

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ
ІНСТИТУТ

Кафедра математичного моделювання та аналізу даних

«На правах рукопису»
УДК 004.94; 517.9:519.6

До захисту допущено:
В.о. зав. кафедри
_____ Ганна ЯЙЛИМОВА
«__» _____ 2024 р.

Магістерська Дисертація
на здобуття ступеня магістра

за освітньо-науковою програмою «Математичні методи
моделювання, розпізнавання образів та комп'ютерного зору»
зі спеціальності 113 «Прикладна математика»
на тему: «Методи й алгоритми ройового інтелекту
знаходження розв'язків прикладних та інженерних завдань»

Виконав:
студент 4 курсу, групи ФІ-21мн
Татенко Вадим Сергійович _____

Науковий керівник:
к.т.н., ст. досл., доцент
Хайдуров Владислав Володимирович _____

Рецензент:
доцент, к. фіз.-мат. н., доцент
Цюпій Тамара Іванівна _____

Засвідчую, що у дипломній роботі
немає запозичень з праць інших
авторів без відповідних посилань.

Студент _____

Київ — 2024

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ
ІНСТИТУТ**

Кафедра математичного моделювання та аналізу дани

Рівень вищої освіти — другий (магістерський)
Спеціальність (освітня програма) — 113 Прикладна математика,
ОПП «Математичні методи моделювання, розпізнавання образів та
комп'ютерного зору»

ЗАТВЕРДЖУЮ

В.о. зав. кафедри

_____ Ганна ЯЙЛИМОВА

«__» _____ 2024 р.

**ЗАВДАННЯ
на магістерську дисертацію студенту
Татенко Вадиму Сергійовичу**

Студент: Татенко Вадим Сергійович

1. Тема дисертації: *«Методи й алгоритми ройового інтелекту
знаходження розв'язків прикладних та інженерних завдань»*,
керівник дисертації Хайдуров Владислав Володимирович к.т.н., ст.
досл., доцент,

затверджені наказом по університету від «10» квітня 2024 р. №1632__с

2. Термін подання студентом дисертації: «30» травня 2024 р.

3. Об'єкт дослідження: Математичні моделі, які описують поведінку
складних об'єктів та їх систем.

4. Предмет дослідження: Методи, алгоритми і математичні моделі
пошуку оптимальних параметрів процесів у складних об'єктах та їх
системах.

5. Перелік завдань, які потрібно розробити: аналіз останніх
наукових робіт з інженерними завданнями в оптимізаційній постановці,
аналіз методів й алгоритмів ройового інтелекту, їх застосування в
інженерних задачах.

6. Перелік ілюстративного матеріалу: презентація доповіді.

7. Орієнтовний перелік публікацій:

– Khaidurov V., Tatenko V., Lytovchenko M., Tsiupii T., Zhovnovach T. (2024) Acceleration of computations in modelling of processes in complex objects and systems. System Research in Energy, No. 1 (76). С. 58–70. Інститут загальної енергетики НАН України. <https://doi.org/10.15407/srenergy2024.02.058> (категорія Б).

– Татенко В.С., Хайдуров В. В. «Методи й алгоритми ройового інтелекту для знаходження розв’язків оптимізаційних задач моделювання поведінки складних об’єктів і систем» XXII Всеукраїнській науково-практичній конференції студентів, аспірантів та молодих вчених, секція «Математичне моделювання та аналіз даних».

8. Дата видачі завдання: 10.10.2022

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання	Примітка
1	Узгодження теми роботи із науковим керівником	01-15 вересня 2023 р.	Виконано
2	Огляд опублікованих джерел за тематикою дослідження	Вересень-жовтень 2023 р.	Виконано
3	Огляд джерел стосовно існуючих модифікацій алгоритмів й методів ройового інтелекту	Жовтень 2023 р.	Виконано
4	Розробка модифікацій методу оптимізації DE	Листопад 2023 р.	Виконано
5	Написання програмної реалізації для тестування розроблених модифікацій	Березень 2023 р.	Виконано
5	Оформлення магістерської дисертації	Квітень 2024 р.	Виконано

Студент _____ Вадим ТАТЕНКО

Керівник _____ Владислав ХАЙДУРОВ

РЕФЕРАТ

Кваліфікаційна робота містить: 66 стор., 15 рисунки, 12 таблиць, 11 джерел.

Оптимізаційні методи й алгоритми ройового інтелекту є ефективним інструментом вирішення прикладних оптимізаційних задач в різних сферах. Це може бути як оптимізація маршрутизації телекомунікаційних мереж, так пошук оптимальних параметрів для покращення характеристик фізичних конструкцій. Прикладні задачі можуть бути кардинально різної складності по кількості параметрів моделі і не тільки. В свою чергу існує не малий набір алгоритмів для вирішення таких задач. Але не є гарантованим те, що алгоритм впорається з поставленою задачею, чи впорається недостатньо точно, чи застрягне в локальному екстремумі, чи буде виконуватись довгий час. Тому розробка модифікацій до існуючих алгоритмів є актуальною задачею для вирішення.

В даній роботі буде представлено та протестовано нові модифікації для оптимізаційного методу диференційної еволюції, які протестовані на прикладних моделях, та сформовані порівняльні таблиці модифікованого методу DE з базовим методом DE.

АЛГОРИТМИ РОЙОВОГО ІНТЕЛЕКТУ, МЕТОДИ Й МОДЕЛІ ОПТИМІЗАЦІЇ, МАТЕМАТИЧНІ МОДЕЛІ, ДИФЕРЕНЦІЙНА ЕВОЛЮЦІЯ, АЛГОРИТМ РОЮ ЧАСТОК, БДЖОЛИНИЙ АЛГОРИТМ, ЗБІЖНІСТЬ, ПОПУЛЯЦІЯ

ABSTRACT

Qualification work contains: 66 pages, 15 figures, 12 tables, 11 sources.

Optimization methods and swarm interest algorithms are effective tool for solving applied optimization problems in various fields. This can be either the optimization of telecommunication network routing or the search for optimal parameters to improve the characteristics of physical structures. Applied problems can be of radically different complexity in terms of the number of model parameters and more. In turn, there is a large set of algorithms for solving such problems. However, it is not guaranteed that the algorithm will cope with the task, or will not cope accurately enough, or will get stuck in a local extremum, or will take a long time to execute. Therefore, the development of modifications to existing algorithms is an urgent problem to solve.

In this paper, we will present and test new modifications to the optimization method of differential evolution, which are tested on applied models, and form comparative tables of the modified DE method with the basic DE method.

SWARM INTELLIGENCE ALGORITHMS, OPTIMIZATION METHODS AND MODELS, MATHEMATICAL MODELS, DIFFERENTIAL EVOLUTION, PARTICLE SWARM ALGORITHM, BEE ALGORITHM, CONVERGENCE, POPULATION

ЗМІСТ

Перелік умовних позначень, скорочень і термінів	8
Вступ.....	9
1 Алгоритми й методи ройового інтелекту	11
1.1 Загальний опис ройового інтелекту	11
1.2 Конкретні приклади застосування ройових методів й алгоритмів оптимізації	13
1.2.1 Задача оптимізації траєкторії ракети-носія на етапі підйому	14
1.2.2 Задача блокового узгодження для оцінки руху при кодуванні відео.....	16
1.2.3 Задачі гнучкого планування роботи цеху	18
1.3 Базові методи й алгоритми РІ.....	19
1.3.1 Алгоритм рою часток	19
1.3.2 Опис PSO алгоритму.....	20
1.3.3 Різниця між Генетичним алгоритмом(GA) та PSO	22
1.3.4 Переваги і недоліки PSO	23
1.3.5 Бджолиний алгоритм	24
1.3.6 Опис ВА	26
1.3.7 Переваги та недоліки ВА	27
1.3.8 Метод диференційної еволюції	28
1.4 Переваги та недоліки DE	30
Висновки до розділу 1	31
2 Модифікації методів й алгоритмів РІ.....	33
2.1 Огляд існуючих модифікацій	33
2.1.1 Реконструйована диференціальна еволюція	33
2.1.2 Приклад модифікації PSO алгоритму.....	35
2.1.3 Приклад модифікації ВО	37
2.2 Розроблені модифікації.....	38
Висновки до розділу 2.....	39

3	Тестування запропонованих модифікацій	7 40
3.1	Опис експериментів	40
3.2	Тестування на базових функціях	41
3.3	Опис моделей для тестування	48
3.3.1	Модель конструкції редуктора	48
3.3.2	Задача мінімізації ваги пружини	51
3.4	Результати тестування на прикладних моделях	52
	Висновки до розділу 3	60
	Висновки	61
	Перелік посилань	63
	Додаток А Тексти програм	65
A.1	Базовий алгоритм DE	65
A.2	Модифікований алгоритм DE	67
A.3	Функція Растрігіна	70
A.4	Функція Розенброка	70
A.5	Модель редуктора	71
A.6	Модель пружини	72

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

<i>PSO</i>	— Алгоритм рою часток
<i>BA</i>	— Бджолиний алгоритм
<i>GA</i>	— Генетичний алгоритм
<i>PI</i>	— Ройовий інтелект
<i>AMPI</i>	— Алгоритми й методи ройового інтелекту

ВСТУП

В світі існує велика кількість прикладних задач, які мають за мету оптимізація процесу, конструкції, маршруту, тощо. Таким чином постійно є попит на пошук лібшого методу або алгоритму для отримання кращого розв'язку поставленої задачі. Адже це може доволі серйозно повпливати на результат виконання поставленого завдання. Що зачепить якість розв'язку, затрати на реалізацію, тощо. Прикладами можуть виступати задачі з відновлення пошкоджених даних чи розподіл електроенергії в інтелектуальних мережах. До того ж, важливим критерієм виступає не тільки якість отриманих результатів а й швидкість їх отримання.

АМРІ уявляють собою популяцію сутностей в просторі пошуку рішень. Така концепція є уособленням поведінки зграй живих істот в природі. До них можна віднести вовків, птахів, мурах, бджіл, світлячків, тощо. Їх всіх поєднує те, що є популяція, яка прагне кращого. Є окрема сутність цієї популяції, яка має свою визначену роль для досягання кращого стану популяції. Є простір в якому цій популяції треба виживати/харчуватись.

Кожен з багатьох існуючих АМРІ може застосовуватись до різного спектру задач. Та в той же час може справлятися з ними з різною точністю/швидкістю/ресурсоемністю. Для покращення зазначених характеристик розробляються методи, які покращать результативність роботи алгоритмів й методів оптимізації.

Метою дослідження є модифікація існуючих методів й алгоритмів пошуку оптимальних розв'язків задач, що зводяться до знаходження оптимальних параметрів моделей, що описують поведінку складних об'єктів та їх систем. Для досягнення мети необхідно проаналізувати счасні публікації за обраною тематикою. Зробити аналіз обраних АМРІ. Розробити необхідне програмне забезпечення для пошуку глобального оптимуму. Здійснити модифікацію одного з проаналізованих методів.

Об'єктом дослідження є математичні моделі, які описують поведінку складних об'єктів та їх систем.

Предметом дослідження є методи, алгоритми і математичні моделі пошуку оптимальних параметрів процесів у складних об'єктах та їх системах.

Наукова новизна полягає в розробці, опису та тестуванню модифікацій для АМРІ.

Практичне значення одержаних результатів: винайдення нових модифікацій для існуючих АМРІ покращує роботу методів й алгоритмів для яких розробляється модифікація. Це призводить до поліпшення результатів виходу алгоритму, а значить і моделі до якої застосовувався алгоритм оптимізації.

Апробація результатів дисертації. Робота оприлюднена на XXII Всеукраїнській науково-практичній конференції студентів, аспірантів та молодих вчених в секції «Математичне моделювання та аналіз даних»

Публікації:

– Khaidurov V., Tatenko V., Lytovchenko M., Tsiupii T., Zhovnovach T. (2024) Acceleration of computations in modelling of processes in complex objects and systems. System Research in Energy, No. 1 (76). С. 58–70. <https://doi.org/10.15407/srenergy2024.02.058> (фахове видання, категорія Б).

– Татенко В.С., Хайдуров В. В. «методи й алгоритми ройового інтелекту для знаходження розв'язків оптимізаційних задач моделювання поведінки складних об'єктів і систем» XXII Всеукраїнській науково-практичній конференції студентів, аспірантів та молодих вчених в секції «Математичне моделювання та аналіз даних»

1 АЛГОРИТМИ Й МЕТОДИ РОЙОВОГО ІНТЕЛЕКТУ

В розділі буде йти мова про загальні характеристики АМРІ. Наведуться приклади конкретних завдань, які розв'язуються за допомогою АМРІ. Також будуть зазначені ті, які будуть розглядатись в подальших розділах.

1.1 Загальний опис ройового інтелекту

Ройовий інтелект - це галузь штучного інтелекту, натхненна колективною поведінкою децентралізованих систем, що самоорганізуються, які часто зустрічаються в природі. Термін «ройовий інтелект» вперше був запропонований Джерардо Бені та Цзин Ваном у 1989 році в контексті клітинних робототехнічних систем. АМРІ в цій галузі призначені для імітації поведінки соціальних організмів, таких як зграї птахів, косяки риб і колонії мурах, для вирішення складних проблем у розподілений і гнучкий спосіб.

Фундаментальна концепція РІ базується на ідеї, що прості агенти, дотримуючись базових правил і локально взаємодіючи один з одним і навколишнім середовищем, можуть створювати складну і розумну глобальну поведінку без централізованого контролю. Ці агенти, яких часто називають частинками, сутностями або індивідами, діють на основі таких принципів, як

- Локальне сприйняття. Агенти сприймають і реагують лише на інформацію, доступну в їхньому безпосередньому оточенні. Ця локальна взаємодія призводить до появи складних моделей і поведінки на глобальному рівні.

- Позитивний і негативний зворотний зв'язок. Позитивний зворотний зв'язок допомагає посилити певні дії або рішення, сприяючи

поведінці, яка призводить до позитивних результатів. І навпаки, негативний зворотний зв'язок перешкоджає менш ефективним діям, допомагаючи відвернути систему від неоптимальної поведінки.

– Випадковість. Включення випадковості в поведінку агентів допомагає уникнути локальних мінімумів і забезпечує більш надійний пошук глобальних рішень. Цей стохастичний елемент дозволяє системі досліджувати ширший спектр можливостей і знаходити більш оптимальні рішення.

АМРІ знайшли застосування в різних галузях завдяки своїй надійності, гнучкості та адаптивності. Наведемо приклад деяких з найбільш відомих застосувань:

– Задачі оптимізації. Такі алгоритми, як PSO та оптимізація мурашиних колоній (АСО), широко використовуються для вирішення складних оптимізаційних задач в інженерії, економіці та логістиці.

– Робототехніка. Принципи РІ застосовуються для координації груп роботів для таких завдань, як розвідка, спостереження та пошуково-рятувальні місії. Децентралізоване управління дозволяє створювати масштабовані та відмовостійкі роботизовані системи.

– Інтелектуальний аналіз даних і машинне навчання. АМРІ використовуються для кластеризації, класифікації та виділення ознак у великих масивах даних. Їх здатність до пошуку та адаптації робить їх придатними для динамічних і багатовимірних середовищ даних.

– Телекомунікації. У мережевій маршрутизації та розподілі ресурсів методи РІ допомагають розробляти ефективні та адаптивні протоколи, які можуть обробляти динамічні умови мережі та оптимізувати шляхи зв'язку.

Системи РІ мають кілька ключових характеристик, які роблять їх придатними для вирішення складних і динамічних завдань:

– Масштабованість. АМРІ можуть ефективно масштабуватися залежно від кількості агентів, що робить їх придатними як для малих, так і для великих систем.

– Надійність. Децентралізована природа систем РІ означає, що вони

є стійкими до збоїв в роботі окремих агентів. Система може продовжувати функціонувати, навіть якщо деякі агенти вийшли з ладу.

– Гнучкість. Системи РІ можуть адаптуватися до мінливого середовища і проблемних ландшафтів, що робить їх дуже гнучкими і універсальними.

Незважаючи на численні переваги, існує кілька викликів, пов'язаних з системами РІ:

– Розробка ефективних правил. Створення простих, але ефективних правил для агентів, які призводять до бажаної глобальної поведінки, може бути складним завданням.

– Налаштування параметрів. АМРІ часто вимагають ретельного налаштування параметрів для досягнення оптимальної продуктивності, що може зайняти багато часу і залежати від конкретної проблеми.

