунок 3.6). Регулюючий клапан (підсистема) i містить n_i паралельних контролерів, кожен з яких має однакову надійність r_i.

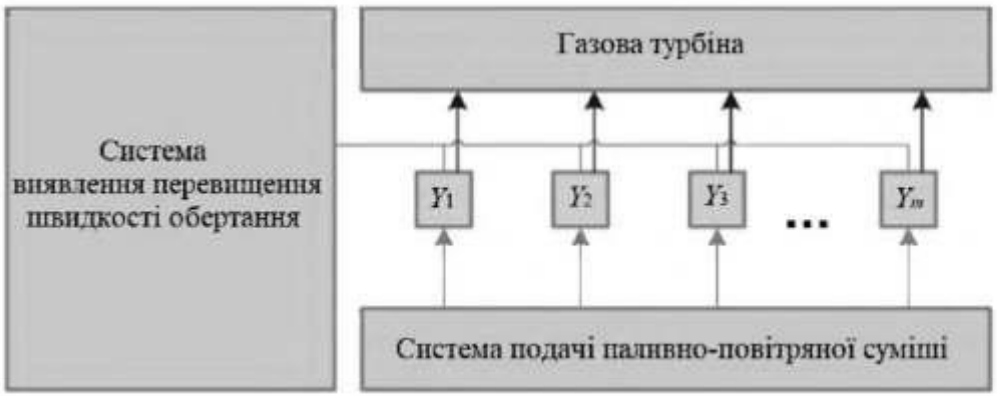

Рисунок 3.6 – Структура системи захисту газової турбіни від розгону

Під надійністю підсистеми розуміється ймовірність її безвідмовної роботи впродовж певного проміжку часу за умови працездатності системи в початковий момент часу.

Типовим підходом до досягнення вищої (максимальної) надійності системи $R_s = f(r,n)$ є підвищення надійності $r = (r_1,...,r_m)^T$ компонентів системи та збільшення кількості резервних компонентів $n = (n_1,...,n_m)^T$ у різних підсистемах при обмеженнях [27]

$$\begin{aligned} &g(r,n) \le l, \\ &0 \le r_i \le 1, n_i \in Z^+, 1 \le i \le m, \end{aligned} \tag{3.4}$$

де l – межа системних ресурсів; Z^+ – множина натуральних чисел; m – кількість підсистем.

Надійність однієї підсистеми задається виразом

$$R_i = 1 - \prod_{j=1}^{n_i} (1 - r_i) = 1 - (1 - r_i)^{n_i}. \tag{3.5}$$

Тоді цільову функцію задачі оптимізації для m підсистем сформулюємо у вигляді формули [27]

$$R_s = f(r,n) = \prod_{i=1}^{m} (1 - (1 - r_i)^{n_i}) \to \max, \tag{3.6}$$

а відповідні обмеження задані таким чином

$$\begin{cases} g_1(r,n) = \sum_{i=1}^{m} v_i^{\,2} \cdot n_i^{\,2} \le V; \\ g_2(r,n) = \sum_{i=1}^{m} C(r_i,T) \cdot (n_i + e^{0,25 \cdot n_i}) \le C; \\ g_3(r,n) = \sum_{i=1}^{m} w_i \cdot n_i \cdot e^{0,25 \cdot n_i} \le W; \\ 0.5 \le r_i \le 1 - 10^{-6}, r_i \in R^+; \\ 1 \le n_i \le 10, n_i \in Z^+; \\ 1 \le i \le m, m = 4, \end{cases} \tag{3.7}$$

де $g_1(r,n)$, $g_2(r,n)$, $g_3(r,n)$ – обмеження на об'єм, вартість системи та її вагу, а $r = (r_1,...,r_m)^T$, $n = (n_1,...,n_m)^T$; V, C, W – верхні межі об'єму системи, її вартості та ваги; v_i, w_i – об'єм i-ої підсистеми та її вага; r_i, n_i – надійність компонентів i-ої підсистеми та їх кількість; T – час безперебійної роботи компонентів (в годинах); $C(r_i,T) = \alpha_i \cdot \left(-\frac{T}{\ln r_i} \right)^{\beta_i}$ – функція вартості

кожного i-ого компонента з надійністю r_i в i-ій підсистемі; α_i, β_i – коефіцієнти; основні значення величин вказано в таблиці 3.5.

Таблиця 3.5 – Дані для інженерної задачі

i	$10^5 \cdot \alpha_i$	β_i	v_i	w_i	V	C	W	T
1	1.0	1.5	1	6	250	400	500	1000
2	2.3	1.5	2	6	-	-	-	-
3	0.3	1.5	3	8	-	-	-	-
4	2.3	1.5	2	7	-	-	-	-

Для знаходження безумовного мінімуму задачу умовної оптимізації необхідно перетворити. Для цього застосовано метод штрафів, ідея якого полягає в отриманні нової цільової функції $F(X)$, $X = (x_1,...,x_n)^T$ для області пошуку допустимих рішень D розмірності n за рахунок введення у цільову функцію $f(X)$ вибраної спеціальним чином функції штрафу $P(X)$. Для розв'язання представленої задачі використовуємо метод штрафів в адитивній формі [2]:

$$F(X) = f(X) + P(X) = f(X) + \sum_{k=1}^{4} \mu_k \cdot g_k^2(X) \cdot h_k \to \min, \qquad (3.8)$$

де μ_k $(k = \overline{1,4})$ – константи штрафу; $h_k = \begin{cases} 1, \text{ if } g_k(X) > 0; \\ 0, \text{ if } g_k(X) \le 0. \end{cases}$

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

показано в таблиці 3.6 після проведення 40 запусків програми, причому параметри $\{EBees = OBees = 100, SBees = 3, limit = 400, K = 5000\}$, а всі умови g_i повинні бути виконані.

