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 004.89:004.92

«До захисту допущено»
В.о завідувач кафедри
_____ Іван ТЕРЕЩЕНКО
«__» _____ 2026 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою
«Математичні методи моделювання, розпізнавання образів та комп'ютерного зору»

зі спеціальності: 113 Прикладна математика
на тему: **«Методи ідентифікації оптимальних параметрів моделей у задачах інженерного проєктування»**

Виконав:
студент IV курсу, групи ФІ-21
Грунда Ярослав Іванович _____

Керівник:
ст. досл., к.т.н., доц.
Хайдуров Владислав Володимирович _____

Рецензент:
доцент кафедри ММЗІ НН ФТІ КПІ імені Ігоря Сікорського, доцент, канд. фіз.-мат. наук
Ганна Олексіївна Южакова _____

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

Київ — 2026

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
Навчально-науковий фізико-технічний інститут
Кафедра математичного моделювання та аналізу даних

Рівень вищої освіти — перший (бакалаврський)
Спеціальність — 113 Прикладна математика,
ОПП «Кафедра математичного моделювання та аналізу даних»

ЗАТВЕРДЖУЮ
В.о завідувач кафедри
_____ Іван ТЕРЕЩЕНКО
«__» _____ 2026 р.

ЗАВДАННЯ
на дипломну роботу

Студент: Грунда Ярослав Іванович

1. Тема роботи: «*Методи ідентифікації оптимальних параметрів моделей у задачах інженерного проєктування*», науковий керівник дисертації: ст. досл., к.т.н., доц. Хайдуров Владислав Володимирович,

затверджені наказом по університету №__ від «__» _____ 2026 р.

2. Термін подання студентом роботи: «__» _____ 2026 р.

3. Об'єкт дослідження: Процес пошуку ефективних проєктних рішень для складних технічних систем на основі математичного моделювання.

4. Предмет дослідження: Математичні моделі інженерних задач та метаевристичні ройові алгоритми оптимізації (алгоритми горобців, гнойового жука, тунікатів, птаха-секретаря), що застосовуються для знаходження оптимальних проєктних рішень.

5. Перелік завдань: провести аналіз математичних моделей складних інженерних процесів, зокрема: оцінки продуктивності інтегрованих енергетичних систем, роботи пластинчасто-ребристих теплообмінників та проєктування вентильованих гальмівних дисків; дослідити сучасні біоінспіровані методи оптимізації, а саме: алгоритм пошуку горобців (SSA), оптимізатор гнойового жука (DBO та його

вдосконалену версію DPDBO), алгоритм рою тунікатів (TSA) та алгоритм птаха-секретаря (SBOA), визначити їхні особливості та механізми пошуку; реалізувати програмне забезпечення для розв'язання чотирьох задач інженерного проєктування з обмеженнями (проєктування зварної балки, посудини високого тиску, редуктора швидкості та оптимізація конструкції автомобіля для захисту від бокового удару) із застосуванням семи алгоритмів (GA, PSO, SSA, DBO, DPDBO, TSA, SBOA); виконати порівняльний аналіз результатів роботи реалізованих алгоритмів на основі статистичних показників, кривих збіжності та непараметричних статистичних тестів (Фрідмана та Вілкоксона).

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: Презентація доповіді.

7. Дата видачі завдання: 10 вересня 2025 р.

Календарний план

№ з/п	Назва етапів виконання бакалаврської дисертації	Термін виконання	Примітка
1	Узгодження теми роботи із науковим керівником	05–10 вересня 2025 р.	Виконано
2	Огляд опублікованих джерел за тематикою дослідження	Жовтень–листопад 2025 р.	Виконано
3	Розробка ходу проведення досліджень	Грудень 2025 р.	Виконано
4	Розроблення програмного забезпечення та проведення досліджень	Січень–квітень 2026 р.	Виконано
5	Аналіз отриманих результатів	Квітень–травень 2026 р.	Виконано
6	Оформлення дипломної роботи	Травень 2026 р.	Виконано
7	Попередній захист бакалаврської роботи	2 червня 2026 р.	Виконано
8	Подача закінченої бакалаврської роботи на кафедру	7 червня 2026 р.	Виконано
9	Захист бакалаврської роботи	22 червня 2026 р.	Виконано

Студент _____ Ярослав ГРУНДА

Керівник _____ Владислав ХАЙДУРОВ

РЕФЕРАТ

Кваліфікаційна робота містить: 129 стор., 18 рисунки, 25 таблиць, 47 джерел.