– Проблеми масштабованості. Хоча системи РІ є масштабованими, обчислювальна складність може зростати зі збільшенням кількості агентів і взаємодій, що вимагає ефективних стратегій реалізації.

Майбутні дослідження в галузі РІ, ймовірно, будуть зосереджені на розробці більш досконалих моделей, здатних вирішувати все більш складні і динамічні проблеми. Інтеграція РІ з іншими методами РІ, такими як глибоке навчання та еволюційні алгоритми, може призвести до створення більш потужних гібридних систем. Крім того, вивчення застосування РІ в нових галузях, таких як автономні системи, розумні міста та біоінформатика, має великі перспективи.

1.2 Конкретні приклади застосування ройових методів й алгоритмів оптимізації

Можна доволі довго наводити приклади того, де можна застосовувати АМРІ. Але пропонується зупинитись на деяких з них для більш конкретизованого розуміння застосованості РІ.

1.2.1 Задача оптимізації траєкторії ракети-носія на етапі підйому

Першою розглянемо модель яка має назву «Задача оптимізації траєкторії ракети-носія на етапі підйому»[1].

Основною метою цього дослідження є оптимізація траєкторії фази підйому одноступеневої гіперзвукової ракети-носія на рідинному паливі з використанням методу оптимізації PSO. Завдання полягає в досягненні бажаних кінцевих умов з кутом атаки як керуючою змінною. У дослідженні вивчається вплив двох типів інерційних вантажів в алгоритмі PSO: лінійно зменшуваного інерційного вантажу та інерційного вантажу з імітацією відпалу.

Математичне моделювання ракети-носія та рухової установки базується на наступних рівняннях руху:

$$\dot{r} = V \sin \gamma \quad (1.1)$$

$$\dot{V} = \frac{1}{mV}(T \cos \alpha - D - mg \sin \gamma) \quad (1.2)$$

$$\dot{\gamma} = \frac{1}{mV}(T \cos \alpha + L) + \left(\frac{V}{r} - \frac{g}{V}\right) \cos \gamma \quad (1.3)$$

$$\dot{x} = V \cos \gamma \quad (1.4)$$

where:

- r миттєва радіальна відстань від центру Землі,
- V швидкість,
- γ кут траєкторії польоту транспортного засобу,
- T сила тяги,
- m миттєва маса,

- α кут атаки,
- L підйомна сила,
- D сила опору,
- g гравітаційна сила з боку Землі.

Підйомна сила та сила підйому відповідно мають наступне приєднання:

$$D = \frac{1}{2} \rho S V^2 c_D \quad (1.5)$$

$$L = \frac{1}{2} \rho S V^2 c_L \quad (1.6)$$

де

- ρ аеродинамічна густина,
- S аеродинамічна референтна область.

Гравітаційне тяжіння моделюється як функція висоти:

$$g = g_0 \left(\frac{r_e}{r} \right)^2 \quad (1.7)$$

В роботі [1] цільова функція спрямована на мінімізацію похибки кінцевого стану визначається як:

$$J = \sum_{i=1}^n s_i \|x_f - x_{tf}\|^2 \quad (1.8)$$

де

– x_f та x_{tf} значення змінних стану на кінцевому етапі та в кінцевий момент часу відповідно,

- s_i ваговий коефіцієнт для кожної змінної стану.

Змінні стану включають радіальну відстань від центру Землі (r), швидкість (V), та кут траєкторії польоту (γ). На оптимізацію накладено наступні обмеження:

$$5800 \leq V \leq 7700; -6 \leq \gamma \leq 6; \dot{m} = -mc \quad (1.9)$$

Початкові та кінцеві умови для змінних стану мають вигляд:

– Початкові: $r_0 = 6805854.5963 \text{ m}$, $V_0 = 5890.1127 \text{ m/s}$, $\gamma_0 = 5.2229^\circ$,
 $m_0 = 5200 \text{ kg}$

– Кінцеві: $r_f = 6862415.73 \text{ m}$, $V_f = 5890.1127 \text{ m/s}$, $\gamma_f = 0.002^\circ$

Результати наведеної роботи [1] демонструють, що PSO з адаптивними інерційними схемами ваги є ефективним для досягнення оптимальних траєкторій фази підйому для ракет-носіїв, забезпечуючи високу точність в кінцевих умовах без потреби в похідних розрахунках.

1.2.2 Задача блокового узгодження для оцінки руху при кодуванні відео

Одним з яскравих прикладів застосування штучного бджолиного алгоритму (ABC) є алгоритм блокового узгодження для оцінки руху при кодуванні відео на основі ВА оптимізації [2]. Цей алгоритм має на меті зменшити обчислювальну складність, пов'язану з оцінкою руху, що має вирішальне значення для ефективного стиснення відео.

Основною метою запропонованого алгоритму зіставлення блоків на основі ABC є мінімізація суми абсолютних різниць (SAD) між блоками пікселів у послідовних кадрах, що в традиційних методах вимагає значних обчислювальних затрат.

SAD визначається як:

$$\text{SAD}(u, v) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |I_t(x + i, y + j) - I_{t-1}(x + u + i, y + v + j)| \quad (1.10)$$

де (x, y) - верхній лівий кут поточного блоку у поточному кадрі I_t ,
 (u, v) - вектор руху, а N - розмір блоку.

В розглядаємій прикладній задачі, Фітнес функція, яка використовується для оцінки якості кожного рішення-кандидата, має

ВИГЛЯД:

$$\text{Fitness}(J) = \begin{cases} \frac{1}{1+J} & \text{if } J \geq 0 \\ 1 + |J| & \text{if } J < 0 \end{cases} \quad (1.11)$$

де J - значення цільової функції (у цьому контексті SAD).

Параметри, що оптимізуються алгоритмом ABC, включають компоненти вектора руху (u, v) , які мінімізують SAD. Дослідження продемонстрували, що алгоритм на основі ABC дозволив значно зменшити кількість необхідних оцінок функцій, зберігаючи при цьому високий рівень точності оцінки руху.

Результати моделювання показали, що запропонований алгоритм ABC-BM (Artificial Bee Colony-Block Matching) досягнув балансу між обчислювальними витратами та точністю оцінки, що перевершує інші швидкі алгоритми блокового зіставлення. Зокрема, алгоритм ABC-BM зменшив кількість обчислень SAD завдяки використанню стратегії обчислення пристосованості на основі інтерполяції найближчого сусіда (NNI), яка наближає значення пристосованості для деяких рішень-кандидатів замість того, щоб оцінювати їх усі.

Експерименти, проведені на стандартних тестових послідовностях, показали, що алгоритм ABC-BM забезпечує кращі показники відношення пікового сигналу до шуму (PSNR) та ефективності пошуку порівняно з традиційними методами. Коефіцієнт деградації PSNR був значно нижчим, а економія обчислювальних ресурсів - суттєвою, що робить алгоритм ABC-BM життєздатним рішенням для обробки відео в реальному часі.

Підводячи підсумок даного прикладу, можна зробити висновок, що застосування алгоритму ABC у блочному зіставленні для оцінки руху демонструє потенціал ройового інтелекту у вирішенні складних оптимізаційних задач в інженерії.

1.2.3 Задачі гнучкого планування роботи цеху

Метою цього дослідницького проекту [3] є вивчення та застосування методу DE для розв'язання задачі гнучкого планування роботи цеху (FJSP). Основною метою є мінімізація тривалості виконання, тобто загального часу, необхідного для завершення набору робіт. Вдосконалюючи метод DE, дослідження має на меті забезпечити більш ефективні та дієві рішення порівняно з традиційними методами та іншими метаевристичними алгоритмами.

Параметри фітнес-функції, оптимізовані в цьому дослідженні, були спрямовані на мінімізацію тривалості виконання ($C_{\max}$), який оцінюється за формулою:

$$\text{Minimize } Z = f(C1, C2, \dots, Cp), \quad (1.12)$$

де $C1, C2, \dots, Cp$ - можливі розв'язки задачі планування, а Z - цільова функція, що представляє найнижче значення тривалості виконання.

Відносна похибка обчислюється за допомогою:

$$\text{MRE} = \frac{C_{\min} - \text{BKS}}{\text{BKS}} \times 100, \quad (1.13)$$

де $C_{\min}$ - оптимальний розв'язок, знайдений методом, а BKS - найкращий розв'язок.

Результати дослідження показали, що для невеликих задач Кацема комбінація «DE/rand/1» та експоненціального кросовера в позиції 2 забезпечує мінімальну відносну похибку (MRE) 3.25. Для задач Брандімарте середнього розміру «DE/best/2» з експоненціальним кросовером у позиції 2 забезпечив MRE 7,11. Для великих задач Дозере-Переса і Паулі той самий метод досягнув MRE 4.20. Ці результати свідчать про те, що покращений метод DE є ефективним і забезпечує нижчу MRE порівняно з іншими метаевристиками, такими як PSO та іншими варіаціями DE.

1.3 Базові методи й алгоритми PI

В даній роботі будуть розглядатись PSO, BA та DE. Зараз необхідно зрозуміти принцип кожного з перелічених алгоритмів для розуміння контексту в якому далі буде розроблятися модифікація.

1.3.1 Алгоритм рою часток

PSO – це обчислювальний метод, натхненний колективною поведінкою природних організмів, таких як птахи чи риби, які рухаються спільно для досягнення загальної мети. У PSO група частинок, що представляють можливі рішення певної задачі, рухається простором вирішення проблеми, щоб знайти найкраще можливе рішення. Кожна частинка коригує своє положення на основі свого найкращого рішення (особистого рекорду) та найкращого рішення, знайденого всією групою (глобального рекорду). Такий спільний рух дозволяє частинкам поступово зближуватись до оптимальних рішень. PSO широко застосовується для оптимізації в різних сферах, використовуючи силу колективного інтелекту для дослідження складних просторів рішень і ефективного пошуку глобальних оптимальних результатів.

Алгоритм PSO був запропонований Еберхартом (Eberhart) та Кеннеді (Kennedy) [4]. Подібно до еволюційних метаевристик, PSO використовує популяцію особин – можливих рішень проблеми, та метод стохастичної оптимізації, що базується на соціальній поведінці у рої.

PSO стосується пошуку оптимального рішення у просторі задач великої розмірності. Завдання полягає у знаходженні максимуму або мінімуму функції для досягнення оптимального розв'язку. Функція може мати кілька локальних максимумів і мінімумів, але лише один глобальний максимум і мінімум. Якщо цільова функція дуже складна, пошук глобального максимуму стає надзвичайно складним. PSO намагається

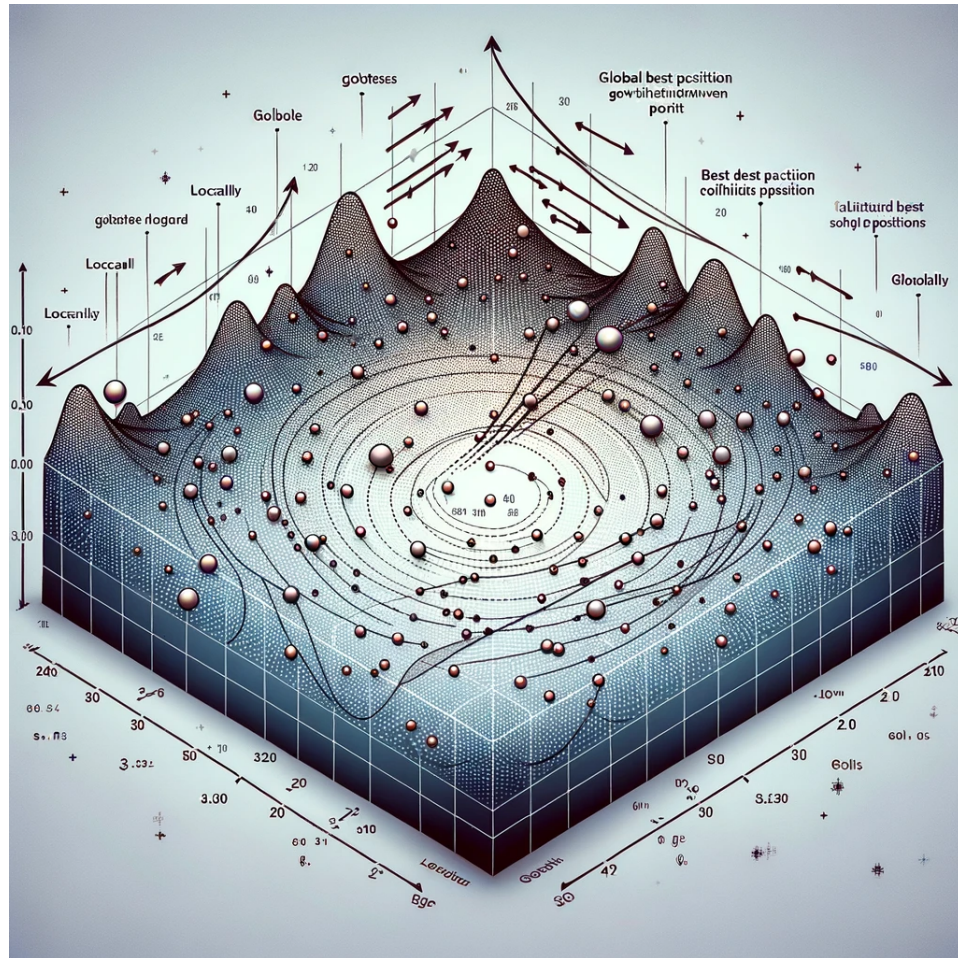


Рисунок 1.1 – Ілюстрація алгоритму PSO

ефективно виявити глобальний максимум або мінімум. Хоча PSO не завжди знаходить точний глобальний максимум або мінімум, він здатен знайти рішення, дуже близьке до оптимального. З практичної точки зору, точне значення глобального екстремуму не є критично важливим, оскільки наближене значення зазвичай задовольняє вимоги дослідження реальних процесів різної природи.

1.3.2 Опис PSO алгоритму

На початку роботи алгоритму відбувається ініціалізація популяції, після чого починається ітеративне переміщення частинок популяції по простору рішень в пошуках субоптимального рішення задачі мінімізації чи максимізації. На відміну від еволюційних алгоритмів, PSO не

використовує генетичні оператори під час роботи алгоритму. В просторі пошуку частинки переміщуються в напрямку який корегується власним досвідом кожної частинки та глобальними результатами популяції. Таким чином можлива ситуація коли частинка в одній частині популяції може змінити свій напрямок в сторону іншого через те, що там оновилося кращий результат популяції.

Кожну i -а частинку можна охарактеризувати за допомогою трьох характеристик: номер ітерації n , положення частинки у просторі рішень $x_i(n)$ та швидкості руху сутності $v_i(n)$.

Швидкість i -ї частинки обчислюється як:

$$v_i(n+1) = v_i(n) + \alpha_1 (x_i^{best}(n) - x_i(n)) r_1 + \alpha_2 (x^*(n) - x_i(n)) r_2, \quad (1.14)$$

позиція i -ї частинки обчислюється у вигляді

$$x_i(n+1) = x_i(n) + v_i(n+1), \quad (1.15)$$

де

$x_i(n) = (x_{i1}(n), x_{i2}(n), \dots, x_{iM}(n))$ – позиція i -ї частинки на ітерації n ;

$x_i^{best}(n) = (x_{i1}^{best}(n), x_{i2}^{best}(n), \dots, x_{iM}^{best}(n))$ – найкраща позиція i -ї частинки (персональна найкраща позиція);

$x^*(n) = (x_1^*(n), x_2^*(n), \dots, x_M^*(n))$ – найкраща позиція по всій популяції (глобальна краща позиція);

$v_i(n) = (v_{i1}(n), v_{i2}(n), \dots, v_{iM}(n))$ – вектор швидкості i -ї частинки на ітерації

α_1, α_2 – додатні коефіцієнти прискорення, що регулюють внесок когнітивної та соціальної компонент;

$r_1 = (r_{11}, r_{12}, \dots, r_{1M}), r_2 = (r_{21}, r_{22}, \dots, r_{2M})$ – вектори випадкових чисел, що вносять елемент випадковості у процес пошуку.