Таблиця 3.6 – Результати порівняння методів

Автори	Алгоритм	n	r	R_s
Т. Йокота та інші [28]	Генетичний алгоритм	3, 6, 3, 5	0,965539, 0,760592, 0,972646, 0,804660	0,999468
Т. С. Чен [29]	Штучна імунна система	5, 5, 5, 5	0,903800, 0,874992, 0,919898, 0,890609	0,999942
Л. С. Коельо [30]	Алгоритм рою часток	5, 6, 4, 5	0,902231, 0,856325, 0,948145, 0,883156	0,999953
В. С. Йе, Т. Дж. Хсіе [31]	Алгоритм штучної бджолиної колонії	5, 6, 4, 5	0,901614, 0,849920, 0,948143, 0,888223	0,999955
У даній МКР	Модифікований алгоритм штучної бджолиної колонії	5, 5, 4, 7	0,903354, 0,886875, 0,954590, 0,817874	0,999962

Запропонований алгоритм перевершує розглянуті у таблиці підходи в плані надійності системи. Результати експериментів показали, що модифікований алгоритм штучної бджолиної колонії більш ефективний для розв'язання задач глобальної оптимізації у порівнянні з базовим варіантом. Він характеризується більш високою швидкістю збіжності і точністю знаходження оптимального значення.

3.3 Висновки

У даному розділі описано принципи роботи розробленого програмного забезпечення. Програма дозволяє керувати параметрами модифікованого алгоритму й аналізувати ефективність його роботи. Визначено, що модифікований алгоритм стабільний і збігається до оптимуму, якщо вдало вибрати ключові параметри. Алгоритм штучної бджолиної колонії показав свою конкурентоспроможність і можливість до подальшого розвитку.

4 РОЗДІЛ ЕКОНОМІКИ

4.1 Технологічний аудит модифікованого алгоритму оптимізації

Метою проведення комерційного і технологічного аудиту дослідження за темою «Модифікований алгоритм штучної бджолиної колонії для розв'язання задач оптимізації» є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в таблиці 4.1 [32]. Для встановлення потенційного рівня комерційного використання отриманих під час проведення наукових досліджень результатів проведемо технологічний аудит нашої розробки.

Таблиця 4.1 – Критерії оцінювання рівня комерційного потенціалу розробки та їх бальна оцінка

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

Продовження таблиці 4.1

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

Продовження таблиці 4.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	1	2	3	4	5
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки пові-домлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Для цього було запрошено 3-ох експертів: д.т.н., професора Квєтного Р.Н. (1), к.т.н., доцента Богач І.В. (2) і к.т.н., доцента Кабачія В.В. (3), які є відомими спеціалістами з дослідження цієї задачі, авторами численних наукових праць з моделювання та оптимізації складних систем різного функціонального призначення.

Рівень комерційного потенціалу розробленого алгоритму можна встановити, керуючись таблицею 4.2 [32]. Експерти оцінили розроблений метод так, як це наведено в таблиці 4.3. Середньоарифметична сума балів $\overline{СБ}$ експертів становила:

$$\overline{СБ} = \frac{\sum_{1}^{3} Б_i}{3} = \frac{43+44+41}{3} = \frac{128}{3} = 42,66 .$$

Можна зробити висновок, що розроблений алгоритм має технічний рівень та комерційний потенціал, який вважається "високим".

Таблиця 4.2 – Рівні наукового та комерційного потенціалу розробки

Середньоарифметична сума балів, розрахована на основі висновків експертів	Рівень наукового та комерційного потенціалу розробки
0 – 10	Низький
11 – 20	Нижче середнього
21 – 30	Середній
31 – 40	Вище середнього
41 – 48	Високий

Таблиця 4.3 – Результати оцінювання комерційного потенціалу розробки

Критерії	Експерт		
	1	2	3
	Бали, виставлені експертами:		
1	4	4	4
2	4	3	3
3	4	4	3
4	4	4	4
5	4	4	4
6	4	3	3
7	3	4	3
8	3	3	3
9	4	4	4
10	3	4	3
11	3	3	4
12	3	4	3
Сума балів	$СБ_1 = 43$	$СБ_2 = 44$	$СБ_3 = 41$

4.2 Розрахунок витрат на розробку та проведення досліджень

При виконанні магістерської кваліфікаційної роботи було зроблено такі витрати [33]:

1. Витрати на основну заробітну плату дослідників розраховуємо у відповідності до посадових окладів працівників, за формулою:

$$З_о = \sum_{i=1}^{k} \frac{M_{mi} \cdot t_i}{T_p} = 14760{,}00 \cdot 22 / 22 = 14760{,}00 \; грн, \quad (4.1)$$

де k – кількість посад дослідників залучених до процесу досліджень; M_{mi} – місячний посадовий оклад конкретного дослідника, грн; t_i – кількість днів роботи конкретного дослідника, дні; T_p – середня кількість робочих днів в місяці, T_p = 22 дні.

Проведені розрахунки зведемо до таблиці 4.4.

Таблиця 4.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Кількість днів роботи	Витрати на заробітну плату, грн.
Науковий керівник	14760,00	670,91	22	14760,00
Інженер-програміст	14550,00	661,36	22	14550,00
Консультант з проблем оптимізації процесів	14500,00	659,09	5	3295,45
Технік	7300,00	331,82	20	6636,36
Всього				39241,82

2. Витрати на основну заробітну плату робітників за відповідними найменуваннями робіт НДР розраховуємо за формулою:

$$З_p = \sum_{i=1}^{n} C_i \cdot t_i, \qquad (4.2)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн./год.; t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}}, \qquad (4.3)$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо $M_м$ = 6700,00 грн.; K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду; K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати; T_p – середнє число робочих днів в місяці, приблизно T_p = 22 дні; $t_{зм}$ – тривалість зміни, год.

Результати запишемо в таблицю 4.5. Тоді маємо:

$$C_1 = 6700{,}00 \cdot 1{,}10 \cdot 1{,}65 / (22 \cdot 8) = 69{,}09 \text{ грн.}$$

$$З_{p1} = 69{,}09 \cdot 8{,}50 = 587{,}30 \text{ грн.}$$

3. Додаткову заробітну плату розраховуємо як 10…12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%} = (39241{,}82 + 4105{,}74) \cdot 10 / 100\% = 4334{,}76 \text{ грн.}, \quad (4.4)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати; приймемо 10%.