Роботу присвячено дослідженню та порівняльному аналізу сучасних метаевристичних алгоритмів оптимізації для ідентифікації параметрів математичних моделей у задачах інженерного проєктування.

У роботі розглянуто математичні моделі складних інженерних систем, зокрема інтегрованої енергетичної системи, пластинчасто-ребристого теплообмінника та вентильованого гальмівного диска. Проведено аналіз сучасних біоінспірованих алгоритмів оптимізації, серед яких алгоритм пошуку горобців (SSA), оптимізатор гнойового жука (DBO), його вдосконалена модифікація DPDBO, алгоритм рою тунікатів (TSA) та алгоритм оптимізації птаха-секретаря (SBOA). Для порівняння використано генетичний алгоритм (GA) та оптимізацію роєм частинок (PSO).

У ході дослідження виконано обчислювальні експерименти на чотирьох задачах інженерного проєктування з обмеженнями: проєктування зварної балки, посудини високого тиску, редуктора швидкості та конструкції автомобіля для захисту від бокового удару. Проведено порівняльний аналіз алгоритмів за якістю отриманих розв'язків, збіжністю та статистичними показниками ефективності. Згідно з отриманими даними, алгоритм SBOA характеризується найвищою стабільністю пошуку. Використання модифікації DPDBO забезпечило вищу точність результатів відносно базового алгоритму DBO, натомість класичний PSO виявився ефективним лише для оптимізаційних задач малої розмірності.

МЕТАЕВРИСТИЧНА ОПТИМІЗАЦІЯ, РОЙОВИЙ ІНТЕЛЕКТ, SSA, DBO, DPDBO, TSA, SBOA, GA, PSO, ІНЖЕНЕРНЕ ПРОЄКТУВАННЯ, ОПТИМІЗАЦІЯ З ОБМЕЖЕННЯМИ

ABSTRACT

The thesis contains: 129 pages, 18 figures, 25 tables, 47 sources.

The thesis focuses on the research and comparative analysis of modern metaheuristic optimization algorithms for identifying parameters of mathematical models in engineering design.

The work examines mathematical models of complex engineering systems, including an integrated energy system, a plate-fin heat exchanger, and a ventilated brake disc. An analysis of modern bio-inspired optimization algorithms is conducted, including the Sparrow Search Algorithm (SSA), the Dung Beetle Optimizer (DBO), its improved modification (DPDBO), the Tunicate Swarm Algorithm (TSA), and the Secretary Bird Optimization Algorithm (SBOA). Additionally, the Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) were implemented for comparison.

In the course of the research, computational experiments were performed on four constrained engineering design problems: welded beam design, pressure vessel design, speed reducer design, and car side impact protection. A comparative analysis of the algorithms was conducted based on the quality of solutions obtained, convergence rates, and statistical performance indicators. According to the obtained data, the SBOA algorithm demonstrates the highest search stability. The use of the DPDBO modification provided higher accuracy of results relative to the base DBO algorithm, whereas the classical PSO proved effective only for low-dimensional optimization problems.

METAHEURISTIC OPTIMIZATION, SWARM INTELLIGENCE, SSA, DBO, DPDBO, TSA, SBOA, GA, PSO, ENGINEERING DESIGN, CONSTRAINED OPTIMIZATION.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AGTOA — алгоритм оптимізації штучних загонів горил
AHP — абсорбційний тепловий насос
ARAS — метод багатокритеріального оцінювання
BT — акумуляторна батарея
CCD — центральний композиційний план
CFD — обчислювальна гідродинаміка
CHP — компресійний тепловий насос
COE — питома вартість виробленої енергії
COP — коефіцієнт перетворення тепла
COPRAS — метод багатокритеріального прийняття рішень
DBO — ройовий оптимізатор гнойового жука
DE — диференціальна еволюція
DPDBO — вдосконалений алгоритм DBO з динамічною популяцією
EDAS — метод багатокритеріального оцінювання
EH — електричний нагрівач
FE / FEs — обчислення цільової функції
FEA — аналіз методом скінченних елементів
GA — генетичний алгоритм
GWO — оптимізатор сірого вовка
HBA — алгоритм медоїда
HGAPSO — гібридний алгоритм (GA + PSO)
HPD — тип пластинчасто-ребристих теплообмінників
HSV — сховище водню
IES — інтегрована енергетична система
LEO — оператор локального виходу
MCDM — багатокритеріальне прийняття рішень