Розглянемо детальніше зміну швидкості, а саме з яких

компонент(доданків) вона складається. Перша компонента в рівнянні 1.14, $v_i(n)$, зберігає попередній напрямок швидкості i -тої частинки і може розглядатися як момент, що перешкоджає різкій зміні напрямку швидкості, виконуючи роль інерційної компоненти (inertia velocity). Другий доданок є когнітивною компонентою (cognitive velocity), яка визначає характеристики частинки щодо її минулих досягнень, зберігаючи найкращу позицію цієї частинки. Цей доданок намагається повернути частинку назад до її найкращої досягнутої позиції. Третій доданок визначає соціальну компоненту (social velocity), яка характеризує частинку відносно її сусідів. Соціальна компонента спрямовує кожну частинку в бік глобального оптимуму, досягнутого роєм або його найближчим оточенням.

Оптимізаційний процес може продовжуватись до досягання умови зупинки. Нею може виступати як необхідна точність пошуку, так і максимальна кількість ітерацій. Відповідно, у контексті розв'язання задачі, в якій відомий глобальний мінімум, критерієм зупинки виступає точність. Зазвичай на такого роду задачах проганяється алгоритм для визначення, чи написана реалізація коректно працює або для тестування нових модифікацій які імплементовуються в структуру алгоритма. І звичайно таких тестів треба провести деяку кількість на різних моделях/тестових функціях, для того щоб переконатись, що все працює коректно. В контексті задач, де не відомо глобальний оптимум, то тут критерієм зупинки виступає максимальна кількість ітерацій.

1.3.3 Різниця між Генетичним алгоритмом(GA) та PSO

Генетичні алгоритми GA і PSO обидва застосовуються для знаходження глобального екстремуму функцій, ґрунтуючись на значеннях відповідної функції пристосованості. Обидва алгоритми є ітераційними та мають стохастичну складову. Вони підходять для вирішення схожих оптимізаційних завдань. Основна різниця між PSO і

генетичними алгоритмами GA полягає в тому, що GA не перетинає простір пошуку. У GA варіанти рішень генеруються випадковим чином, і найкращі з них обираються для конкуренції з новим набором випадкових рішень. Крім того, алгоритми PSO потребують нормалізації вхідних векторів для швидшого досягнення «конвергенції» (як евристичні алгоритми, вони не обов'язково сходяться до оптимального рішення). GA можуть працювати як з неперервними, так і з дискретними функціями. У PSO немає створення або видалення особин; частинки просто рухаються по ландшафту, і їхня ефективність оцінюється з часом.

1.3.4 Переваги і недоліки PSO

Переваги алгоритму:

- PSO відносно простий у реалізації та розумінні, що робить його доступним для широкого кола користувачів без глибоких знань математики чи оптимізації;
- PSO здатний адаптуватися до змін у навколишньому середовищі або задачі оптимізації, вносячи зміни до параметрів або стратегії пошуку;
- PSO легко розпаралелюється для здійснення одночасної обробки даних;
- PSO не використовує жодної інформації про похідні цільової функції та відповідних обмежень на неї;
- у PSO-алгоритмі порівняно мало параметрів;
- дуже ефективний для пошуку глобального екстремуму.

Недоліки алгоритму:

- пошук локального екстремуму PSO є слабким;
- ефективність PSO може залежати від вибору параметрів, таких як швидкість навчання, кількість частинок тощо. Неправильно вибрані параметри можуть призвести до неефективного пошуку;
- PSO не гарантує знаходження оптимального рішення за кінцевий час. Результати можуть змінюватись при різних запусках через випадковий

характер руху частинок;

– PSO може споживати значні обчислювальні ресурси, особливо під час роботи з великими просторами пошуку чи дуже великим роєм частинок.

1.3.5 Бджолиний алгоритм

Бджолині алгоритми є відносно новим методом оптимізації, заснованим на моделюванні поведінки бджіл під час пошуку ділянок з багатим вмістом нектару. Основна ідея цього алгоритму полягає в тому, що кілька бджіл-розвідників вилітають з вулика у випадкових напрямках у пошуках найбільш продуктивних ділянок. Після повернення до вулика вони інформують інших бджіл про місця та кількість знайденого нектару. Наступним кроком на виявлені багаті нектаром ділянки вирушають інші бджоли – фуражири(робітники). Кількість фуражирів, що летять до конкретної ділянки, залежить від очікуваної кількості нектару на ній. Тим часом розвідники знову вилітають на пошуки нових ділянок, і процес повторюється. Для ефективного моделювання колективної поведінки бджіл необхідно враховувати щонайменше такі аспекти: джерела нектару, зайняті та незайняті фуражири.

Якість джерела їжі залежить від таких факторів: відстань до вулика, концентрація нектару та зручність збору. Для спрощення оцінки використовують один критерій - «рентабельність» джерела їжі. Зайняті фуражири експлуатують певне джерело їжі, маючи інформацію про нього. Вони діляться цією інформацією з іншими бджолами з певною ймовірністю.

Дослідження показують, що у бджолиних вуликах можна виділити компоненти, які присутні у всіх. Найважливу роль в обміні інформацією між бджолами грає «танець». Він уявляє собою інформування інших бджіл бджолою розвідником про інформацію стосовно знайденого джерела їжі. За видом цього танцю можна визначити якість знайденого джерела. Незайняті

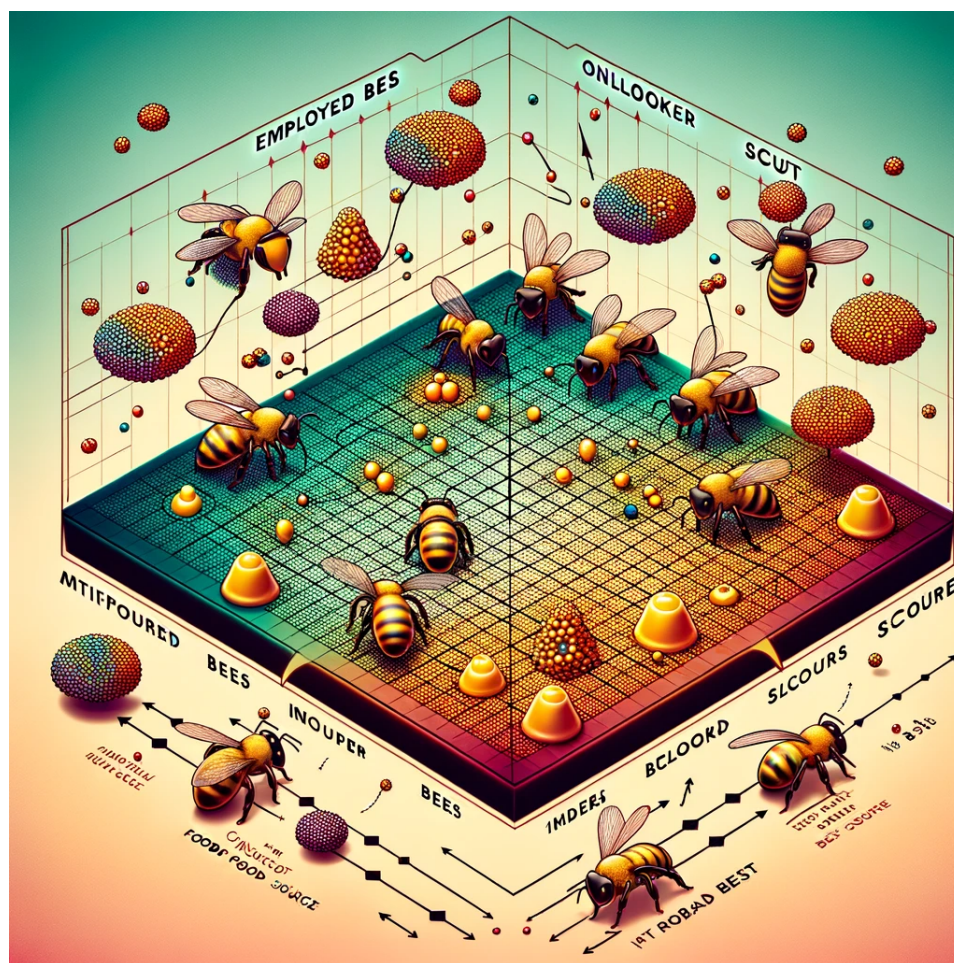


Рисунок 1.2 – Ілюстрація алгоритму ВО

бджоли спостерігають за «танцями», за їх виглядом і тривалістю з деякою ймовірністю пропорційної рентабельності джерела, вирішують летіти до цього джерела їжі.

Самоорганізація мультиагентних систем базується на ряді динамічних механізмів, які дають результати на глобальному рівні системи через взаємодію компонентів нижчого рівня. Ці механізми засновані на основних правилах взаємодії між компонентами системи. Слід зазначити, що правила часто використовують локальну інформацію без урахування глобального стану системи. В основі самоорганізації системи лежать чотири основні властивості: позитивний зворотний зв'язок, негативний зворотний зв'язок, випадкові флуктуації та множинні взаємодії.

Позитивний зворотний зв'язок ґрунтується на простих правилах

поведінки, що забезпечують створення зручних (необхідних) структур. Наприклад, при збільшенні кількості нектару з виявлених джерел їжі збільшується також кількість бджіл, що їх відвідали.

Негативний зворотний зв'язок врівноважує позитивний і стабілізує поведінку системи. Наприклад, процес пошуку джерел їжі може сповільнитися зі збільшенням фуражування.

Флуктуації як випадкові відхилення й перемикання між особинами рою життєво необхідні для придбання нових властивостей та пошуку нових рішень. Наприклад, бджоли-розвідники випадково досліджують нові області простору у пошуках нектару. Загалом, самоорганізація вимагає взаємодії між особами, наприклад інформація, отримана однією бджолою про джерело живлення, стає доступною іншим бджолам після виконання відповідного «танцю». При функціонуванні рою одночасно виконуються багато різних завдань, що вимагає «поділу праці» різних особин. Кооперація спеціалізованих груп особин під час вирішення певної проблеми, зазвичай, ефективніша послідовного розв'язання завдання неспеціалізованими особами.

Тому найважливішими є дві фундаментальні концепції — самоорганізація та «поділ праці», які забезпечують розумну поведінку розподілених мультиагентних систем та їх адаптацію до навколишнього середовища.

1.3.6 Опис ВА

ВО запропонований Фемом (Pham)[5] і заснований на поведінці медоносних бджіл. Основу алгоритму складають дві фази — фаза бджоли розвідника і фаза бджоли робітника. Метою алгоритму є визначення положення якісних ділянок з їжею у просторі пошуку. Після того, як розвідники знаходять ділянки, на них відправляються Бджоли робітники для «збору їжі» (більш детального вивчення ділянки). Рішення алгоритму є відповідно до кращої позиції бджоли, чи то розвідник, чи то

робітник.

На початку алгоритму йде випадкове ініціалізування популяції в рамках простору пошуку. Також необхідно визначити кількість елітних ділянок та звичайних ділянок. Їх різниця полягає в тому, скільки бджіл робітників відправиться на ту чи іншу ділянку для детального її вивчення.

Принцип зупинки роботи алгоритму є ідентичним до того, який був описаний в PSO. Тобто або досягнута максимальна кількість ітерацій, або досягнуто заданої точності, якщо відомий глобальний оптимум.

1.3.7 Переваги та недоліки ВА

Переваги алгоритму:

- ВА відносно простий у розумінні та реалізації. Це робить його доступним для широкого кола користувачів без глибоких знань у галузі оптимізації;
- ВА здатний досліджувати як широкі області пошуку, так і заглиблюватись у локальні області, що допомагає уникнути застрягання у локальних оптимумах;
- ВА добре справляється з адаптацією до умов, що змінюються, завдання оптимізації. Бджоли можуть змінювати свою стратегію залежно від довкілля;
- ВА зазвичай менш чутливий до вибору параметрів, ніж інші алгоритми оптимізації, що полегшує його використання.

Недоліки алгоритму:

- Результати ВА можуть значно змінюватись в залежності від випадкового початкового стану та інших факторів, що роблять його недетермінованим;
- у деяких випадках ВА може мати нижчу швидкість збіжності порівняно з іншими методами оптимізації, особливо у складних просторах пошуку;

- у деяких задачах ВА може виявитися менш ефективним через специфічні характеристики завдання або простору пошуку;
- при роботі з великими просторами пошуку або великими числами бджіл ВА може споживати значні обчислювальні ресурси.

1.3.8 Метод диференційної еволюції

DE був розроблений в 1997 році Рональдом Сторном і Кеннетом Прайсом. З тих пір цей метод став фундаментальним інструментом в області оптимізації.

Основна ідея методу DE [6] полягає у поєднанні мутації та кросинговеру для ефективного пошуку оптимальних рішень у багатовимірних просторах параметрів. Цей метод особливо ефективний для роботи з нелінійними, недиференційованими та мультимодальними цільовими функціями. На відміну від традиційних методів оптимізації, DE не потребує інформації про градієнт і може легко орієнтуватися в складних ландшафтах.

DE черпає натхнення з GA, використовуючи схожі принципи природного відбору. Однак він відрізняється тим, що не покладається на двійкове кодування рішень. Натомість, DE оперує безпосередньо з векторами реальних значень, що спрощує роботу з реальними оптимізаційними задачами. Процес починається з популяції рішень-кандидатів, які еволюціонують протягом ітерацій завдяки застосуванню операторів мутації, кросинговеру та відбору.

Мутація в DE передбачає створення нового рішення-кандидата шляхом додавання зваженої різниці між двома векторами популяції до третього вектора. Ця стратегія збільшує різноманітність і допомагає ефективніше досліджувати простір пошуку. Потім кросовер поєднує мутований вектор з цільовим вектором для отримання пробного вектора, що додає нові характеристики рішення і полегшує локальний пошук. Нарешті, процес відбору гарантує, що до наступного покоління перейдуть

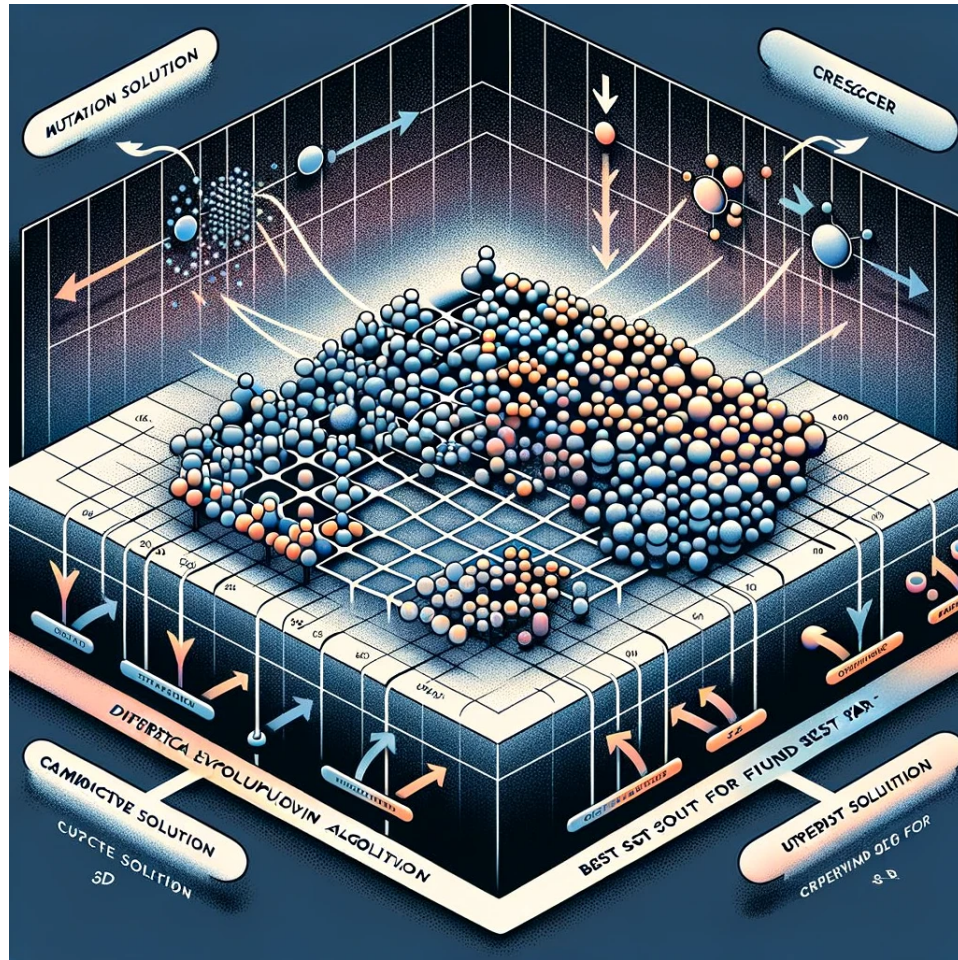


Рисунок 1.3 – Ілюстрація алгоритму DE

лише найкращі рішення.