Таблиця 4.5 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год.	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн.	Величина оплати на робітника грн.
Підготовка робочого місця розробника	8,50	2	1,10	69,09	587,30
Інсталяція програмного забезпечення моделювання	6,90	3	1,35	84,80	585,10
Введення програмних блоків	6,20	4	1,50	94,22	584,16
Налагодження програмних блоків	9,00	5	1,70	106,78	961,03
Тестування системи	13,00	5	1,70	106,78	1388,16
Всього					4105,74

4. Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} = (39241{,}82 + 4105{,}74 + 4334{,}76) \cdot \frac{22}{100} = 10490{,}11 \text{ грн.}, \quad (4.5)$$

де $Н_{зп}$ – норма нарахування на заробітну плату; приймаємо 22%.

5. Витрати на матеріали розраховуються окремо по кожному виду матеріалів за формулою:

$$M = \sum_{j=1}^{n} H_j \cdot Ц_j \cdot K_j - \sum_{j=1}^{n} B_j \cdot Ц_{вj} = 3{,}0 \cdot 268{,}00 \cdot 1{,}12 - 0 \cdot 0 = 900{,}48 \text{ грн.}, \quad (4.6)$$

де H_j – норма витрат матеріалу j-го найменування, кг; n – кількість видів матеріалів; $Ц_j$ – вартість матеріалу j-го найменування, грн./кг; K_j – коефіцієнт транспортних витрат, (K_j = 1,1 … 1,15); B_j – маса відходів j-го найменування, кг; $Ц_{вj}$ – вартість відходів j-го найменування, грн./кг.

Проведені розрахунки зведемо до таблиці 4.6.

Таблиця 4.6 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за одиницю, грн.	Норма витрат	Величина відходів, кг	Ціна відходів, грн./кг	Вартість витраченого матеріалу, грн.
Офісний папір А4 500	268,00	3,0	-	-	900,48
Папір для записів А4 250	146,00	5,0	-	-	817,60
Органайзер офісний	163,00	2,0	-	-	365,12
Набір офісний	203,00	3,0	-	-	682,08
Картридж для принтера	986,00	1,0	-	-	1104,32
Диск оптичний	22,70	3,0	-	-	76,27
Flesh-пам'ять	615,00	1,0	-	-	688,80
Тека	108,00	4,0	-	-	483,84
Всього					5118,51

6. Витрати на комплектуючі, які використовують при проведенні НДР відсутні.

7. Балансову вартість спецустаткування розраховуємо за формулою:

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.і} \cdot K_i = 105890{,}00 \cdot 1 \cdot 1{,}12 = 118596{,}80 \text{ грн.}, \quad (4.7)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн.; $C_{пр.і}$ – кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.; K_i – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, K_i = 1,10…1,12; k – кількість найменувань устаткування.

Отримані результати зведемо до таблиці 4.7.

Таблиця 4.7 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн.	Вартість, грн.
Комп'ютерне обладнання для вирішення проблем оптимізації процесів EOM HP Z6 G4 WKS Tower / Xeon Silver 4108	1	105890,00	118596,80
Всього			118596,80

8. Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.і} \cdot K_i = 9860{,}00 \cdot 1 \cdot 1{,}11 = 10944{,}60 \text{ грн.}, \quad (4.8)$$

де $Ц_{інпг}$ – ціна придбання одиниці програмного засобу даного виду, грн.; $С_{прг.і}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.; $К_і$ – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, $К_і$ = 1,10…1,12; k – кількість найменувань програмних засобів.

Отримані результати зведемо до таблиці 4.8.

Таблиця 4.8 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт.	Ціна за одиницю, грн.	Вартість, грн.
Математичне середовище MatLab	1	9860,00	10944,60
Прикладне ПЗ Mathematica	1	7580,00	8413,80
Всього			19358,40

9. У спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12} = (18650{,}00 \cdot 1) / (3 \cdot 12) = 518{,}06 \ грн., \quad (4.9)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн.; $t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців; $T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

Проведені розрахунки зведемо до таблиці 4.9.

Таблиця 4.9 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн.
Персональний комп'ютер розробника ПЗ	18650,00	3	1	518,06
Робоче місце інженера-програміста	7840,00	5	1	130,67
Пристрої передачі даних	6700,00	4	1	139,58
Оргтехніка	9500,00	5	1	158,33
Приміщення лабораторії розробки ПЗ	245000,00	20	1	1020,83
ОС Windows	8320,00	3	1	231,11
Прикладний пакет Microsoft Office	7460,00	3	1	207,22
Середовище програмування Matlab	6900,00	3	1	191,67
Всього				2597,47

10. Витрати на силову електроенергію розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{ani}}{\eta_i} = 0{,}42 \cdot 160{,}0 \cdot 6{,}20 \cdot 0{,}95 / 0{,}97 = 416{,}64 \text{ грн.}, \quad (4.10)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт; t_i – тривалість роботи обладнання на етапі дослідження, год.; $Ц_e$ – вартість 1 кВт-години електроенергії, грн.; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_e$ = 6,20 грн.; K_{ani} – коефіцієнт, що враховує використання потужності, $K_{ani} < 1$; η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

Проведені розрахунки зведемо до таблиці 4.10.

Таблиця 4.10 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год.	Сума, грн.
Персональний комп'ютер розробника ПЗ	0,42	160,0	416,64
Комп'ютерне обладнання для вирішення проблем оптимізації процесів EOM HP Z6 G4 WKS Tower / Xeon Silver 4108 (6QP06EA)	0,36	120,0	267,84
Робоче місце інженера-програміста	0,12	120,0	89,28
Пристрої передачі даних	0,10	300,0	186,00
Оргтехніка	0,56	180,0	624,96
Всього			1584,72

11. Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників:

$$B_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%} = (39241{,}82 + 4105{,}74) \cdot 20 / 100\% = 8669{,}51 \text{ грн.}, \quad (4.11)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св} = 20\%$.

12. Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%} = (39241{,}82 + 4105{,}74) \cdot 35 / 100 = 15171{,}65 \text{ грн.}, \quad (4.12)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп} = 35\%$.