MHR — багатошаровий тепловий резервуар

NFL — теорема про відсутність безкоштовних обідів

$NSGA\text{-}II$ — генетичний алгоритм II з недомінованим сортуванням

OBL — навчання на основі протилежних рішень

$PFHE$ — пластинчасто-ребристий теплообмінник

PH — фотокаталітичний реактор

PSO — оптимізація роєм частинок

PT — фототермальний колектор

PV — фотоелектрична панель

$PWLCM$ — кусково-лінійне хаотичне відображення

RBF — радіально-базисна функція

RCA — алгоритм скорочення регіону

SA — імітація відпалу

$SBOA$ — алгоритм оптимізації птаха-секретаря

SBX — імітоване бінарне схрещування

$SOFC$ — твердооксидний паливний елемент

SSA — алгоритм пошуку горобців

SVR — регресія опорних векторів

TAC — сукупні річні витрати

$TOPSIS$ — метод близькості до ідеального рішення

TSA — алгоритм рою тунікатів

D, d — розмірність простору пошуку

R — питома сонячна іррадіація

S — безрозмірний стан заряду

w — змінна проєктування (розмір компонента)

$\mathbf{x}$ — вектор змінних проєктування

ΔP — перепад тиску

Δt — часовий крок

$\varepsilon\text{-}NTU$ — ефективність – кількість одиниць переносу теплоти

φ — частка спектрального діапазону

Ω — допустима область розв'язків

ВСТУП

Актуальність дослідження. Сучасне інженерне проєктування складних технічних систем вимагає врахування великої кількості взаємопов'язаних параметрів та обмежень. Традиційні методи проєктування часто не здатні забезпечити знаходження глобального екстремуму функцій цілі за прийнятний час, особливо коли йдеться про багатовимірні та нелінійні задачі. Актуальність теми зумовлена необхідністю підвищення ефективності функціонування та оптимізації конструктивних параметрів складних технічних систем різної фізичної природи за допомогою сучасних метаевристичних методів ройового інтелекту, які дозволяють ефективно знаходити глобальний оптимум у задачах з жорсткими обмеженнями. Зокрема, їхнє практичне використання дозволяє автоматизувати процес пошуку найкращих інженерних рішень, знизити матеріаломісткість виробів та покращити їх експлуатаційні характеристики.

Метою дослідження є порівняльний аналіз сучасних метаевристичних методів та алгоритмів ройового інтелекту для ідентифікації оптимальних параметрів математичних моделей інженерних об'єктів та визначення найбільш ефективних методів для різних задач інженерного проєктування.

Для досягнення мети необхідно розв'язати **задачу дослідження**, яка полягає у порівняльному аналізі та програмній реалізації методів та алгоритмів ройового інтелекту для конкретних прикладних задач інженерії. Для розв'язання задачі необхідно вирішити такі завдання:

1) провести огляд опублікованих джерел за тематикою дослідження;

2) дослідити сучасні ройові алгоритми оптимізації (алгоритми поведінки горобців, гнойового жука, рою тунікатів та птаха-секретаря), визначити їхні особливості;

3) програмно реалізувати обрані алгоритми та застосувати їх для

пошуку оптимальних параметрів вибраних інженерних задач;

4) виконати порівняльний аналіз збіжності та ефективності використаних алгоритмів на тестових інженерних задачах.

Об'єктом дослідження є процеси пошуку ефективних проєктних рішень для складних технічних систем на основі математичного моделювання.

Предметом дослідження є математичні моделі інженерних задач та метаевристичні ройові алгоритми оптимізації (алгоритми горобців, пеліканів, гнойового жука, птаха-секретеря), що застосовуються для ідентифікації параметрів цих моделей.

При розв'язанні поставлених завдань використовувались такі *методи дослідження*: методи математичного моделювання (для формалізації інженерних задач), теорія оптимізації (для визначення цільових функцій та обмежень), методи ройового інтелекту та еволюційних обчислень (для пошуку глобального екстремуму), методи статистичного аналізу (для порівняння результатів на основі множинних запусків), а також методи об'єктно-орієнтованого програмування (для реалізації алгоритмів).

1 МАТЕМАТИЧНІ І КОМП'ЮТЕРНІ МОДЕЛІ СКЛАДНИХ ІНЖЕНЕРНИХ ПРОЦЕСІВ ПРОЄКТУВАННЯ

У першому розділі досліджено, як сучасні математичні підходи допомагають проєктувати складні технічні системи, де необхідно враховувати безліч взаємопов'язаних факторів та нелінійних процесів. Основну увагу приділено формалізації реальних інженерних задач у вигляді оптимізаційних моделей, де ключовим є правильний вибір цільових функцій, змінних проєктування та системи експлуатаційних обмежень.