Сила DE полягає в його простоті та надійності, що робить його придатним для широкого спектру застосувань, від інженерного проектування до налаштування параметрів машинного навчання. Його здатність знаходити глобальні оптимуми в складних ландшафтах з відносно невеликою кількістю керуючих параметрів закріпила його місце як універсального і потужного інструменту оптимізації.

Задача: $F(X) \rightarrow \min, X = X_1, X_2, \dots, X_M$.

Математичний опис основних етапів методу:

1) Для кожної хромосоми $\bar{x}_i$ в популяції,

$$\bar{x}_i = (\bar{x}_{i1}, \bar{x}_{i2}, \dots, \bar{x}_{iM}) \quad (1.16)$$

(N – розмірність популяції) обираються три інші випадкові члени цієї

популяції x_1, x_2, x_3 ,

$$\overline{x_i} \neq x_1, \overline{x_i} \neq x_2, \overline{x_i} \neq x_3, \quad (1.17)$$

$$x_1 \neq x_2 \neq x_3.$$

2) Генерується мутантний вектор:

$$v = x_1 + F(x_2 - x_3), F \in (0; 2). \quad (1.18)$$

3) Різниця векторів $x_2 - x_3$ масштабується за визначеним користувачем гіперпараметром F , $F \in (0; 2)$.

4) Формуємо пробний вектор на основі схрещування $\overline{x_i}$ з мутантним вектором v : для кожної координати хромосоми генерується випадкове число $r \in (0; 1)$ за нормальним розподілом. Якщо $r < P$, P – задана константа (ще один параметр методу, $P \in (0; 1)$), то відповідна координата хромосоми замінюється на

$$v_i = \overline{x_{ij}}, j \in \overline{1; M}, \quad (1.19)$$

M – розмірність простору пошуку (кількість незалежних змінних фітнес-функції).

5) Якщо $f(\overline{x_i}) > f(v)$, $\overline{x_i} = v$.

Усі вищенаведені кроки виконуються до критерію зупинки методу (класичні, як і в інших AMPI).

1.4 Переваги та недоліки DE

Переваги методу:

- DE відносно простий у реалізації і може бути легко адаптований до різних завдань оптимізації.
- DE має здатність знаходити глобальні оптимуми у багатовимірних просторах параметрів без вимог додаткових градієнтних даних.
- DE може успішно вирішувати завдання оптимізації з негладкими,

дискретними або галасливими функціями, що робить його корисним для різних прикладних задач.

– DE має невелику кількість гіперпараметрів, що спрощує процес підбору параметрів для конкретного завдання.

Недоліки методу:

– Ефективність методу може залежати від вибору цих параметрів, таких як коефіцієнт F , що масштабує, і ймовірність кросовера P .

– DE може мати різну ефективність в залежності від характеристик оптимізованої функції, і його продуктивність може сильно змінюватись на різних типах завдань.

– Іноді DE може застрягти в локальних оптимумах, особливо за наявності виражених локальних мінімумів або погано збалансованих параметрів методу.

– DE може вимагати значних обчислювальних ресурсів під час роботи з великими розмірами простору параметрів.

Висновки до розділу 1

За результатами цього огляду літератури можна зробити висновок, що AMPI широко застосовуються в багатьох кардинально різних галузях. Ці сфери охоплюють від інженерії та робототехніки до фінансів і охорони здоров'я, демонструючи широке застосування та універсальність методів PI. Важливо, що огляд підкреслив не лише ефективність цих алгоритмів, але й їхню адаптивність до різних проблемних областей, що робить їх безцінними інструментами у вирішенні сучасних обчислювальних проблем.

Одним з ключових висновків огляду є вражаючі показники продуктивності, пов'язані з методами PI. Ці алгоритми стабільно демонструють хороші показники ефективності, часто перевершуючи традиційні методи у вирішенні складних реальних проблем. Їх гнучкість дозволяє адаптувати їх до конкретних застосувань, забезпечуючи

оптимальну продуктивність у різних сценаріях.

Окрім загального огляду, в розділі також були розглянуті конкретні представники РІ, модифікації до яких будуть більш детально розглянуті в наступних розділах. Серед них були детально розглянуті DE PSO та BO. Порівняльний аналіз між РІ та ГА виявив кілька подібностей та відмінностей, які мають вирішальне значення для розуміння їхніх сильних сторін та контекстів застосування.

Наприклад, і РІ, і ГА є популяційними методами оптимізації, які передбачають колектив потенційних рішень, що «блукають» у просторі рішень у пошуках оптимального значення. Цей популяційний підхід дозволяє обом методам ефективно досліджувати широкий простір пошуку, тим самим збільшуючи ймовірність знаходження глобального оптимуму.

Однією з помітних подібностей є концепція кросовера, яка присутня як в DE, так і в ГА. У DE кросинговер використовується для об'єднання різних рішень-кандидатів для отримання нового потомства, полегшуючи обмін інформацією в популяції. Цей механізм має вирішальне значення для підтримки генетичного різноманіття і запобігання передчасній конвергенції на неоптимальних рішеннях.

Однак, незважаючи на цю схожість, між РІ та ГА існують суттєві відмінності. Наприклад, в той час як ГА часто покладається на двійкове кодування розв'язків, методи РІ, такі як DE, працюють безпосередньо з векторами дійсних чисел. Така пряма маніпуляція з дійсними числами спрощує алгоритм і покращує його застосовність до задач неперервної оптимізації.

Таким чином, огляд літератури підкреслює ефективність і універсальність алгоритмів РІ у різних областях. Порівняльний аналіз РІ і ГА висвітлює як спільні елементи, так і відмінні риси, надаючи цінну інформацію про їхні відповідні переваги. Ці висновки закладають міцний фундамент для подальшого детального обговорення конкретних методів РІ та їх застосування.

2 МОДИФІКАЦІЇ МЕТОДІВ Й АЛГОРИТМІВ РІ

В даному, розділі мовай піде про існуючі модифікації АМРІ. Також будуть запропоновані розроблені модифікації для покращення оптимізаційного методу DE.

2.1 Огляд існуючих модифікацій

В обчислювальній математиці кожна модифікація методу / алгоритму оптимізації передбачає роботу в кількох напрямках:

1) зменшення загальної кількості ітерацій для досягнення певної похибки. Позитивний ефект зменшення загальної кількості ітерацій призводить до зменшення загального процесорного часу розв'язання конкретної задачі або вирішення конкретного завдання. Це в свою чергу зменшує загальне навантаження на систему;

2) підвищення точності обчислень. Позитивний ефект від цього полягає у прискоренні збіжності методу / алгоритму оптимізації. Це ж у свою чергу знову призводить до зменшення кількості ітерацій, а значить – до зменшення процесорного часу, який необхідний для вирішення поставленої задачі або розв'язання конкретного завдання.

Пропонується ознайомитись з деякими модифікаціями алгоритмів та методів оптимізації, про які йшла мова в попередньому розділі.

2.1.1 Реконструйована диференціальна еволюція

Основною метою модифікації [7] є покращення продуктивності методу DE при розв'язуванні одноклітинних задач обмеженої оптимізації. Запропонована реконструйована диференціальна еволюція (RDE) об'єднує кілька передових стратегій з останніх варіантів DE для

створення більш ефективного та надійного алгоритму. Тож подивимось детальніше на складові даної модифікації.

Зовнішній архів використовується для збереження деяких батьківських особин з історії, які були замінені в поточній популяції. Такий підхід збалансовує розробку алгоритму та його дослідницьку продуктивність, підвищуючи його стабільність. Розмір архіву визначається як $A_r \cdot N$, де N - кількість особин у популяції.

Було введено нову стратегію мутації з назвою *DE/current-to-order-pbest/1*. Вона сортує і рекомбінує індивідів на основі їхніх значень пристосованості, генеруючи відповідні рішення:

$$v_{i,j}^{(k+1)} = x_{i,j}^{(k)} + F \cdot (x_{\text{ord pbest},j}^{(k)} - x_{i,j}^{(k)}) + F \cdot (x_{\text{ord median},j}^{(k)} - x_{\text{ord worst},j}^{(k)}) \quad (2.1)$$

де $x_{\text{ord pbest},j}^{(k)}$, $x_{\text{ord median},j}^{(k)}$, та $x_{\text{ord worst},j}^{(k)}$ представляють осіб на відповідних позиціях після сортування.

RDE використовує обидві стратегії *DE/current-to-pbest/1* та *DE/current-to-order-pbest/1*, з адаптивною стратегією для контролю співвідношення розподілу популяційних ресурсів між ними. Співвідношення розподілу визначається на основі середнього покращення придатності, спричиненого кожною стратегією:

$$\gamma_1^{(k+1)} = \begin{cases} 0.5 & \text{if } \omega_{m1}^{(k)} = \omega_{m2}^{(k-1)} = 0 \text{ or } k = 1 \\ \frac{\omega_{m1}^{(k)}}{\omega_{m1}^{(k)} + \omega_{m2}^{(k-1)}} & \text{otherwise} \end{cases} \quad (2.2)$$

Значення p для стратегії вибору «pbest» лінійно зменшується після кожної ітерації, що покращує збіжність за формулою:

$$p^{(k)} = p_{\max} \cdot \left(1 - 0.5 \cdot \frac{\text{nfes}}{\max \text{nfes}} \right) \quad (2.3)$$

RDE використовує *лінійне зменшення розміру популяції*, зменшуючи

розмір популяції по мірі оптимізації:

$$N^{(k+1)} = \text{round} \left(\left(\frac{N_{\min} - N_{\max}}{\max \text{ nfes}} \right) \cdot \text{nfes} + N_{\max} \right) \quad (2.4)$$

Стратегія збурення Коші використовується для заміни особин з ймовірністю p_r , що підвищує різноманітність популяції:

$$u_{i,j}^{(k)} = \begin{cases} v_{i,j}^{(k)} & \text{if } j = j_{\text{rand}} \text{ or } \text{rand} < Cr_i \\ \text{CauchyRand}(x_{i,j}^{(k)}, 0.1) & \text{if } \text{rand} < p_r \\ x_{i,j}^{(k)} & \text{otherwise} \end{cases} \quad (2.5)$$

Результати експериментів показують, що RDE ефективно розв'язує складні оптимізаційні задачі, демонструючи свій потенціал розвитку в галузі ройового інтелекту.

2.1.2 Приклад модифікації PSO алгоритму

Основною метою модифікацій запропонованих у статті [8] базового алгоритму PSO є підвищення ефективності та точності.

Модифікації спрямовані на вирішення критичних аспектів алгоритму PSO:

1) *Початкове налаштування популяції.* Простір пошуку розбито на окремі інтервали, причому початкове положення кожної частинки вибирається з іншого інтервалу. Це забезпечує більш рівномірний розподіл і зменшує ймовірність того, що алгоритм потрапить у пастку локальних оптимумів.

2) *Постійні прискорення та вага інерції.* Стратегія на основі нечіткої логіки використовується для динамічного налаштування констант прискорення (c_1 , c_2) та ваги інерції (w). Це дозволяє алгоритму більш ефективно досягати балансу на різних етапах процесу оптимізації.

3) *Розрахунок поточного положення.* Для оновлення положення

кожної частинки використовується байєсівський підхід. Цей метод використовує фільтр Калмана для поєднання поточного вимірювання, попередньої оцінки та апостеріорної оцінки, в результаті чого отримується статистично оптимальна оцінка положення частинки.

Описані модифікації з статті оперують наступними формулами:

1) Початкове налаштування популяції:

$$x_{pi}(0) \sim U \left(x_{min,i} + (p-1) \frac{x_{max,i} - x_{min,i}}{D}, x_{min,i} + p \frac{x_{max,i} - x_{min,i}}{D} \right) \quad (2.6)$$

2) Нечітка логіка для ваги інерції та констант прискорення:

$$w = \frac{2}{|2 - \phi - \sqrt{\phi^2 - 4\phi}|}, \quad \phi = c_1 + c_2 \quad (2.7)$$

3) Оновлення байєсівської позиції:

$$\mu_{X|L}(k) = \frac{\sigma_{L|X}^2 x_{p,i}(k-1) + \sigma_{pi}^2(k-1) l_{pi}}{\sigma_{L|X}^2 + \sigma_{pi}^2(k-1)} \quad (2.8)$$

$$\mu_{Y|B}(k) = \frac{x_{pi}(k-1) + \delta_{L|X}(k-1) l_{pi} + \delta_{B|Y}(k-1) b_i}{1 + \delta_{L|X}(k-1) + \delta_{B|Y}(k-1)} \quad (2.9)$$

Запропоновані варіанти PSO були оцінені за допомогою функцій Розенброка та Гриванка, а також протестовані на практичному кейсі з оптимізації електричних мереж. Результати показали, що модифіковані алгоритми перевершили базовий PSO як з точки зору точності, так і з точки зору ефективності. Зокрема, точність запропонованих методів була значно вищою, про що свідчать нижчі значення досягнутих цільових функцій. Крім того, нові методи продемонстрували кращу ефективність, вимагаючи меншої кількості ітерацій для досягнення оптимального рішення.

2.1.3 Приклад модифікації ВО

Алгоритм Adaptive Group Collaborative Artificial Bee Colony (AgABC) [9] має на меті покращити продуктивність традиційного алгоритму ВА шляхом усунення його обмежень у балансуванні між глобальною розвідкою та локальним пошуком. Основна мета модифікації AgABC - покращити здатність алгоритму знаходити оптимальні рішення в складних оптимізаційних задачах, підвищуючи тим самим його точність і стабільність.

Алгоритм AgABC вводить адаптивний механізм групової співпраці, де кілька стратегій пошуку з різними функціями застосовуються до різних фаз або популяцій.

Новий розв'язок $\mathbf{x}'_{mk}$ оновлюється на основі $\mathbf{x}_{mk}$ за наступною формулою:

$$\mathbf{x}'_{mk} = \mathbf{x}_{mk} + \text{rand}(-1,1) \times (\mathbf{x}_{mk} - \mathbf{x}_{nk}) \quad (2.10)$$

Ймовірність P_m для бджоли-спостерігача працювати навколо конкретної зайнятої бджоли визначається наступним чином:

$$P_m = \frac{\text{fitness}_m}{\sum_{m=1}^{SN} \text{fitness}_m} \quad (2.11)$$

Для покращення стратегій пошуку введено механізм перебору Леві:

$$\mathbf{x}'_{ik} = \mathbf{x}_{ik} + \text{Levy}(\lambda) \quad (2.12)$$

де $\text{Levy}(\lambda)$ - модель польоту Леві, визначена як:

$$\text{Levy}(\lambda) = \frac{\lambda \times \Gamma(1 + \lambda) \times \sin\left(\frac{\pi\lambda}{2}\right)}{\Gamma\left(\frac{1+\lambda}{2}\right) \times \lambda^{\frac{1}{\lambda}}} \quad (2.13)$$

Для стратегії пошуку з коротким кроком введено змінний коефіцієнт

кроку:

$$\mathbf{x}'_{ik} = \mathbf{x}_{ik} + \text{rand}(-1,1) \times (\mathbf{x}_{ik} - \mathbf{x}_{jk}) \times \left(\frac{1}{2} \times \text{iter} \times W \right) \quad (2.14)$$

де iter - номер ітерації, а W - вага, встановлена відповідно до конкретної задачі. .

2.2 Розроблені модифікації

У даній роботі пропонується кілька модифікацій описаних вище алгоритмів.

1) Модифікація положення найгіршого елемента популяції. Найгірший елемент популяції x_j^{worst} замінити на усереднене положення елемента по всій популяції x_j , $\forall j \in \overline{1; M}$ за такою формулою:

$$x_j^{Average} = \frac{1}{N} \sum_{i=1}^N x_{ij}, \forall j \in \overline{1; M}, \quad (2.15)$$

де N – загальна кількість елементів у популяції, а M – розмірність простору пошуку. Такий підхід можна виконувати на кожній ітерації або періодично – кожні K ітерацій.