13. До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$І_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%} = (39241{,}82 + 4105{,}74) \cdot 60 / 100 = 26008{,}53 \text{ грн.}, \quad (4.13)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів} = 60\%$.

14. Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%} = (39241{,}82 + 4105{,}74) \cdot 110 / 100 = 47682{,}31 \text{ грн.}, \quad (4.14)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв} = 110\%$.

15. Витрати на проведення науково-дослідної роботи розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{зг} = З_о + З_р + З_{дод} + З_н + М + К_в + B_{спец} + B_{прг} + A_{обл} + B_е + B_{ос} + B_{он} + I_в + B_{нзв}, \quad (4.15)$$

$$B_{заг} = 39241{,}82 + 4105{,}74 + 4334{,}76 + 10490{,}11 + 5118{,}51 + 0{,}00 +$$
$$+ 118596{,}80 + 19358{,}40 + 2597{,}47 + 1584{,}72 + 8669{,}51 + 15171{,}65 +$$
$$+ 26008{,}53 + 47682{,}31 = 302960{,}33 \text{ грн.}$$

16. Загальні витрати на завершення науково-дослідної роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta} = 302960{,}33 / 0{,}9 = 336622{,}59 \text{ грн.}, \quad (4.16)$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо $\eta = 0{,}9$.

4.3 Прогнозування комерційного ефекту від можливої комерціалізації розробки

Під час проведення досліджень було встановлено, що сьогодні на ринку подібне програмне забезпечення коштує приблизно 110 тис. грн. Нашу розробку можна буде реалізовувати на ринку за 130 тис. грн., тобто на 20 тис. грн. дорожче.

Аналіз місткості ринку показує, що сьогодні в Україні кількість потенційних користувачів нашої розробки складає щороку приблизно 10 осіб, але цей попит буде зростати. Припустимо, що розробка буде користуватися попитом на ринку впродовж 3-ох років після впровадження.

Якщо розробка буде впроваджена з 1 січня 2023 року, то її результати виявляться впродовж 2023-го, 2024-го та 2025-го років. Прогноз зростання попиту на нашу розробку (відносно базового року) складає по роках:

1-й рік після впровадження (2023 р.) – +4 шт.;

2-й рік після впровадження (2024 р.) – +7 шт. (до попереднього року);

3-й рік після впровадження (2025 р.) – +10 шт. (до попереднього року).

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

$$\Delta П_i = (\pm \Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \qquad (4.17)$$

де $\Delta Ц_o$ – це є збільшення ціни нашого програмного забезпечення; $\Delta Ц_o$ = (130 - 110) = 20 тис. грн; N – обсяг діяльності у році до впровадження результатів розробки (N = 10 шт.); ΔN – покращення основного кількісного показника від впровадження результатів розробки. Таке покращення становитиме (відносно базового 2022 року) по роках, відповідно: +4, +7, +10 штук; $Ц_o$ – ціна нового програмного забезпечення; $Ц_o$ = 130 тис. грн; n – кількість років, впродовж яких очікується отримання позитивних результатів від впровадження розробки; n = 3; λ – коефіцієнт, який враховує сплату податку на додану вартість; $\lambda = 0{,}8333$;

ρ – коефіцієнт, який враховує рентабельність продукту. Рекомендується приймати $\rho = (0{,}2...0{,}5)$; візьмемо $\rho = 0{,}45$; υ – ставка податку на прибуток; $\upsilon = 18\%$.

Тоді зростання чистого прибутку $\Delta П_1$ для потенційного інвестора впродовж першого (2023) року може скласти:

$$\Delta П_1 = [20 \cdot 10 + 130 \cdot 4] \cdot 0{,}8333 \cdot 0{,}45 \cdot (1 - \frac{18}{100}) = 221{,}39 \ тис.\ грн.$$

Зростання чистого прибутку $\Delta П_2$ для потенційного інвестора впродовж другого (2024) року може скласти:

$$\Delta П_2 = [20 \cdot 10 + 130 \cdot 7] \cdot 0{,}8333 \cdot 0{,}45 \cdot (1 - \frac{18}{100}) = 341{,}31 \ тис.\ грн.$$

Зростання чистого прибутку $\Delta П_3$ для потенційного інвестора впродовж третього (2025) року може скласти:

$$\Delta П_3 = [20 \cdot 10 + 130 \cdot 10] \cdot 0{,}8333 \cdot 0{,}45 \cdot (1 - \frac{18}{100}) = 461{,}23 \ тис.\ грн.$$

Теперішню вартість інвестицій PV, які можуть бути вкладені в розроблене нами програмне забезпечення можна розрахувати за формулою:

$$PV = k_{інв} \cdot ЗВ \ грн., \qquad (4.18)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію; приймаємо $k_{інв} = 2,5$.

Для нашого випадку отримаємо:

$$PV = 2,5 \times 50 = 125 \text{ тис. грн.}$$

Далі розраховуємо абсолютний ефект від вкладених інвестицій:

$$E_{абс} = ПП - PV, \tag{4.19}$$

де $ПП$ – приведена вартість всіх можливих чистих прибутків від реалізації нашої розробки, грн.; PV – теперішня вартість інвестицій.

У свою чергу, приведена вартість всіх чистих прибутків розраховується за формулою [33]:

$$ПП = \sum_{t=1}^{T} \frac{\Delta П_t}{(1+\tau)^t}, \tag{4.20}$$

де $\Delta П_t$ – збільшення чистого прибутку у кожному з років, впродовж яких виявляються результати впровадження науково-технічної розробки, грн.; T – період часу, впродовж якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки; τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0,085$; t – період часу (в роках) від моменту початку впровадження розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

Тоді приведена вартість усіх можливих чистих прибутків, що їх може отримати потенційний інвестор від можливої реалізації розробки, складе:

$$ПП = \frac{221,39}{(1+0,085)^2} + \frac{341,31}{(1+0,085)^3} + \frac{461,23}{(1+0,085)^4} =$$

$$= 188,06 + 267,21 + 332,81 = 788,08 \approx 789 \text{ тис. грн.}$$

Абсолютний ефект від можливої комерціалізації розробки складе:

$$E_{абс} = 789 - 125 = 664 \text{ тис. грн.}$$

Далі розрахуємо відносну ефективність вкладених у комерціалізацію нашої розробки потенційних інвестицій:

$$E_в = \sqrt[Тэс]{1 + \frac{E_{абс}}{PV}} - 1 = (1 + 664 / 125)^{1/4} = 0,585. \qquad (4.21)$$

Мінімальна дохідність визначається за формулою:

$$\tau_{мін} = d + f = 0,10 + 0,45 = 0,55 \text{ або } \tau_{мін} = 55\%. \qquad (4.22)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; $d = 0,10$; f – ризикованість вкладення інвестицій, приймемо 0,45.