Для дослідження обрано три принципово різні системи: інтегровану енергетичну систему зі спектральним розщепленням випромінювання, пластинчасто-ребристий теплообмінник та вентильований гальмівний диск. Такий вибір дозволяє продемонструвати універсальність запропонованих підходів для задач різної фізичної природи. Для кожного об'єкта розроблено математичний опис, який став фундаментом для подальшого пошуку найкращих характеристик.

1.1 Математична модель оцінки продуктивності інтегрованої енергетичної системи на базі сонячного спектрального розщеплення

Інтегровані енергетичні системи (англ. Integrated Energy Systems, IES) – це клас розподілених енергетичних комплексів, що поєднують кілька різнорідних генеруючих і накопичувальних підсистем з метою одночасного задоволення різнотипних навантажень: електричного, теплового та хімічного, зокрема, водневого. Поширення IES у контексті урбанізованих районів мотивоване необхідністю зниження вуглецевого сліду, покращення енергетичної незалежності та ощадливого

використання обмежених міських площ [2, 43].

Задача оптимального проєктування IES полягає у визначенні розмірів, наприклад таких як, потужність і площа кожного компонента системи, що забезпечують мінімальну питому вартість виробленої енергії за дотримання балансових і технічних обмежень. Математично це нелінійна задача умовної оптимізації великої розмірності, ускладнена часовою дискретизацією: оптимальні розміри обладнання та графіка його роботи залежать від добового та сезонного профілю сонячної іррадіації та навантажень споживача [2].

Задача оптимального вибору розмірів компонентів IES розв'язується у літературі різними метаевристичними та еволюційними підходами. У роботі [23] для багатокритеріальної оптимізації IES за вартістю та надійністю застосовано NSGA-II з побудовою фронту Парето; у близькому до розглянутого дослідженні [43], присвяченому IES із технологією спектрального розщеплення, використано гібрид алгоритму медоїда (англ. Honey Badger Algorithm, HBA) та імітації відпалу (англ. Simulated AnnNealing, SA). У цьому підрозділі як основу прийнято однокритеріальну постановку та алгоритм скорочення регіону (Region Contraction Algorithm, RCA) [46], застосований у роботі [2]: метод декомпозує задачу на точно розв'язувану лінійну підзадачу диспетчеризації та стохастичний пошук вектора розмірів, що знижує обчислювальну складність порівняно з прямою метаевристичною оптимізацією повного простору змінних.

Система (рис. 1.1) складається з таких функціональних підсистем.

1) *Підсистема сонячного перетворення* включає: фотоелектричні панелі (PV), фототермальні колектори (PT) та фотокаталітичний реактор (PH).

2) *Підсистема управління навантаженнями* включає: електричний нагрівач (EH), компресійний тепловий насос (CHP), абсорбційний тепловий насос (AHP) та твердооксидний паливний елемент ($SOFC$).