2) Модифікація положення елементів популяції до найкращого з кроком h . Оновлення кожного елемента популяції виконується за формулою:

$$x_{ij} = x_{ij} + h \cdot \frac{x_{Best,j} - x_{ij}}{\|x_{Best,j} - x_{ij}\|}, \quad (2.16)$$

де x_{Best} – кращий вектор популяції за значенням цільової функції. Слід зазначити, що крок руху h можна визначати пропорційно (за лінійним законом) околу популяції, яка збігатиметься до глобального оптимуму з плином часу.

3) Зміна гіперпараметрів методів / алгоритмів оптимізації. Такий підхід передбачає зміну одного або кількох гіперпараметрів у процесі

роботи програми. Наприклад, у алгоритмі диференціальної еволюції можна параметр F замінити на випадкове дійсне число у діапазоні від 0 до 2 через K ітерацій. У ВА присутня залежність зміни параметра $\eta(n)$ від кількості ітерацій n . У такому випадку для оптимізації функцій великих розмірностей виникає проблема, яка полягає у завчасному зменшенні цього параметра. Для того, щоб таке не відбулось до знаходження глобального екстремуму функції або функціоналу такому випадку передбачається

$$\eta(n) = \eta_{max} \alpha^{[n/K]}, \quad (2.17)$$

де $[s]$ – ціла частина від числа s , K – період, який визначає те, через скільки ітерацій буде оновлене значення параметра η .

У методі DE пропонується брати гіперпараметр F випадковим числом від 0 до 2 кожні K ітерацій. Це обумовлено тим, що на кожній ітерації метод знаходить ближчі значення до глобального оптимуму функції або функціоналу.

Висновки до розділу 2

Підводячи підсумок для даного розділу слід зазначити те, що під час огляду існуючих модифікацій було виявлено, що більшість з них пропонують доволі серйозне втручання в базову структуру алгоритму. Вони представляють собою комплексні модифікації, які, як результат, дають новий алгоритм на основі базового, з якого був взятий основний принцип роботи. В той час як запропоновані модифікації спрямовані на пряме покращення роботи базового алгоритму при внесенні доволі невеликих змін в структуру роботи оптимізаційного алгоритму, результати яких можна буде проспостерігати в наступному розділі.

3 ТЕСТУВАННЯ ЗАПРОПОНОВАНИХ МОДИФІКАЦІЙ

У цьому розділі представлено результати експериментів оцінки ефективності отриманого модифікованого методу DE та базового. Для цього методи будуть перевірятись на базових тестових функціях та на прикладних задачах з оптимізації моделей ваги пружини та редуктора. Основна мета експериментів полягала у тестуванні базового та модифікованого методів DE для досягнення кращої продуктивності та точності.

3.1 Опис експериментів

Перед початком опису тестування необхідно зазначити середу в якій будуть проводитись експерименти.

Середа розробки методів і написання алгоритмів виконані на мові програмування Matlab R2017b (9.3.0).

Характеристики ноутбуку на якому проводились запуски: процесор – Intel(R) Core(TM) i5-7300HQ CPU 2.50GHz; ОЗУ – 16Gb; OS - Windows 10 Pro x64.

Написаний проєкт має в собі 2 модулі з описом моделей для тестування, 2 модулі з базовими тестовими функціями і 2 модулі з реалізованими класичним алгоритмом DE та модифікованим з зазначеними раніше модифікаціями.

Як зазначалось раніше, то оптимізаційні AMPI мають два варіанти критерію зупинки — досягання максимальної кількості ітерацій та досягання задані точності пошуку глобального оптимуму. В контексті обраних моделей було вирішено спиратись на знайдені оптимуми з статей де розглядались ці моделі для моделі редуктора [10] та пружини [11].

Будемо порівнювати результати роботи алгоритмів (базового та

модифікованого) за результатами запусків у кількості 5 штук. Тобто треба прогнати 5 раз 2 методи на 2х базових тестових функціях та 2х моделях. Критерієм зупинки є досягання точності $1e - 7$

За результатами запусків будемо порівнювати результати за критеріями:

- 1) Швидкість роботи алгоритму
- 2) Точність цільової функції
- 3) Кількість викликів обрахунку фітнесс функції

Швидкість алгоритмів в секундах буде замірятись починаючи відлік від моменту першого обрахунку цільової функції. Відповідно, швидкість також буде замірятись в кількості ітерацій необхідних алгоритму для досягнення заданої точності обрахунку глобального мінімуму моделі. Кількість викликів цільової функції заміряється підрахунком кількості сутностей, для яких обраховувалась цільова функція.

3.2 Тестування на базових функціях

Перед початком тестування отриманого модифікованого методу оптимізації DE, необхідно протестувати його показники на базових тестових функціях. Це мають бути тестові функції без обмежень та з обмеженнями. В контексті даної роботи будуть використовуватись функція Растрігіна без обмежень та функція Розенброка з функціональними обмеженнями кубічною кривою та прямою.

Дані тести проводяться для того, щоб впевнитись, що отримана модифікація успішно може себе демонструвати на простих відомих задачах. Завдяки цьому можна вже бачити різницю з базовим алгоритмом, для того щоб переконатись, що отримана модифікація не має гіршої результативності, ніж базовий алгоритм. Інакше отримана модифікація не буде виконувати поставленої перед нею задачі, а саме покращення результативності в порівнянні з базовим алгоритмом.

Перейдемо до опису базової тестової функції Растрігіна. На рисунку

3.1 можна побачити її візуалізацію.

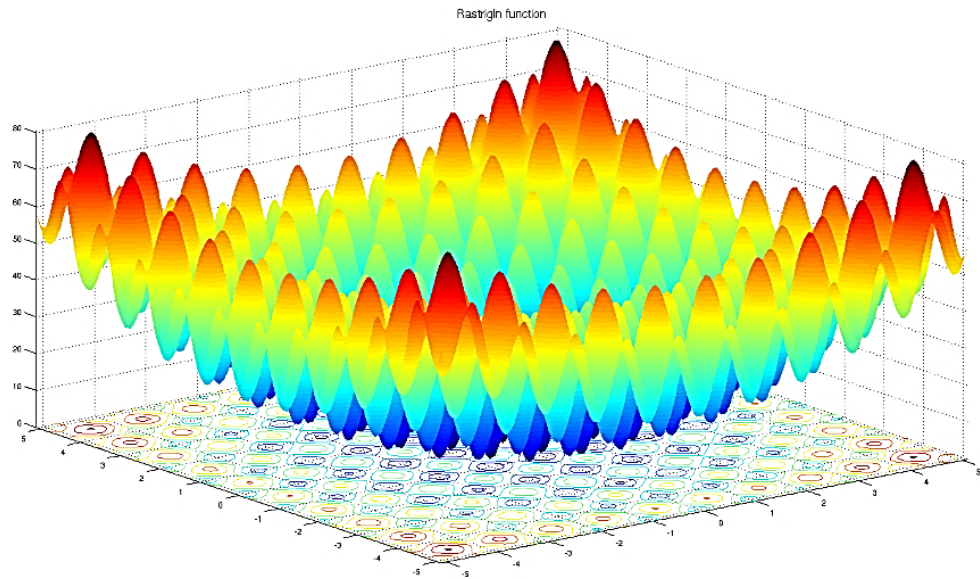


Рисунок 3.1 – Візуалізація функції Растрігіна

В свою чергу, її математичний вигляд є наступним:

$$f(X) = An + \sum_{i=1}^n (x_i^2 - A \cos(2\pi x_i)), \quad (3.1)$$

де $A = 10$, $x_i \in [-5.12, 5.12]$, $i = 1, \dots, n$.

Глобальний мінімум в точці $X^* = (x_1^*, x_2^*, \dots, x_n^*) = (0, \dots, 0)$.
 $f(X^*) = 0$.

Тестування буде виконуватись на даній(3.1) функції з заданою розмірністю 20. Це робиться з думок про те, що сучасні реальні задачі мають доволі великі розмірності простору рішень і тому, має сенс дивитись на поведінку методів на більших розмірностях.

Перейдемо до самого тестування, а саме подивимось на результативність базового методу DE (Таблиця 3.1).

Як можна помітити, то даний, базовий метод досягає збіжності приблизно на 1200й ітерації (Рисунок 3.2). Так, він досяг кінцевого, заданого результату з отримання точності цільової функцій, але затрата

Таблиця 3.1 – Результати роботи базового методу DE на моделі оптимізації ваги пружини $P = 0,7$, $F = 1$

Розмір популяції	Простір	К-сть ітерацій	Час роботи (с)	Виклики цільової ф-ції	Результат цільової ф-ції
$N = 200$	20	2248	13,440867	1349000	9,18571743
$N = 200$	20	2279	13,862111	1367600	8,99877647
$N = 200$	20	2250	13,738745	1350200	6,20469222
$N = 200$	20	2283	14,011194	1370000	7,43434043
$N = 200$	20	2239	13,683337	1343600	8,53200106

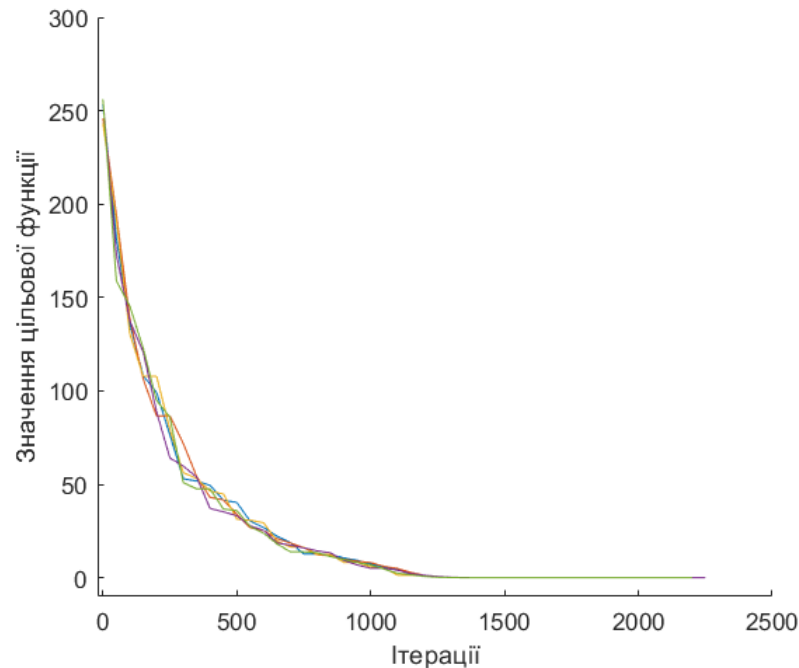


Рисунок 3.2 – Графік збіжності тестування базового методу DE на 20 вимірній функції Растрігіна

ресурсів і часу для цього доволі висока. А це не є гарним показником.

Аналогічно протестуємо модифікований метод DE.

Як можна помітити з незультатів таблиці 3.2 і графіку на рисунку 3.3, модифікований метод досягає заданої точності пошуку глобального екстремума в більше ніж два рази швидше, якщо говорити в контексті

Таблиця 3.2 – Результати роботи базового методу DE на моделі оптимізації ваги пружини $P = 0,7$, $F = rand(0, 2)$

Розмір популяції	Простір	К-сть ітерацій	Час роботи (с)	Виклики цільової ф-ції	Результат цільової ф-ції
$N = 200$	20	718	1,674132	287400	8,7591615738
$N = 200$	20	1106	2,840101	442600	9,1640060873
$N = 200$	20	531	1,502353	212600	8,7464627540
$N = 200$	20	1031	2,431870	412600	8,8600955905
$N = 200$	20	573	1,592970	229400	9,6510802904

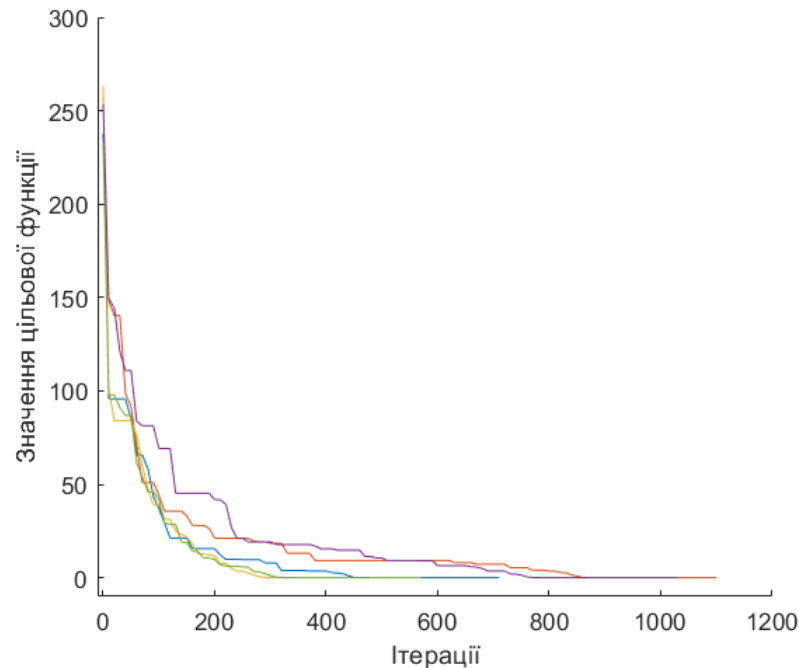


Рисунок 3.3 – Графік збіжності тестування модифікованого методу DE на 20 вимірній функції Растрігіна

кількості ітерацій. Якщо порівнювати за часом виконання, то тут вже мова йде про швидкість в 10 разів швидше за базовий метод. Також слід зазначити, що модифікований метод досягає збіжності приблизно на 800 ітерації в середньому.

Тепер перейдемо до другої базової функції для тестування, а саме

функції Розенброка з функціональними обмеженнями кубічною кривою та прямою (Рисунок 3.4).

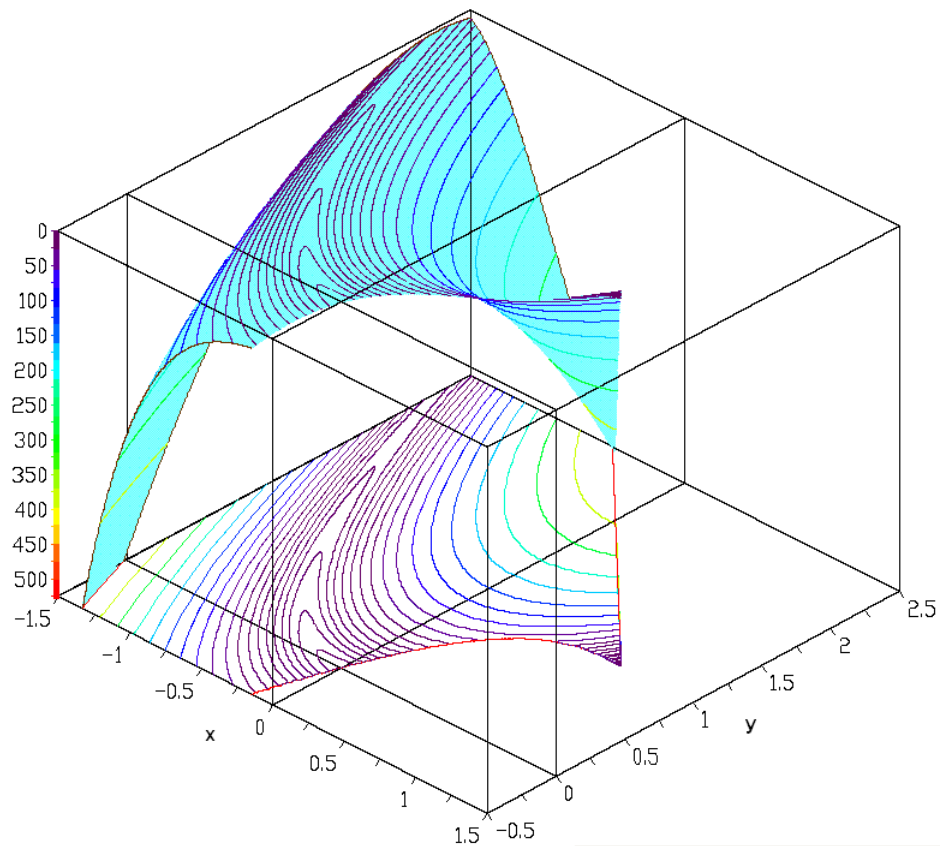


Рисунок 3.4 – Візуалізація функції Розенброка

Дана тестова функція має наступний математичний запис функції:

$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2 \rightarrow \min. \quad (3.2)$$

Функціональні обмеження:

$$(x - 1)^3 - y + 1 < 0; \quad x + y - 2 < 0. \quad (3.3)$$

Область пошуку:

$$x \in [-1.5; 1.5], \quad y \in [-0.5; 2.5]. \quad (3.4)$$

Оптимальне значення:

$$f_{\min}(1; 1) = 0. \quad (3.5)$$

Тестування на функції з функціональними обмеженнями необхідно для перевірки модифікації методу оптимізації для того, щоб перевірити здатність оптимізатора на пошук глобального екстремуму в обмеженому просторі пошуку. Адже більшість прикладних задач мають такого роду обмеження на певні параметри моделі, яка розглядається як ціль оптимізації.