Оскільки величина $E_в = 58,5\% > \tau_{мін} = 52\%$, то потенційний інвестор у принципі може бути зацікавлений у комерціалізації нашої розробки.

Розраховуємо термін окупності коштів, які можуть бути вкладені у програмне забезпечення:

$$T_{ок} = \frac{1}{E_в} = 1 / 0{,}585 = 1{,}709 \text{ років.} \quad (4.23)$$

Оскільки $T_{ок} < 3$ років, то комерціалізація нашої розробки, тобто виведення її на ринок потенційним інвестором, є можливою.

4.4 Висновки

Аналіз комерційного потенціалу розробки показав, що вона за своїми характеристиками випереджає аналоги, що підтверджує її перспективність. Результати економічної частини магістерської кваліфікаційної роботи зведено у таблицю 4.11. Отже, основні техніко-економічні показники удосконаленого алгоритму, визначені у технічному завданні, виконані.

Таблиця 4.11 – Результати економічної частини

Показники	Задані у технічному завданні	Досягнуті у магістерській кваліфікаційній роботі	Висновок
1. Витрати на розробку	не більше 350 тис. грн.	336 тис. грн.	Виконано
2. Абсолютний ефект від впровадження	не менше 500 тис. грн.	664 тис. грн.	Виконано
3. Внутрішня дохідність інвестицій	не менше 50%	58,5 %	Виконано
4. Термін окупності потенційних інвестицій	до 3-ох років	1,709 років	Виконано

ВИСНОВКИ

У роботі розглядаються алгоритми ройового інтелекту. Визначено, що одним з найбільш ефективних та зручних є алгоритм штучної бджолиної колонії. Тому детально розглянуто його математичну модель та представлено модифікацію для розв'язання задач неперервної оптимізації. Також описано особливості розробленого програмного забезпечення, яке дозволяє керувати параметрами алгоритму та представляти результати графічно.

Тестування ефективності модифікованого алгоритму проведено на декількох функціях безумовної та умовної оптимізації. Визначено, що запропонована модифікація має більш високу швидкість збіжності і точність знаходження оптимуму порівняно з базовим варіантом на заданих задачах. Крім того, необхідно вдало підбирати параметри під кожну конкретну задачу. Таким чином, застосування алгоритму штучної бджолиної колонії не може гарантувати знаходження глобального екстремуму, але дає хороше субоптимальне рішення, не потребуючи громіздких обчислень.

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

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

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

2. Gendreau M., Potvin J.-Y. Handbook of Metaheuristics. Springer, 2010. 648 p.

3. Akay B., Karaboga D. Artificial Bee Colony Algorithm for Large-Scale Problems and Engineering Design Optimization. *Journal of intelligent manufacturing*. 2012. Vol. 23 (4). P. 1001-1014.

4. Karaboga D., Basturk B. A Powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm. *Journal of Global Optimization*. 2007. Vol. 39. P. 459-471.

5. Karaboga D., Basturk B. Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems. *Lecture Notes in Computer Science*. 2007. P. 789-798.

6. Karaboga D., Basturk B. On the Performance of Artificial Bee Colony (ABC) Algorithm. *Applied Soft Computing*. 2008. Vol. 8. P. 687-697.

7. Liu J.-L., Li C.-C. An Improved Artificial Bee Colony Algorithm Applied to Engineering Optimization Problems. *Journal of information science and engineering*. 2016. Vol. 32. P. 863-886.

8. Gao W.F., Liu S.Y., Huang L.L. A Global Best Artificial Bee Colony Algorithm for Global Optimization. *Journal of Computational and Applied Mathematics*. 2012. Vol. 236. P. 2741-2753.

9. Устенко С.В., Бібко О.О. Використання методів біоніки в інтелектуальних інформаційних системах. *Інформаційні системи та мережі*. Київ: Київський національний економічний університет імені В. Гетьмана. 2015. С. 501-508.

10. Ярська В.І., Саєнко А.П., Грицюк Б.П., Іванов Ю.Ю. Деякі аспекти роботи біоінспірованих алгоритмів розв'язання задачі оптимізації. *Науково-технічна конференція підрозділів ВНТУ*: матер. L науково-технічної конференції. Вінниця: ВНТУ, 2021. С. 1128-1129.

11. Ярська В.І., В.В. Коваль, Іванов Ю.Ю. та інші. Застосування стохастичних алгоритмів оптимізації з декомпозицією простору пошуку *Актуальні задачі медичної, біологічної фізики та інформатики*: матер. науково-практичної конференції з міжнародною участю. Вінниця: ВНМУ ім. М.І. Пирогова, 2022. С. 72-73.

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

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

14. Bonnans J.F., Gilbert J.C., Lemarechal C., Sagastizabal C.A. Numerical Optimization: Theoretical and Practical Aspects. Berlin, 1997. 491 p.

15. Кузьмін І.В., Биков М.М., Москвіна С.М. Методи оптимізації складних систем. Вінниця: ВДТУ, 2003. 165 с.

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

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

18. MathWorks – MATLAB. The Language of Technical Computing. USA: MathWorks. URL: http://mathworks.com/help/matlab/index.html (дата звернення 10.09.2022).

19. Teodorovic D. Bee Colony Optimization (BCO). URL: https://www.researchgate.net/publication/226530817_Bee_colony_optimization_BCO (дата звернення 15.09.2022).