3) *Підсистема зберігання* включає: акумуляторну батарею (BT),

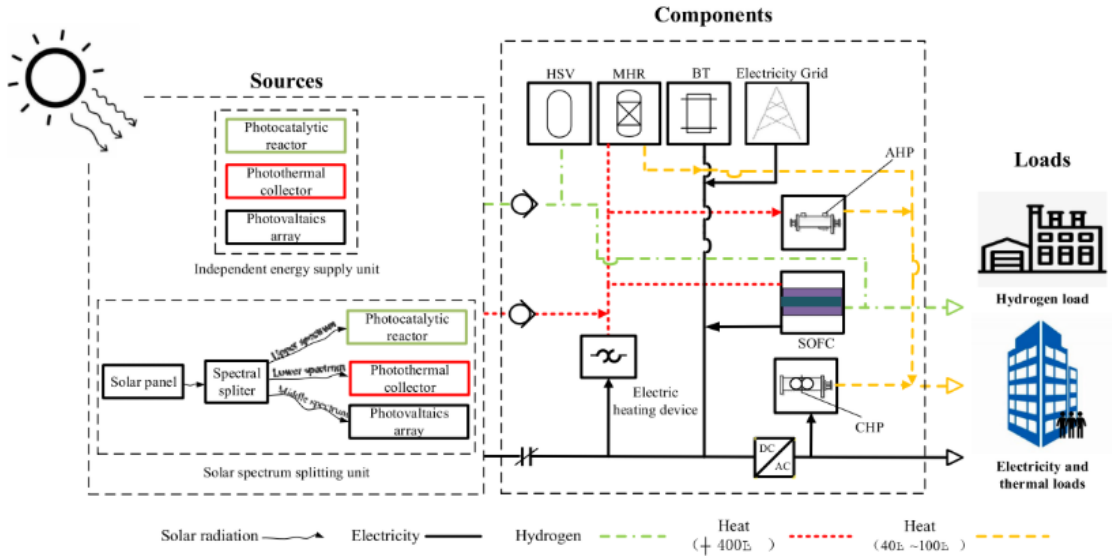

Рисунок 1.1 – Конфігурація інтегрованої енергетичної системи [2]

багатошаровий тепловий резервуар (MHR) та сховище водню (HSV).

1.1.1 Постановка оптимізаційної задачі та змінні проєктування

У роботі [2] розглядаються два альтернативних сценарії конфігурації підсистеми сонячного перетворення. У **Сценарії 1** (блок спектрального розщеплення) оптичний дільник спектра розподіляє вхідне сонячне випромінювання між трьома перетворювачами відповідно до довжини хвилі λ:

– $\lambda < 500$ нм — спрямовується до PH (генерація водню);

– $500 \leq \lambda \leq 800$ нм — до PV (генерація електроенергії);

– $\lambda > 800$ нм — до PT (генерація тепла).

Усі три перетворювача спільно використовують одну панель площею w_{panel}, тому задача зводиться до визначення одного розміру замість трьох окремих. У **Сценарії 2** (блок незалежного енергопостачання) PV, PT та PH незалежно опрацьовують «сире» сонячне випромінювання і мають власні площі w_{PV}, w_{PT}, w_{PH}.

Вектор *змінних проєктування* (змінних розміру), що підлягає оптимізації, об'єднує встановлені потужності та площі всіх компонентів системи:

$$\begin{aligned}\mathbf{w} = \big(&w_{PV},\, w_{PT},\, w_{PH},\, w_{EH},\, w_{CHP},\\ &w_{AHP},\, w_{SOFC},\, w_{BT},\, w_{MHR},\, w_{HSV}\big)^{\top} \in \mathbb{R}^{N_w},\end{aligned} \tag{1.1}$$

де $N_w = 10$ – загальна кількість змінних розміру; у Сценарії 1 перші три параметри замінюються на w_{panel}, що зменшує розмірність задачі до $N_w = 8$.

Для кожного часового кроку $k \in \{1, \ldots, M\}$ додатково визначаються *змінні диспетчеризації* $\mathbf{p}_k$ – вектор генерованих / споживаних потужностей кожного компонента у момент k. Таким чином, задача є дворівневою: зовнішній рівень – вибір $\mathbf{w}$; внутрішній рівень – для фіксованого $\mathbf{w}$ знаходження оптимальної диспетчеризації $\mathbf{p}_k$ для кожного кроку k.

1.1.2 Математична модель перетворення та зберігання енергії

Потужності перетворювачів сонячного випромінювання у Сценарії 2 (незалежні блоки) за часовий крок Δt [год] при питомій іррадіації R_k [Вт/м^2] у момент k визначаються як [2]:

$$\begin{cases} p_{PV,k} = \eta_{PV}\, R_k\, w_{PV}\, \Delta t, \\ q_{PT,k} = \eta_{PT}\, R_k\, w_{PT}\, \Delta t, \\ g_{PH,k} = \dfrac{\eta_{PH}\, R_k\, w_{PH}\, \Delta t}{Q_{H_2}}, \end{cases} \tag{1.2}$$

де $p_{PV,k}$ [кВт·год] – вироблена електроенергія; $q_{PT,k}$ [кВт·год] – вироблена теплова енергія; $g_{PH,k}$ [кг] – вироблений водень; η_{PV}, η_{PT}, $\eta_{PH} \in (0,1)$ – ефективності відповідних перетворювачів; Q_{H_2} [кВт · год/м^3] – нижча

теплота згоряння водню.