Тож перейдемо до результатів тестування базового методу DE на функції Розенброка. Почнемо з базової моделі.

Таблиця 3.3 – Результати роботи базового методу DE на функції Розенброка $P = 0,7$, $F = 1$

Розмір популяції	Простір	К-сть ітерацій	Час роботи (с)	Виклики цільової ф-ції	Результат цільової ф-ції
$N = 200$	2	436	1,698789	261800	2.7015194
$N = 200$	2	493	2,190711	296000	1,5919489
$N = 200$	2	373	1,439477	224000	5,0429602
$N = 200$	2	409	1,592584	245600	3,6963825
$N = 200$	2	513	2,291005	308000	5,3662809

Як можна помітити по результатам з таблиці 3.3, в порівнянні з попереднім тестом (таблиця 3.1), метод впорався краще, але не треба забувати про те, що це все ще функція з двовимірним простором рішень.

Збіжність базового методу, виходячи з графіку на рисунку 3.5 досягається приблизно на 170й ітерації.

Перейдемо до результатів тестування модифікованого методу на даній функції Розенброка з функціональними обмеженнями.

Як можна помітити з таблиці 3.4, швидкість досягання кінцевого

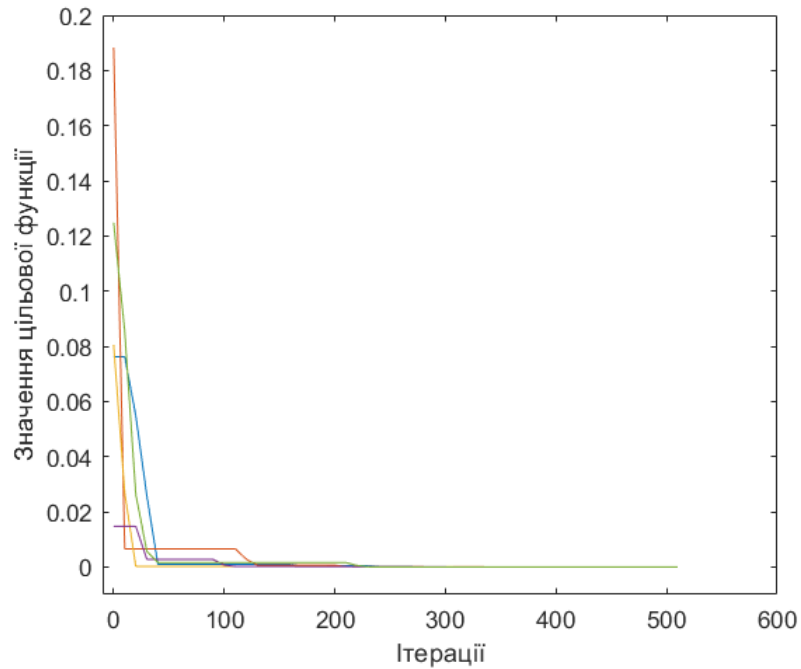


Рисунок 3.5 – Графік збіжності базового методу DE на ф-ції Розенброка

Таблиця 3.4 – Результати роботи базового методу DE на функції Розенброка $P = 0,7$, $F = rand(0, 2)$

Розмір популяції	Простір	К-сть ітерацій	Час роботи (с)	Виклики цільової ф-ції	Результат цільової ф-ції
$N = 200$	2	75	0,179772	30200	1,27558463
$N = 200$	2	81	0,186048	32600	3,14792937
$N = 200$	2	63	0,153731	25400	6.53572718
$N = 200$	2	70	0,166000	28200	8,82505588
$N = 200$	2	76	0,175912	30600	5.79790383

результату модифікованим методом DE склала 70 ітерацій в середньому, що в 6-7 разів швидше за базовий метод. В свою чергу, час який модифікований метод витратив на це – в 10 разів кращий.

Збіжність модифікованим методом DE, виходячи з результатів на графіку 3.6, досягається на 20й ітерації.

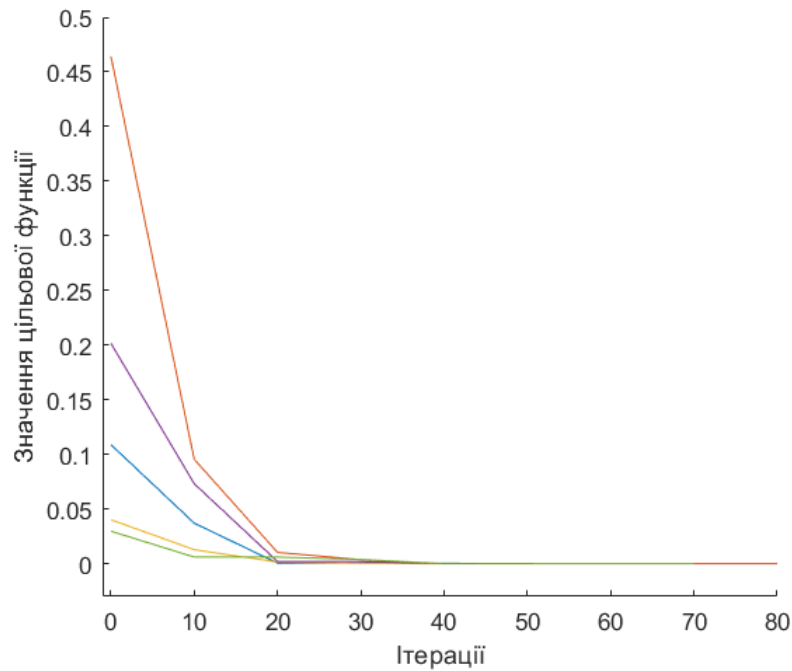


Рисунок 3.6 – Графік збіжності модифікованого методу DE на ф-ції Розенброка

3.3 Опис моделей для тестування

Як вже неодноразово було зазначено в роботі, АМРІ широко застосовуються в оптимізації параметрів багатовимірних прикладних задачах.

3.3.1 Модель конструкції редуктора

Першою моделлю для тестування є "Модель конструкції редуктора"[10]. Задача полягає у мінімізації ваги редуктора. Конструкцію предмета моделювання можна побачити на рисунку 3.7.

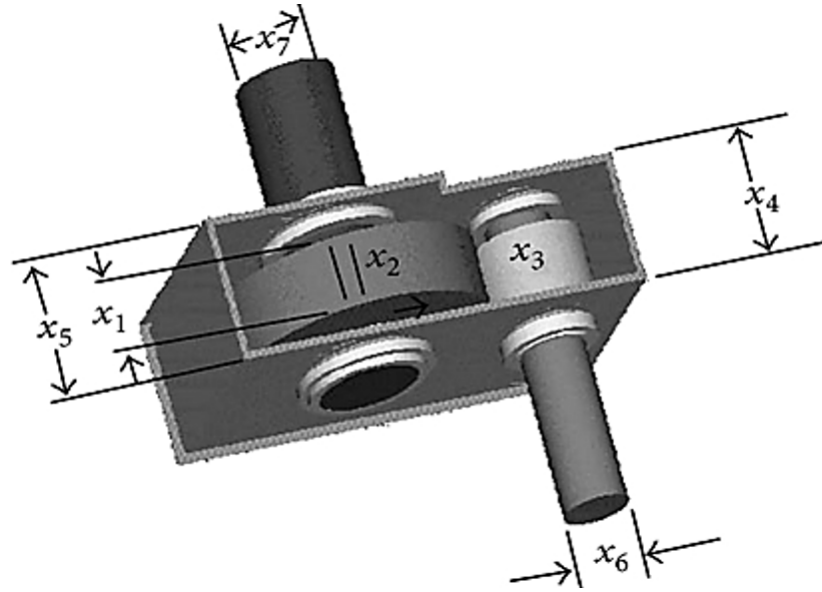


Рисунок 3.7 – Конструкція редуктора

Цільова функція зазначеної моделі виглядає наступним чином:

$$\begin{aligned}
 f(\bar{x}) = & 0.7854x_1x_2^2 \times \\
 & \times (3.3333x_3^2 + 14.9334x_3 - 43.0934) - \\
 & - 1.508x_2(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3) + \\
 & + 0.7854(x_4x_6^2 + x_5x_7^2) \rightarrow \min,
 \end{aligned} \tag{3.6}$$

де x_1 – ширина обшивки; x_2 – модуль колеса; x_3 – кількість зубів колеса (ціла змінна); x_4 – довжина першого валу між підшипниками; x_5 – довжина другого валу між підшипниками; x_6 – діаметр першого валу; x_7 – діаметр другого валу.

В матеріалах до моделі [10] надається інформація стосовно верхніх і

нижніх меж параметрів моделі які мають наступні значення:

$$\begin{aligned}
 2,6 &\leq x_1 \leq 3,6, 0,7 \leq x_2 \leq 0,8 \\
 17 &\leq x_3 \leq 28, 7,3 \leq x_4 \leq 8,3 \\
 7,8 &\leq x_5 \leq 8,3, 2,9 \leq x_6 \leq 3,9, \\
 5 &\leq x_7 \leq 5,5
 \end{aligned} \tag{3.7}$$

Також під час виконання завдання слід враховувати функціональні обмеження на параметри моделі. Вони включають обмеження на напруження згину зубів шестерні, поверхневі напруження, поперечні прогини двох валів під дією переданої сили, а також напруження у валах. Їх представлення має наступне формулювання:

$$\begin{aligned}
 g_1(\bar{x}) &= \frac{27}{x_1 x_2^2 x_3} - 1 \leq 0, \\
 g_2(\bar{x}) &= \frac{397,5}{x_1 x_2^2 x_3^2} - 1 \leq 0, \\
 g_3(\bar{x}) &= \frac{1,93 x_4^3}{x_2 x_3^2 x_6^4} - 1 \leq 0, \\
 g_4(\bar{x}) &= \frac{1,93 x_5^3}{x_2 x_3 x_7^4} - 1 \leq 0, \\
 g_5(\bar{x}) &= \frac{1}{110 x_6^3} \sqrt{\left(\frac{745 x_4}{x_2 x_3}\right)^2 + \frac{16,9}{10^{-6}}} - 1 \leq 0, \\
 g_6(\bar{x}) &= \frac{1}{85 x_7^3} \sqrt{\left(\frac{745 x_5}{x_2 x_3}\right)^2 + \frac{157,5}{10^{-6}}} - 1 \leq 0, \\
 g_7(\bar{x}) &= \frac{x_2 x_3}{40} - 1 \leq 0, \\
 g_8(\bar{x}) &= \frac{5 x_2}{x_1} - 1 \leq 0, \\
 g_9(\bar{x}) &= \frac{x_1}{12 x_2} - 1 \leq 0, \\
 g_{10}(\bar{x}) &= \frac{1,5 x_6 + 1,9}{x_4} - 1 \leq 0, \\
 g_{11}(\bar{x}) &= \frac{1,1 x_7 + 1,9}{x_5} - 1 \leq 0.
 \end{aligned} \tag{3.8}$$

3.3.2 Задача мінімізації ваги пружини

Другою моделлю для тестування буде виступати "Модель конструкції пружини розтягування/стиснення"[11]. Тут задача полягає в тому, щоб мінімізувати вагу пружини. Конструкцію пружини можна побачити на рисунку 3.8

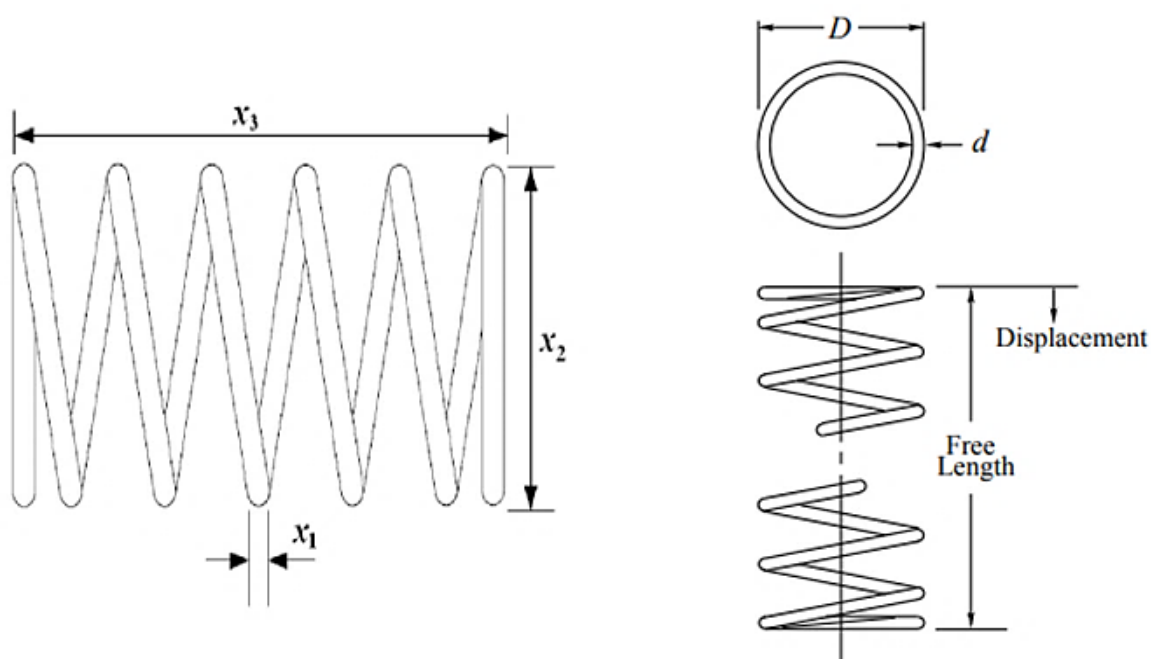


Рисунок 3.8 – Конструкція пружини

Дана модель має цільову функцію, яка представлена наступним чином

$$f(\vec{x}) = (x_3 + 2)x_2x_1^2 \rightarrow \min. \quad (3.9)$$

де x_1 – діаметр дроту; x_2 – середній діаметр котушки; x_3 – кількість активних котушок(ціле значення).

В матеріалах до моделі [11] надається інформація стосовно верхніх і

нижніх меж параметрів моделі які мають наступні значення:

$$\begin{aligned} 0,005 \leq x_1 \leq 2,0, 0,25 \leq x_2 \leq 1,3 \\ 2,0 \leq x_3 \leq 15 \end{aligned} \quad (3.10)$$

В свою чергу дана модель має чотири функціональні обмеження, які мають наступний вигляд:

1) Відхилення:

$$g_1(\vec{x}) = 1 - \frac{x_2^3 x_3}{7,178 x_1^4} \leq 0; \quad (3.11)$$

2) Напруга стиснення:

$$g_2(\vec{x}) = \frac{4x_2^2 - x_1 x_2}{12,566(x_2 x_3^4) - x_1^4} + \frac{1}{5,108 x_2^2} - 1 \leq 0; \quad (3.12)$$

3) Частота:

$$g_3(\vec{x}) = 1 - \frac{140,45 x_1}{x_2^2 x_3} \leq 0; \quad (3.13)$$

4) Зовнішній діаметр:

$$g_4(\vec{x}) = \frac{x_2 + x_1}{1,5} - 1 \leq 0. \quad (3.14)$$

3.4 Результати тестування на прикладних моделях

Почнемо з моделі оптимізації ваги пружини.

Запустимо базовий алгоритм 5 раз і занесемо результати в таблицю.

Відповідно до кожного з запусків алгоритм видав координати кращого положення сутності за час роботи алгоритму, тобто оптимізовані параметри моделі.

Також можемо подивитись на загальний графік збіжності п'яти експериментів.