20. Ram N. Implementing Artificial Bee Colony Algorithm to Solve Business Problems. URL: https://towardsdatascience.com/implementing-artificial-bee-colony-algorithm-to-solve-business-problems-cb754f3b9255 (дата звернення 15.09.2022).

21. Xiao S., Wang W., Tan D. and others. An Improved Artificial Bee Colony Algorithm Based on Elite Strategy and Dimension Learning. *Mathematics.* 2019. №7(289). 17 p. URL: https://www.researchgate.net/publication/331946579_An_Improved_Artificial_Bee_Colony_Algorithm_Based_on_Elite_Strategy_and_Dimension_Learning (дата звернення 15.10.2022).

22. Kang F., Li J., Li H. Artificial Bee Colony Algorithm and Pattern Search Hybridized for Global Optimization. *Applied Soft Computing.* 2013. Vol. 13. P. 1781-1791.

23. Kumar S., Kumar V.S., Kumari R. Improved Onlooker Bee Phase in Artificial Bee Colony Algorithm. *International Journal of Computer Applications*. 2014. Vol. 90 (6). P. 31-39.

24. Bullinaria J.A., AlYahya K. Artificial Bee Colony Training of Neural Networks: Comparison with Back-Propagation. *Memetic Computing.* 2014. URL: https://www.cs.bham.ac.uk/~jxb/PUBS/MC14.pdf (дата звернення 17.10.2022).

25. Pham D.T., Haj Darwish A., Eldukhr E.E. Optimization of a Fuzzy Logic Controller Using the Bees Algorithm. *International Journal of Computing Aided Engineering*. 2009. P. 250-264.

26. Порплиця Н.П. Порівняльний аналіз ефективності генетичного та "бджолиного" алгоритмів у задачі структурної ідентифікації інтервального різницевого оператора. *Інформаційні технології та комп'ютерна інженерія*. Львів, 2015. № 1. С. 55-67.

27. Floudas C.A., Pardalos P.M., Adjiman C.S. and others. Handbook of Test Problems in Local and Global Optimization. Kluwer Academic Publishers, 1999. 442 p.

28. Yokota T., Gen M., Li H.H. Genetic Algorithm for Nonlinear Mixed-Integer Programming Problems and Its Application. *Computers and Industrial Engineering*. 1996. Vol. 30 (4). P. 905-917.

29. Chen T.C. IAs Based Approach for Reliability and Redundancy Allocation Problems. *Applied Mathematics and Computation*. 2006. Vol. 182. P. 1556-1567.

30. Coelho L.S. An Efficient Particle Swarm Approach for Mixed-Integer Programming in Reliability-Redundancy Optimization Applications. *Reliability Engineering and System Safety*. 2009. Vol. 94 (4). P. 830-837.

31. Yeh W.C., Hsieh T.J. Solving Reliability Redundancy Allocation Problems Using an Artificial Bee Colony Algorithm. *Computers & Operations Research*. 2011. Vol. 38 (11). P. 1465-1473.

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

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

ДОДАТКИ

Додаток А

(обов'язковий)

Технічне завдання

ЗАТВЕРДЖУЮ

в. о. завідувача кафедри АІІТ

д.т.н., проф. Бісікало О. В.

«___»_____ 2022 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на магістерську кваліфікаційну роботу

«Модифікований алгоритм штучної бджолиної колонії

для розв'язання задач оптимізації»

08-31.МКР.010.02.000 ТЗ

Керівник роботи:

к.т.н., доц. каф. АІІТ

Іванов Ю. Ю.

«__» _____ 20__ р.

Виконавець:

ст. гр. 1ІСТ-21м

Ярська В. І.

«__» _____ 20__ р.

Вінниця ВНТУ – 2022

1. Назва та галузь застосування

Робота спрямована на дослідження недетерміністичних поліноміальних алгоритмів розв'язання задач оптимізації. У даній роботі розглядається модифікація алгоритму штучної бджолиної колонії, яку можна використовувати для розв'язання різноманітних прикладних задач оптимізації в умовах невизначеності.

2. Підстави для розробки

Розробку системи здійснювати на підставі наказу по університету №203 від 14.09.2022 та завдання до магістерської кваліфікаційної роботи, складеного та затвердженого кафедрою «Автоматизації та інтелектуальних інформаційних технологій».

3. Мета та призначення розробки

Метою роботи є підвищення ефективності розв'язання задачі неперервної оптимізації функцій з використанням алгоритму штучної бджолиної колонії за рахунок застосування нової моделі поведінки бджіл-спостерігачів.

4. Джерела розробки

1. Karaboga D., Basturk B. A Powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm. *Journal of Global Optimization*. 2007. Vol. 39. P. 459-471.

2. Gao W.F., Liu S.Y., Huang L.L. A Global Best Artificial Bee Colony Algorithm for Global Optimization. *Journal of Computational and Applied Mathematics*. 2012. Vol. 236. P. 2741-2753.

3. Устенко С.В., Бібко О.О. Використання методів біоніки в інтелектуальних інформаційних системах. *Інформаційні системи та мережі*. 2015. С. 501-508.

4. Liu J.-L., Li C.-C. An Improved Artificial Bee Colony Algorithm Applied to Engineering Optimization Problems. *Journal of information science and engineering*. 2016. Vol. 32. P. 863-886.

5. Показники призначення

Розроблено програмне забезпечення, яке дозволяє провести експериментальні дослідження з використанням алгоритму штучної бджолиної колонії у ході розв'язання задач оптимізації функцій.

Основні технічні вимоги та мінімальні системні вимоги до програми: операційна система *Windows*; кількість ядер комп'ютера – не менше 4 (*Core* i5 або *Ryzen* 5); оперативна пам'ять – не менше 8 ГБ; відеопам'ять – 512 Mb; жорсткий диск – 500 ГБ і більше.

Вхідні дані: розмірність простору пошуку D; параметри алгоритму (популяція бджіл N; кількість ітерацій, на яких забувається рішення K; кількість ітерацій алгоритму *limit*); кількість запусків для збору статистики m; набір тестових функцій. Результати роботи програми: значення екстремуму x^* функції f.