У Сценарії 1 три перетворювача спільно опрацьовують поле площею w_{panel}, причому ефективний ККД кожного з них коригується на частку відповідного спектрального діапазону $\varphi_j \in (0,1)$:

$$\begin{cases} p_{PV,k} = \eta^*_{PV}\, R_k\, w_{\text{panel}}\, \Delta t, \\ q_{PT,k} = \eta^*_{PT}\, R_k\, w_{\text{panel}}\, \Delta t, \qquad \eta^*_j = \eta_j \cdot \varphi_j, \quad j \in \{PV,PT,PH\}. \\ g_{PH,k} = \dfrac{\eta^*_{PH}\, R_k\, w_{\text{panel}}\, \Delta t}{Q_{H_2}}, \end{cases} \tag{1.3}$$

Твердооксидний паливний елемент (SOFC) спалює водень із резервуара HSV і виробляє електричну та теплову енергії. Допустима область роботи SOFC обмежена встановленою потужністю [2]:

$$\begin{cases} 0 \;\le\; p_{SOFC,k} \;\le\; \eta_{SOFC}\, w_{SOFC}\, \Delta t, \\ 0 \;\le\; q_{SOFC,k} \;\le\; (1 - \eta_{SOFC} - \eta_l)\, w_{SOFC}\, \Delta t, \end{cases} \tag{1.4}$$

де w_{SOFC} [кВт] – встановлена потужність; η_{SOFC} – електричний ККД; η_l – частка теплоти, що розсіюється в навколишнє середовище. Витрата водню на SOFC пов'язана з виробленою електроенергією:

$$g_{SOFC,k} = \frac{p_{SOFC,k}}{\eta_{SOFC}\, Q_{H_2}}. \tag{1.5}$$

Електричний нагрівач (EH) перетворює електроенергію на теплову з ККД η_{EH}, а теплові насоси CHP та AHP характеризуються коефіцієнтами перетворення тепла COP_{CHP} та COP_{AHP} відповідно. Їхні обмеження на генерацію тепла записуються як [2]:

$$\begin{cases} 0 \;\le\; q_{EH,k} \;\le\; \eta_{EH}\, w_{EH}\, \Delta t, \\ 0 \;\le\; q_{CHP,k} \;\le\; \text{COP}_{CHP}\, w_{CHP}\, \Delta t, \\ 0 \;\le\; q_{AHP,k} \;\le\; \text{COP}_{AHP}\, w_{AHP}\, \Delta t, \end{cases} \tag{1.6}$$

де w_{EH}, w_{CHP}, w_{AHP} [кВт] – встановлені потужності відповідних агрегатів.

Динаміка трьох накопичувачів описується уніфіковано через безрозмірний стан заряду $S_k \in [0,1]$. Для **акумуляторної батареї (BT)** стан заряду S_k^C визначається з урахуванням обмежень на граничну силу струму [2]:

$$\begin{cases} \max\Big(-V_{BT}^{ch}\, I_{MAX}^{ch}\, \Delta t,\ -\dfrac{w_{BT}}{\eta_{BT}^{ch}}\Big) \ \le\ p_{BT,k} \ \le\ 0, \\ 0 \ \le\ p_{BT,k} \ \le\ \min\Big(V_{BT}^{dch}\, I_{MAX}^{dch}\, \Delta t,\, w_{BT}\, \eta_{BT}^{dch}\Big), \\ 0{,}4 \ \le\ S_k^C = S_{k-1}^C + \dfrac{p_{BT,k}}{\eta_{BT}^{dch}\, w_{BT}} \ \le\ 1 \quad \text{(розряд)}, \\ 0{,}4 \ \le\ S_k^C = S_{k-1}^C + \dfrac{\eta_{BT}^{ch}\, p_{BT,k}}{w_{BT}} \ \le\ 1 \quad \text{(заряд)}, \end{cases} \tag{1.7}$$

де V_{BT}^{ch}, V_{BT}^{dch} – напруги заряду та розряду; I_{MAX}^{ch}, I_{MAX}^{dch} – максимальні струми заряду та розряду; η_{BT}^{ch}, η_{BT}^{dch} – відповідні ефективності.

Для **теплового резервуара (MHR)** стан теплового заряду S_k^H описується аналогічно:

$$\begin{cases} 0 \ \le\ S_k^H = S_{k-1}^H + \dfrac{q_{MHR,k}}{\eta_{MHR}^{dch}\, w_{MHR}} \ \le\ 1 \quad \text{(розряд)}, \\ 0 \ \le\ S_k^H = S_{k-1}^H + \dfrac{\eta_{MHR}^{ch}\, q_{MHR,k}}{w_{MHR}} \ \le\ 1 \quad \text{(заряд)}, \end{cases} \tag{1.8}$$

де $q_{MHR,k}$ [кВт·год] – теплова енергія, що надходить або вилучається у момент k; w_{MHR} [кВт·год] – теплова ємність резервуара; η_{MHR}^{ch}, η_{MHR}^{dch} – ефективності теплообміну.