З нього (рисунок 3.9) можемо побачити, що всі експерименти наблизились до мінімуму простору рішень за меншу кількість ітерацій,

Таблиця 3.5 – Результати роботи базового методу DE на моделі оптимізації ваги пружини $P = 0,7$, $F = 1$

Розмір популяції	Простір	К-сть ітерацій	Час роботи (с)	Виклики цільової ф-ції	Результат цільової ф-ції
$N = 200$	3	4383	19,289184	2630000	0,012665976
$N = 200$	3	4068	21,38259	2441000	0,012665953
$N = 200$	3	4396	20,616931	2637800	0,012665959
$N = 200$	3	3593	20,496191	2156000	0,012665787
$N = 200$	3	5181	24,342347	3108800	0,012665639

Таблиця 3.6 – Знайдені базовим методом DE параметри моделі пружини

X_1	X_2	X_3
0,051609	0,354784452	11,40377
0,05185	0,360593618	11,06549
0,051773	0,358730044	11,17236
0,051663	0,356093864	11,32603
0,051815	0,359741448	11,11396

ніж досягли необхідної точності. Це значення варюється біля 600 ітерацій. Базовий алгоритм більшість часу витратив на досягання заданої точності в умові нупинки. Нагадаю, що це є $1e - 7$. З цього можна зробити проміжний висновок, що базовий метод погано по часу і ресурсоемності справляється з досяганням більш точних результатів, які впливають на якість загальної роботи а відповідно і оцінки методу.

Тепер зберемо дані з модифікованого методу.

Відповідні знайдені параметри модифікованим методом DE

Подивимось на відповідний графік збіжності для модифікованого методу.

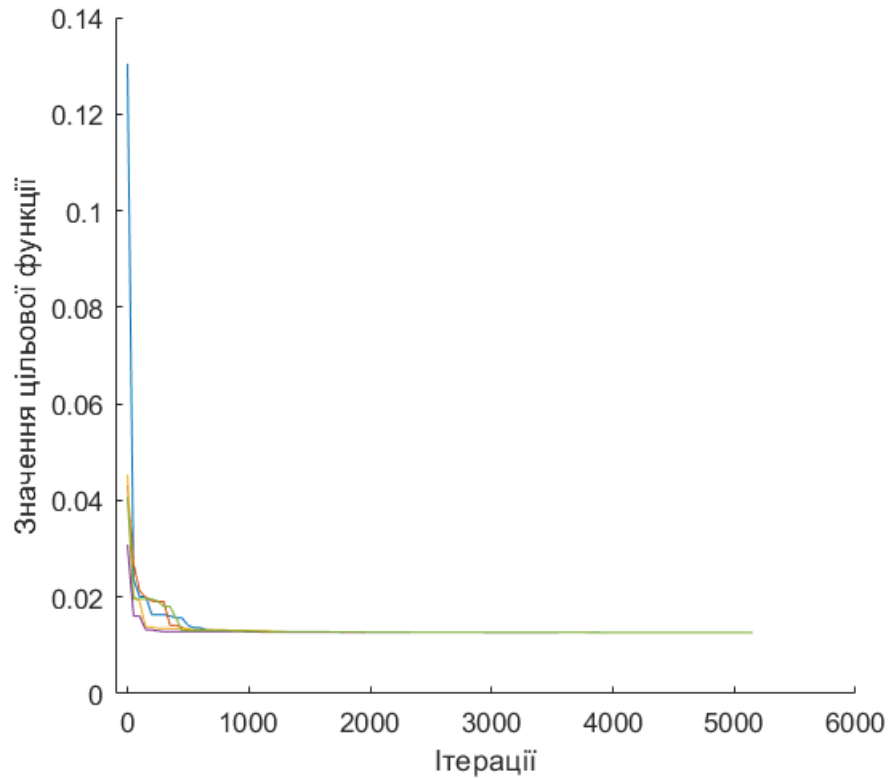


Рисунок 3.9 – Збіжність експериментів базового методу на моделі оптимізації ваги пружини

Таблиця 3.7 – Результати роботи модифікованого методу DE на моделі оптимізації ваги пружини $P = 0,7$, $F = rand(0, 2)$

Розмір популяції	Простір	К-сть ітерацій	Час роботи (с)	Виклики цільової ф-ції	Результат цільової ф-ції
$N = 200$	3	217	0,538156	87000	0,012655974
$N = 200$	3	377	0,894281	151000	0,012655979
$N = 200$	3	180	0,493545	72200	0,012655867
$N = 200$	3	369	0,889519	147800	0,012655994
$N = 200$	3	158	0,412776	63400	0,012655676

Розбираючи графік модифікованого методу (рисунок 3.10) можемо побачити, що ситуація схожа на класичний метод, але лише в контексті того, що метож, відносно загальної картини, швидко сходиться, і більшу

Таблиця 3.8 – Знайдені модифікованим методом DE параметри моделі пружини

0,05178008	0,358902824	11,16243578
0,051499509	0,352173292	11,56051513
0,051740145	0,357942354	11,21802419
0,051489457	0,351934552	11,57502818
0,051589147	0,354318576	11,43137984

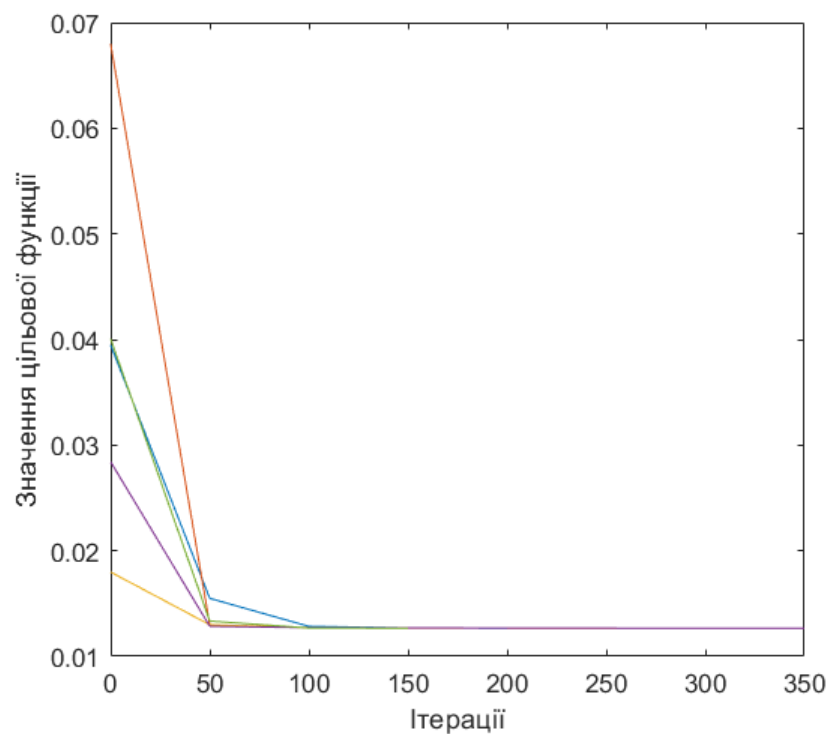


Рисунок 3.10 – Збіжність експериментів модифікованого методу на моделі оптимізації ваги пружини

частину від загальної кількості ітерацій витрачає на «уточнення» знайденого значення до необхідної точності, яка вимагається в критерії зупинки.

Виходячи з отриманих проміжних результатів можемо побачити те, що модифікований метод DE впорався в рази швидше, ніж базовий в контексті застосування на моделі оптимізації ваги пружини. Це можна

помітити з різниці в кількості необхідних ітерацій, відповідно менша кількість викликів обрахунку цільової функції і відповідно менший загальний час збіжності методу.

Тепер перейдимо до тестування на другій моделі, а саме «Модель оптимізації ваги редуктора».

Експерименти проводяться з аналогісним принципом, що й для попередньої моделі.

В таблиці 3.9 наведені результати роботи базового методу. Можна помітити, що хоч ця модель має більшу розмірність в порівнянні з моделлю пружини, але попри це з цією задачею, базовий метод впорався в рази швидше.

Таблиця 3.9 – Результати роботи базового методу DE на моделі оптимізації ваги редуктора $P = 0,7$, $F = 1$

Розмір популяції	Простір	К-сть ітерацій	Час роботи (с)	Виклики цільової ф-ції	Результат цільової ф-ції
$N = 200$	7	693	5,580056	416000	2996,34816507
$N = 200$	7	705	5,188952	423200	2996,34816500
$N = 200$	7	702	5,269009	421400	2996,34816509
$N = 200$	7	771	5,926614	462800	2996,34816506
$N = 200$	7	686	5,147230	411800	2996,34816508

Далі перейдемо до тестування модифікованого методу DE на моделі оптимізації ваги редуктора.

Як можна помітити з результатів, метод працює в декілька разів швидше, в порівнянні з базовим методом. Це можна помітити за зменшеними кількістю ітерації для досягання заданої точності. Через це менша кількість викликів обрахунку цільової(фітнес) функції і відповідно менше час роботи методу взагалом. А значить досягнута поставлена мета для розроблених модифікацій. А саме більшити ефективність методу DE та зменшити час виконання, при цьому

Таблиця 3.10 – Знайдені базовим методом DE параметри моделі редуктора

X_1	X_2	X_3	X_4	X_5	X_6	X_7
3,5	0,7	17	7,3	7,8	3,3502146662	5,286683229
3,5	0,7	17	7,3	7,8	3,3502146661	5,286683229
3,5	0,7	17	7,3	7,8	3,3502146661	5,286683229
3,5	0,7	17	7,3	7,8	3,3502146662	5,286683229
3,5	0,7	17	7,3	7,8	3,3502146661	5,286683229

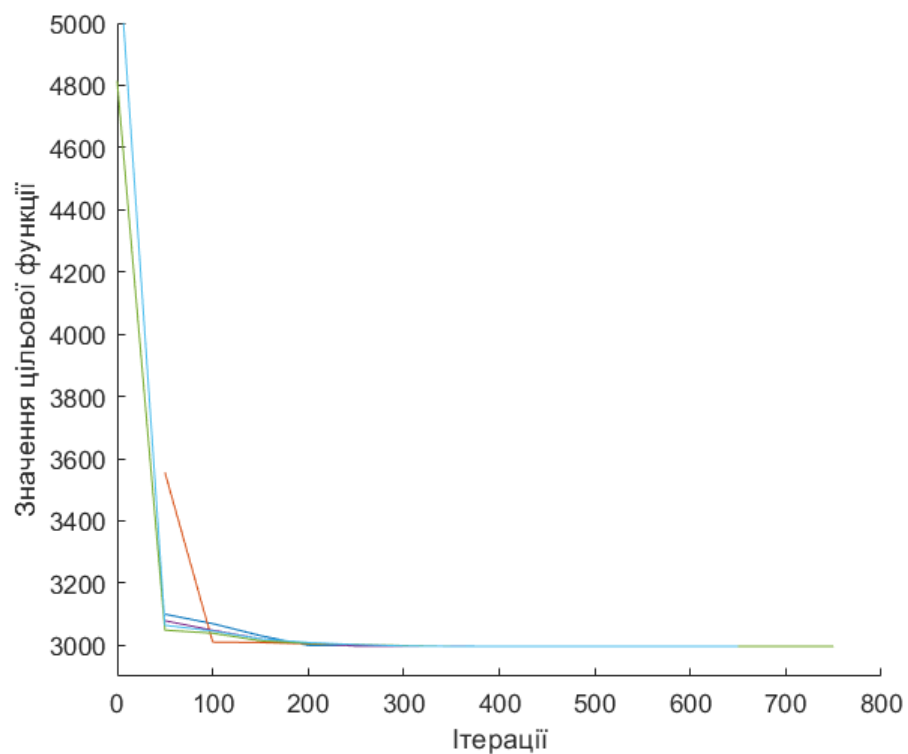


Рисунок 3.11 – Збіжність експериментів базового методу на моделі оптимізації ваги редуктора

перевіряючи результати видачі методів DE з заданням точності для встановлення критерію зупинки.

Також можна простежити певну «схожість» між усіма експериментами, проведеними на обох моделях і обома методами. А саме, що метод досить швидко сходиться до значення, близького до

Таблиця 3.11 – Результати роботи модифікованого методу DE на моделі оптимізації ваги редуктора $P = 0,7$, $F = random(0, 2)$

Розмір популяції	Простір	К-сть ітерацій	Час роботи (с)	Виклики цільової ф-ції	Результат цільової ф-ції
$N = 200$	7	238	1,254037	95400	2996,34816507
$N = 200$	7	251	1,301812	100600	2996,34816509
$N = 200$	7	259	1,338412	103800	2996,34816509
$N = 200$	7	258	1,298142	103400	2996,34816505
$N = 200$	7	280	1,488943	112200	2996,34816508

Таблиця 3.12 – Знайдені модифікованим методом DE параметри моделі редуктора

X_1	X_2	X_3	X_4	X_5	X_6	X_7
3,5	0,7	17	7,3	7,8	3,3502146660	5,2866832298
3,5	0,7	17	7,3	7,8	3,3502146661	5,2866832297
3,5	0,7	17	7,3	7,8	3,3502146663	5,2866832298
3,5	0,7	17	7,3	7,8	3,3502146661	5,2866832297
3,5	0,7	17	7,3	7,8	3,3502146661	5,2866832297

оптимального, після чого виконується більш локальний перебір простору розв'язків, щоб якомога точніше визначити значення, близьке до глобального мінімуму. Така швидка збіжність свідчить про високу ефективність методу в навігації по простору розв'язків для виявлення перспективних областей на ранній стадії процесу оптимізації. Згодом локальний пошук уточнює це початкове наближення, гарантуючи, що остаточний розв'язок буде не тільки близьким до оптимального, але й надійним і точним.

Зрештою, не слід забувати, що АМРІ побудовані так, щоб знаходити значення, дуже близьке до глобального. Притаманна цим алгоритмам конструкція підкреслює їхню силу в точному налаштуванні рішень у

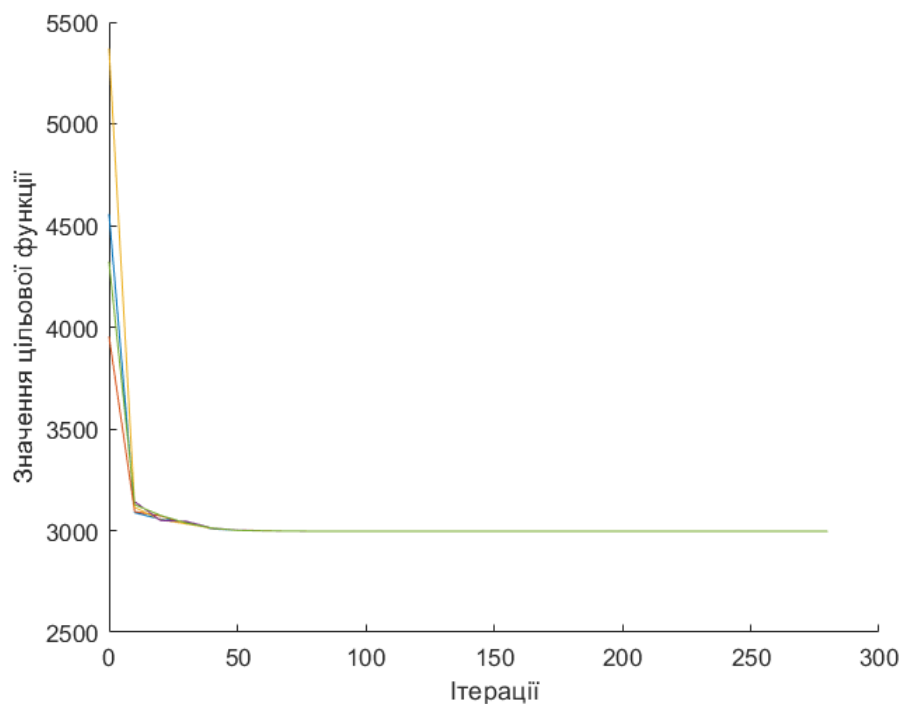


Рисунок 3.12 – Збіжність експериментів модифікованого методу на моделі оптимізації ваги редуктора

вузькому діапазоні, що робить їх особливо ефективними для складних задач, де глобальна оптимізація має важливе значення. Цей двофазний підхід - початкова швидка збіжність з наступною ретельною локальною оптимізацією - демонструє стійкість і надійність методів, тим самим посилюючи їхню застосовність до широкого кола прикладних та інженерних проблем.