6. Економічні показники

До економічних показників входять:

- витрати на розробку – не більше 350 тис. грн.;
- приведена вартість прибутку за 3 роки – не менше 500 тис. грн.;

- мінімальна дохідність – 50 %;

- термін окупності – не менше 3-ох років;

- інші економічні переваги у порівнянні з аналогами.

7. Стадії розробки

1. Розділ 1 «Аналіз предметної області роботи» має бути виконаний до 10.10.22.

2. Розділ 2 «Розробка математичної моделі модифікованого алгоритму штучної бджолиної колонії» має бути виконаний до 25.10.22.

3. Розділ 3 «Розробка програмного забезпечення та експериментальні дослідження» має бути виконаний до 10.11.22.

4. Економічний розділ має бути виконаний до 20.11.22.

8. Порядок контролю та приймання

1. Рубіжний контроль провести до 07.12.22.

2. Попередній захист магістерської кваліфікаційної роботи провести до 07.12.22.

3. Захист магістерської кваліфікаційної роботи провести до 23.12.22.

Додаток Б

(обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

МОДИФІКОВАНИЙ АЛГОРИТМ ШТУЧНОЇ БДЖОЛИНОЇ КОЛОНІЇ ДЛЯ РОЗВ'ЯЗАННЯ ЗАДАЧ ОПТИМІЗАЦІЇ

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

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

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

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

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

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

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

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

Опонент _____ _____

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

Студентка гр. ІІСТ-21м _____ Ярська В. І.

(підпис) (ініціали та прізвище)