Для **сховища водню (HSV)** стан заряду S_k^G за воднем:

$$\begin{cases} 0 \ \le\ S_k^G = S_{k-1}^G + \dfrac{g_{HSV,k}}{\eta_{HSV}^{dch}\, w_{HSV}} \ \le\ 1 \quad \text{(вилучення)}, \\ 0 \ \le\ S_k^G = S_{k-1}^G + \dfrac{\eta_{HSV}^{ch}\, g_{HSV,k}}{w_{HSV}} \ \le\ 1 \quad \text{(закачування)}, \end{cases} \tag{1.9}$$

де $g_{HSV,k}$ [кг] – маса водню, що вилучається або закачується; w_{HSV} [кг] – місткість сховища; η_{HSV}^{ch}, η_{HSV}^{dch} – ефективності компресора та редуктора.

1.1.3 Цільова функція

Метою оптимізації є мінімізація питомої вартості виробленої енергії (Cost of Energy, COE) [₴/кВт·год] – поширеного техніко-економічного показника для розподілених енергетичних систем. Відповідно до [2], об'єднуючи складові витрат, цільова функція записується як:

$$\min\ COE = \frac{\sum_{k=1}^{M}\sum_{i=1}^{N}\max\left(C_{phy,i}^{dep},\, C_{run,i}^{dep}\right)}{\sum_{k=1}^{M} d_k} + \\ + \frac{g\, C_{coal}\, f_{coal}\, C_{CO_2}^{emi} + C_{aux}^{dep} + C^{mai}}{\sum_{k=1}^{M} d_k}, \tag{1.10}$$

де M – кількість часових кроків у розрахунковому горизонті; N – загальна кількість компонентів; $C_{phy,i}^{dep}$, $C_{run,i}^{dep}$ – складові амортизації i-го компонента (пояснено нижче); g [кВт·год] – обсяг електроенергії, закупленої з мережі; C_{coal} [кг у.п./кВт·год] – коефіцієнт перерахунку електроенергії у вугільний еквівалент; f_{coal} [кг CO_2/кг у.п.] – питомий коефіцієнт викидів CO_2; $C_{CO_2}^{emi}$ [₴/кг CO_2] – ставка екологічного збору; C_{aux}^{dep} [₴] – амортизація допоміжного обладнання (труби, кабелі тощо), прийнята як 30% від загальних капітальних витрат; C^{mai} [₴] – витрати на технічне обслуговування всіх компонентів; d_k [кВт·год] – сумарне корисне навантаження споживача у крок k.

Амортизація кожного компонента розраховується за двома складовими: зносу від часу (фізичного) $C_{phy,i}^{dep}$ та зносу від пропускної здатності $C_{run,i}^{dep}$. Фізична амортизація є рівномірним нарахуванням за один часовий крок, а амортизація від навантаження пропорційна до

фактично виробленої енергії [2]:

$$C^{dep}_{phy,i} = \Delta t \cdot \frac{w_i\, c^{cap}_i}{t_{i,phy}}, \tag{1.11}$$

$$C^{dep}_{run,i} = a_{i,k} \cdot \frac{w_i\, c^{cap}_i}{a_{i,life}}, \tag{1.12}$$

де $w_i\, c^{cap}_i$ [¥] – загальна капітальна вартість i-го компонента (c^{cap}_i – питома вартість, w_i – розмір); $t_{i,phy}$ [год] – паспортний термін служби; $a_{i,k}$ [кВт·год або кг] – обсяг виробленого ресурсу i-м компонентом на кроці k; $a_{i,life}$ [кВт·год або кг] – сумарна пропускна здатність за весь термін служби.

Таким чином, у кожний момент часу нараховується більший із двох видів зносу: $\max(C^{dep}_{phy,i}, C^{dep}_{run,i})$, що реалістично відображає прискорений знос при інтенсивній експлуатації або природне старіння при простої.