Крім того, така поведінка підкреслює адаптивність AMRI, оскільки вони ефективно балансують між розвідкою і розробкою в межах простору рішень. Початкова фаза використовує глобальну розвідку, щоб уникнути неоптимальних регіонів, тоді як наступна фаза локального пошуку забезпечує детальну розробку перспективних ділянок, що призводить до високоточного наближення до глобального мінімуму. Ця характеристика робить AMPI особливо цінним у сценаріях, де точність і надійність є критично важливими, пропонуючи надійне рішення для складних оптимізаційних задач.

Висновки до розділу 3

За результатами третього розділу, було протестовано базовий і модифікований методи DE на базових тестових функціях. А саме 20и-вимірній функції Растрігіна без функціональних обмежень і 2-вимірній функції Розенброка з функціональними обмеженнями кубічною кривою та прямою. За результатами цих тестів, можна переконатись, що модифікований метод справляється з базовими адачами в 10 разів швидше по часу роботи алгоритму.

Після цього, ознайомились з прикладними математичними моделями отримізації ваги пружини та оптимізації ваги редуктора. З відповідних статей [10, 11] було визначено глобальні мінімуми для даних моделей, на які спирались при тестуванні базового та модифікованого оптимізаційних методів DE.

Результати експериментів показали, що завдяки модифікаціям, які буди описані в другому розділі, модифікований метод DE розв'язує поставлені прикладні задачі оптимізації в рази швидше й ефективніше за базовий метод. Також було визначно, що базовий і модифікоманий методи більшу частину ітерацій і, відповідно, часу витрачають на те, щоб досягти більшої точності результату оптимізації математичної моделі.

ВИСНОВКИ

Ройовий інтелект - це видатний напрям у штучному інтелекті, натхненний колективною поведінкою децентралізованих систем, що самоорганізуються, які зазвичай зустрічаються в природі. У дисертації досліджується застосування АМРІ у вирішенні складних прикладних та інженерних проблем, підкреслюється адаптивність і надійність цих методів. АМРІ, такі як оптимізація рою частинок (PSO), диференціальна еволюція (DE) та бджолині алгоритми (BO), ефективно вирішують різноманітні оптимізаційні задачі, від маршрутизації телекомунікаційних мереж до покращення проектних параметрів фізичних структур.

Це дослідження робить значний внесок у галузь оптимізації, розробляючи та тестуючи нові модифікації методу диференційної еволюції (DE). Ці модифікації оцінюються порівняно з базовим методом DE, демонструючи покращену продуктивність у вирішенні прикладних моделей. Модифікації усувають поширені проблеми, такі як застрягання в локальних екстремумах та тривалий час виконання, що є критичним для практичного застосування. Також слід зазначити, що під час тестування було помічено, що обидва варіанти алгоритмів витрачають найбільшу кількість ітерацій для досягання необхідної точності пошуку глобального оптимуму. А значить можна додати до висновку, що слід фокусувати майбутні дослідження на пришвидшенні локального пошуку методу DE.

Методологія дослідження є суворою, включаючи детальний аналіз існуючої літератури, детальні описи АМРІ, систематичне тестування та порівняння результатів. Підхід дослідження забезпечує надійність та застосовність отриманих результатів до широкого спектра задач оптимізації.

За результатами роботи відкривається кілька напрямків для майбутніх досліджень, включаючи інтеграцію АМРІ з іншими методами оптимізації, такими як глибоке навчання та еволюційні алгоритми, для

створення більш потужних гібридних систем. Крім того, перспективними є дослідження застосування РІ у нових галузях, таких як автономні системи, розумні міста та біоінформатика.

Ширший вплив цього дослідження полягає в його потенціалі покращити ефективність та результативність вирішення різноманітних інженерних та прикладних задач. Завдяки покращенню продуктивності АМРІ, ця робота сприяє розвитку методів оптимізації, що може привести до кращого розподілу ресурсів, удосконалення процесів проектування та більш ефективних операційних стратегій у різних галузях.

На завершення, робота надає цінні висновки щодо АМРІ, представляючи інноваційні модифікації, які підвищують продуктивність існуючих оптимізаційних технік. Практичні наслідки та методологічна строгість дослідження підкреслюють його значимість у галузі прикладної математики та оптимізації.

ПЕРЕЛІК ПОСИЛАНЬ

1. M.V. Dileep, and Surekha Kamath, and Vishnu G. Nair, "Particle Swarm Optimization Applied to Ascent Phase Launch Vehicle Trajectory Optimization Problem *Procedia Computer Science*, vol. 54, pp. 516-522, 2015. ISSN: 1877-0509. doi: <https://doi.org/10.1016/j.procs.2015.06.059>. URL: <https://www.sciencedirect.com/science/article/pii/S1877050915013836>.
2. Erik Cuevas, and Daniel Zaldivar, and Marco Perez, and Humberto Sossa, and Valentin Osuna, "Block matching algorithm for motion estimation based on Artificial Bee Colony (ABC) 2014. arXiv:1407.0061 [cs.NE].
3. Prasert Sriboonchandr, and Nuchsa Kriengkarakot, and Preecha Kriengkarakot, "Improved Differential Evolution Algorithm for Flexible Job Shop Scheduling Problems 2019.
4. Fernández Martínez, J.L., and García Gonzalo, E., "The Generalized PSO: A New Door to PSO Evolution 2008, pp. 1-15, <https://doi.org/10.1155/2008/861275>.
5. Verma, B.K., and Kumar, D., "A review on Artificial Bee Colony algorithm 2013, <https://doi.org/10.14419/ijet.v2i3.1030>.
6. Dey, N., "Enhancing Differential Evolution Algorithm for Solving Continuous Optimization Problems 2013, pp. 1-7, <http://urn.kb.se/resolve?urn=urn:nbn:se:mdh:diva-33466>.
7. Sichen Tao, and Ruihan Zhao, and Kaiyu Wang, and Shangce Gao, "An Efficient Reconstructed Differential Evolution Variant by Some of the Current State-of-the-art Strategies for Solving Single Objective Bound Constrained Problems 2024. arXiv:2404.16280 [cs.NE].
8. Silvano Chiaradonna, and Felicita Di Giandomenico, and Nadir Murru, "On enhancing efficiency and accuracy of particle swarm optimization algorithms 2020. arXiv:2004.06326 [math.OC].
9. Haiquan Wang, and Hans-Dietrich Haasis, and Panpan Du, and Xiaobin Xu, and Menghao Su, Shengjun Wen, and Wenxuan Yue, and Shanshan

Zhang, "Adaptive Group Collaborative Artificial Bee Colony Algorithm 2021. arXiv:2112.01215 [cs.NE].

10. Ming-Hua Lin, and Jung-Fa Tsai, and Nian-Ze Hu, and and Shu-Chuan Chang, "Design Optimization of a Speed Reducer Using Deterministic Techniques Mathematical Problems in Engineering, vol. 2013, 2013.

11. Leticia Cagnina, and S. C. Esquivel, and Carlos A. Coello Coello, "Solving Engineering Optimization Problems with the Simple Constrained Particle Swarm Optimizer 2008.

ДОДАТОК А ТЕКСТИ ПРОГРАМ

А.1 Базовий алгоритм DE

```

function DE_Main

clc; clear all; format long;

N = 2;

% xmin = -5.12; xmax = 5.12;
xmin = [-1.5, -0.5]; xmax = [1.5, 2.5];
% xmin = [0.05, 0.25, 2]; xmax = [2, 1.3, 15];
% xmin = [2.6, 0.7, 17, 7.3, 7.8, 2.9, 5];
% xmax = [3.6, 0.8, 28, 8.3, 8.3, 3.9, 5.5];
% xmax = [2, 1.3, 15]; xmin = [0.05, 0.25, 2];

PopSize = 200;

P = 0.7;

F = 1;

x = rand(PopSize,N).*(xmax-xmin) + xmin;

MaxIter = 10000;

% ArticleGlobal = 2996.348165;

% ArticleGlobal = 0.012665;

Ff = []; It = [];

It = [It 0];

goalFuncCals = 0;

tic;

goalFuncCals = goalFuncCals + PopSize;

fObj = RosenbrokFunc(x);

[Best_F, i] = min(fObj);

Ff = [Ff Best_F];

```

```

for iter = 1:MaxIter
    for i=1:PopSize
        ind = randperm(PopSize);
        j = ind(1:3);
        %           F = 2*rand;
        v = x(j(1),:) + F*(x(j(2),:)-x(j(3),:));
        x = max(xmin, x);  x = min(xmax, x);
        r = rand(1,N);
        ind = find(r<P);
        v(ind) = x(i,ind);
        goalFuncCals = goalFuncCals + 2;
        if RosenbrokFunc(v) < RosenbrokFunc(x(i,:))
            x(i,:) = v;
        end
    end
end

    goalFuncCals = goalFuncCals + length(x);
    fObj = RosenbrokFunc(x);
    [Best_F, i] = min(fObj);
    [Worst_F, Worst_i] = max(fObj);

    if mod(iter, 10) == 0
        iter
        Best_F
        Best_X = x(i,:)
%   fprintf('loss %8.12d \n', abs(ArticleGlobal - Best_F))
        Num_Iter = iter;
        It = [It Num_Iter];
        Ff = [Ff Best_F];
    end

%           if abs(Best_F - ArticleGlobal) < 1e-7

```

```

%         if iter == MaxIter
            if abs(Best_F) < 1e-7
                fprintf("iter %5d \n", iter);
                fprintf("goalFuncCals %5d \n", goalFuncCals);
%         fprintf('loss %8.12d \n', ArticleGlobal - abs(Best_F))
                fprintf('abs %8.12d \n', abs(Best_F))
                break;
            end
% %     end
end
toc;
plot(It, Ff);
[Best_F, i] = min(f(x));
Best_X = x(i,:);
Best_F

```

A.2 Модифікований алгоритм DE

```

function DE_Main_gpt_modified
    clc; clear all; format long;
    N = 2;

%     xmin = -5.12; xmax = 5.12;
    xmin = [-1.5, -0.5]; xmax = [1.5, 2.5];
%     xmin = [2.6, 0.7, 17, 7.3, 7.8, 2.9, 5];
%     xmax = [3.6, 0.8, 28, 8.3, 8.3, 3.9, 5.5];
%     xmax = [2, 1.3, 15]; xmin = [0.05, 0.25, 2];

    PopSize = 200;
    P = 0.7;
%     F = 0.8;

```

```

%     ArticleGlobal = 2996.348165;
%     ArticleGlobal = 0.012665;

x = rand(PopSize, N) .* (xmax - xmin) + xmin;
MaxIter = 10000;
Ff = []; It = [];
goalFuncCals = 0;
BestSolution = Inf;
BestX = [];

tic;
fObj = RosenbrokFunc(x);
goalFuncCals = goalFuncCals + PopSize;
[BestSolution, BestIdx] = min(fObj);
BestX = x(BestIdx, :);
It = [It 0];
Ff = [Ff BestSolution];

for iter = 1:MaxIter
    F = 2*rand;
    for i = 1:PopSize
        indices = randperm(PopSize, 3);
        while any(indices == i)
            indices = randperm(PopSize, 3);
        end

        v = x(indices(1), :) + F * (x(indices(2), :) - x(indices(3), :));
        v = min(max(v, xmin), xmax);

        r = rand(1, N);
        cross_points = r < P;
    end
end

```

```

        if ~any(cross_points)
            cross_points(randi(N)) = true;
        end

        trial = x(i, :);
        trial(cross_points) = v(cross_points);

        goalFuncCals = goalFuncCals + 1;

        % Selection
        if RosenbrokFunc(trial) < RosenbrokFunc(x(i, :))
            x(i, :) = trial;
        end
    end

    fObj = RosenbrokFunc(x);
    goalFuncCals = goalFuncCals + PopSize;

    [CurrentBestF, BestIdx] = min(fObj);
    [WorstF, WorstIdx] = max(fObj);

    if CurrentBestF < BestSolution
        BestSolution = CurrentBestF;
        BestX = x(BestIdx, :);
    end

    x(WorstIdx, :) = mean(x);

    if mod(iter, 10) == 0
        disp(['Iteration: ', num2str(iter), ' Best F: ', num2str(BestSolution)]);
    end

```

```

%     fprintf('Loss: %8.12d \n', abs(BestSolution - ArticleGlobal))
        It = [It iter];
        Ff = [Ff BestSolution];
    end

%         if abs(BestSolution - ArticleGlobal) < 1e-7
    if abs(BestSolution) < 1e-7
        fprintf('Converged at iteration %d\n', iter);
        fprintf('Function evaluations: %d\n', goalFuncCals);
        fprintf('Best solution: %.12f\n', BestSolution);
        break;
    end
end

toc;
plot(It, Ff);
Best_X = BestX
Best_F = BestSolution
end

```

A.3 Функція Растригіна

```

function z = f(x)
A = 10; N = numel(x(1,:));
z = A*N + sum(x.^2,2) - A*sum(cos(2*pi*x),2);

```

A.4 Функція Розенброка

```

function y = RosenbrokFunc(x)
y = (1 - x(:,1)).^2 + 100 .* (x(:,2) - x(:,1).^2).^2;

```

```

condition = ((x(:,1) - 1).^3 - x(:,2) + 1 >= 0) & (x(:,1)...
+ x(:,2) - 2 >= 0);

y(condition) = +Inf;

end

```

A.5 Модель редуктора

```

function y = Reduktor(x)

y = 0.7854 .* x(:,1) .* x(:,2).^2 .* (3.3333 .* x(:,3).^2...
+ 14.9334 .* x(:,3) - 43.0934) - 1.508 .* x(:,1)...
.* (x(:,6).^2 + x(:,7).^2) + 7.4777 .* (x(:,6).^3...
+ x(:,7).^3) + 0.7854 .* (x(:,4) .* x(:,6).^2 + x(:,5) .* x(:,7).^2);

limit1 = ((27 ./ (x(:,1) .* x(:,2).^2 .* x(:,3))) - 1 > 0);
y(limit1) = +Inf;

limit2 = ((397.5 ./ (x(:,1) .* x(:,2).^2 .* x(:,3).^2)) - 1 > 0);
y(limit2) = +Inf;

limit3 = (((1.93 .* x(:,4).^3) ./ (x(:,2)...
.*x(:,3).*x(6).^4)) - 1 > 0);
y(limit3) = +Inf;

limit4 = (((1.93 .* x(:,5).^3) ./ (x(:,2)...
.* x(:,3) .* x(:,7).^4)) - 1 > 0);
y(limit4) = +Inf;

limit5 = ((1 ./ (110 .* x(:,6).^3) .* sqrt(((745 .* x(:,4))...
./ (x(:,2) .* x(:,3))))).^2 + 16.9 .* 10^6)) - 1 > 0);

```

```

y(limit5) = +Inf;

limit6 = ((1 ./ (85 .* x(:,7).^3).* sqrt(((745 .* x(:,5))...
./ (x(:,2) .* x(:,3))).^2 + 157.5 .* 10^6)) - 1 > 0);
y(limit6) = +Inf;

limit7 = (((x(:,2) .* x(:,3)) ./ 40) - 1 > 0);
y(limit7) = +Inf;

limit8 = (((5 .* x(:,2)) ./ x(:,1)) - 1 > 0);
y(limit8) = +Inf;

limit9 = ((x(:,1) ./ (12 .* x(:,2))) - 1 > 0);
y(limit9) = +Inf;

limit10 = (((1.5 .* x(:,6)) + 1.9) ./ x(:,4)) - 1 > 0);
y(limit10) = +Inf;

limit11 = (((1.1 .* x(:,7) + 1.9) ./ x(:,5)) - 1 > 0);
y(limit11) = +Inf;

end

```

A.6 Модель пружины

```

function y = Prujina(x)
y = (x(:,3) + 2) .* x(:,2) .* x(:,1).^2;

limit1 = (1 - ((x(:,2).^3 .* x(:,3)) ./ (71785 .* x(:,1).^4)) > 0);
y(limit1) = +Inf;

```

```

limit2 = ((4.*x(:,2).^2 - x(:,1).*x(:,2)) ./ (12566 .* ( x(:,2)...
.* x(:,1).^3 - x(:,1).^4)) + (1 ./ (5108 .* x(:,1).^2)) - 1 > 0);
y(limit2) = +Inf;

limit3 = (1 - ( (140.45 .* x(:,1)) ./ (x(:,2).^2 .* x(:,3)) ) > 0);
y(limit3) = +Inf;

limit4 = (((x(:,2) + x(:,1)) ./ 1.5) - 1 > 0);
y(limit4) = +Inf;
end

```