Рисунок Б.1 – Приклад випадкового блукання

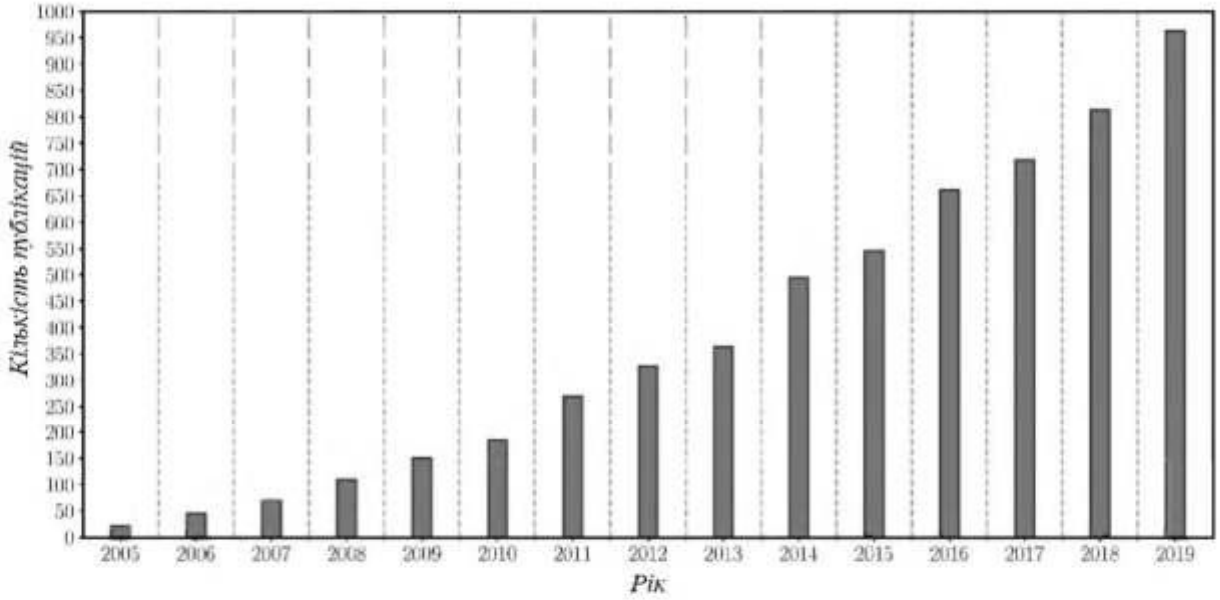

Рисунок Б.2 – Залежність кількості публікацій

на тему ройового інтелекту від року

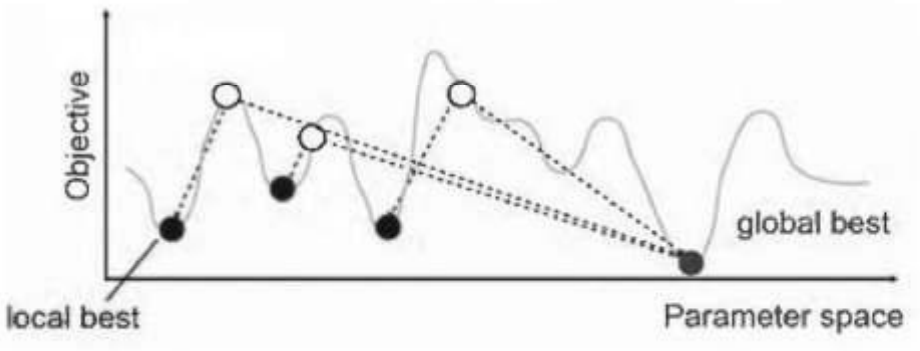

Рисунок Б.3 – Приклад роботи ройової оптимізації

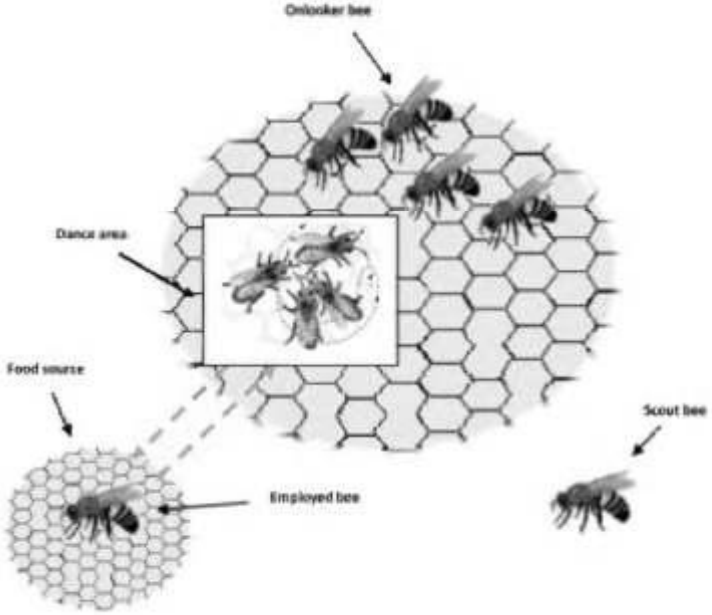

Рисунок Б.4 – Приклад поведінки бджіл

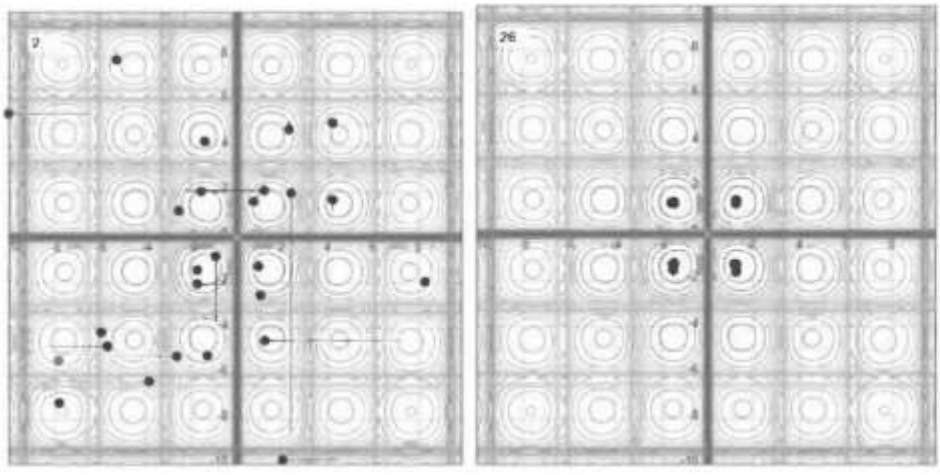

Рисунок Б.5 – Приклад роботи алгоритму штучної бджолиної колонії

Рисунок Б.6 – Схема програми

Додаток В

(обов'язковий)

Лістинг програми

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc;
close all;
CostFunction = rs_ObjectiveFunction(solutions);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
N = 5;
Size = [1 N];
Min = -10;
Max = 10;
Iter = 200;
Pop = 100;
C = zeros(Pop,1);
BestCost = zeros(Iter,1);
nOnlooker = Pop;
L = round(0.6*N*Pop);
a = 1;
empty_bee.Position = [];
empty_bee.Cost = [];
pop = repmat(empty_bee,Pop,1);
BestSol.Cost = inf;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i = 1:Pop
   pop(i).Position = unifrnd(Min,Max,Size);
   pop(i).Cost = CostFunction(pop(i).Position);
   if pop(i).Cost <= BestSol.Cost
      BestSol = pop(i);
   end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for it = 1:Iter
   for i = 1:Pop
      K = [1:i-1 i+1:Pop];
      k = K(randi([1 numel(K)]));
      phi = a*unifrnd(-1,+1,Size);
      newbee.Position = pop(i).Position+phi.*(pop(i).Position-pop(k).Position);
      newbee.Cost = CostFunction(newbee.Position);
      if newbee.Cost <= pop(i).Cost
         pop(i) = newbee;
```

```
      else
         C(i) = C(i)+1;
      end
   end

   F = zeros(Pop,1);
   MeanCost = mean([pop.Cost]);
   for i = 1:Pop
      F(i) = exp(-pop(i).Cost/MeanCost); % Convert Cost to Fitness
   end
   P = F/sum(F);

   for m=1:nOnlooker
      i = RouletteWheelSelection(P);
      K = [1:i-1 i+1:Pop];
      k = K(randi([1 numel(K)]));
      phi = a*unifrnd(-1,+1,Size);
      newbee.Position = pop(i).Position+phi.*(pop(i).Position-pop(k).Position);
      newbee.Cost = CostFunction(newbee.Position);
      if newbee.Cost <= pop(i).Cost
         pop(i) = newbee;
      else
         C(i) = C(i)+1;
      end
   end


   for i = 1:Pop
      if C(i ) >= L
         pop(i).Position = unifrnd(Min,Max,Size);
         pop(i).Cost = CostFunction(pop(i).Position);
         C(i) = 0;
      end
   end

   for i = 1:Pop
      if pop(i).Cost <= BestSol.Cost
         BestSol = pop(i);
      end
   end

   BestCost(it)=BestSol.Cost;
   disp(['Iteration ' num2str(it) ': Best Cost = ' num2str(BestCost(it))]);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure;
semilogy(BestCost,'LineWidth',2);
xlabel('Iteration');
ylabel('Best Cost');
```

```
grid on;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function i = RouletteWheelSelection(P)
    r = rand;
    C = cumsum(P);
    i = find(r<=C,1,'first');
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function z = fobj(u)
z = f(i) + (1-x(j))^2 + 100*(x(j+1) - x(j)^2)^2;
Z = 0;
P = 10^15;
g(1) = (u(1)^2 + u(2)^2)^3 - 4*2*u(1)^2*u(2)^2;
geq = [];

for k=1:length(g),
    if g(k) <= 0
       H = 0;
    else
       H = 1;
    end
    Z = Z + P*g(k)^2*H;
end
for k = 1:length(geq)
   if geq == 0,
      H = 0;
    else
      H = 1;
    end
    Z = Z + lam*geq(k)^2*H;
end

z = z + Z;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Додаток Г

(обов'язковий)

Довідка про можливість впровадження розробки

Додаток Д

(обов'язковий)

Протокол перевірки МКР

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Модифікований алгоритм штучної бджолиної колонії для розв'язання задач оптимізації.

Тип роботи: магістерська кваліфікаційна робота

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

Показники звіту подібності Plagiat.pl (StrikePlagiarism)

Оригінальність 97,1% Схожість 2,9 %

Аналіз звіту подібності (відмітити потрібне):

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

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

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

Автор роботи _____ Ярська В. І.
(підпис) (прізвище, ініціали)

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