1.1.4 Система обмежень та зведена постановка задачі

Для кожного часового кроку k повинна виконуватись умова балансу по кожному типу енергоносія – електрична енергія (P), теплова (Q), водень (G):

$$\sum_{i=1}^{N} a^P_{i,k} = d^P_k, \quad \sum_{i=1}^{N} a^Q_{i,k} = d^Q_k, \quad \sum_{i=1}^{N} a^G_{i,k} = d^G_k, \tag{1.13}$$

де $a^P_{i,k}$, $a^Q_{i,k}$, $a^G_{i,k}$ – кількість електричної, теплової та хімічної (водневої) енергії, що генерується або споживається i-м компонентом на кроці k (від'ємне значення відповідає споживанню); d^P_k, d^Q_k, d^G_k – навантаження споживача.

Кожна змінна проєктування w_i обмежена фізично обґрунтованими нижньою та верхньою межами:

$$w^{min}_i \;\le\; w_i \;\le\; w^{max}_i, \qquad i = 1, \ldots, N_w. \tag{1.14}$$

Як було докладно показано у підрозділі 1.1.2, кожен компонент системи має власні експлуатаційні обмеження. Замість повторного переліку всіх нерівностей, зручно об'єднати обмеження на генерацію та стан накопичувачів (1.2)–(1.9), а також умови балансу (1.13) та межі простору пошуку (1.14) у загальну допустиму область Ω.

На основі введених позначень оптимізаційна задача набуває стандартної форми:

$$\min_{\mathbf{w},\,\{\mathbf{p}_k\}_{k=1}^{M}} \; COE(\mathbf{w},\{\mathbf{p}_k\}) \quad \text{за умов} \quad (\mathbf{w},\{\mathbf{p}_k\}) \in \Omega, \tag{1.15}$$

де $\mathbf{w}$ визначено рівнянням (1.1), а $\{\mathbf{p}_k\}$ – вектори режимів роботи компонентів.

Задача (1.15) є нелінійною задачею умовної оптимізації великої розмірності. Нелінійність зумовлена двома факторами: по-перше, операцією max у функції амортизації (1.11)–(1.12), яка робить цільову функцію кусково-лінійною; по-друге, білінійними членами виду $w_i \cdot a_{i,k}$ у складовій $C^{dep}_{run,i}$, де одночасно фігурують змінна розміру w_i та змінна диспетчеризації $a_{i,k}$.

При горизонті $M = 120$ кроків і десяти змінних проєктування загальна кількість змінних задачі перевищує 10^3, що унеможливлює застосування методів повного перебору та вимагає використання спеціалізованих алгоритмів декомпозиції, таких як алгоритм скорочення регіону (Region Contraction Algorithm, RCA).

1.1.5 Алгоритм скорочення регіону

Для розв'язання задачі (1.15) у роботі [2] застосовано алгоритм скорочення регіону (RCA), запропонований у [46]. Ключова ідея методу полягає у декомпозиції змінних: для кожного фіксованого вектора розмірів $\mathbf{w}$ внутрішня підзадача диспетчеризації є лінійною задачею і розв'язується точно, тоді як зовнішня підзадача вибору $\mathbf{w}$ вирішується

шляхом ітераційного звуження простору пошуку.

Початковий простір розмірів задається як [2]:

$$W_{space,i} = \left[w_i^{min},\, w_i^{max}\right], \tag{1.16}$$

де початково $w_i^{min} = 0$, а верхня межа визначається максимальним навантаженням та характеристиками обладнання:

$$w_i^{max} = \begin{cases} \dfrac{d^{max}}{\Delta t}, & \text{для EH, CHP, AHP, SOFC,} \\ \dfrac{d^{max}}{R_k\,\eta\,\Delta t}, & \text{для PV, PT, PH,} \\ \dfrac{d^{max}}{\eta^{dch}\,\Delta t\,(1 - S^{min})}, & \text{для BT, MHR, HSV,} \end{cases} \tag{1.17}$$

де d^{max} [кВт·год] – максимальне навантаження за часовий крок.

Алгоритм виконує такі кроки [2]:

1) після ініціалізації простору розмірів випадково вибрати групу комбінацій $\mathbf{w}$ у межах поточного простору;

2) з урахуванням обмежень розв'язати внутрішню задачу диспетчеризації та знайти оптимальний розклад $\mathbf{p}$ для отриманої комбінації $\mathbf{w}$;

3) обчислити $COE(\mathbf{w})$, додати комбінацію до пулу кандидатів C_{pool} та відсортувати елементи пулу за значенням COE;

4) коли кількість доданих кандидатів досягає порогу N_u, звузити простір пошуку на основі найкращої підмножини $\mathcal{C}_{best}$:

$$w_i^{min} = \min_{\mathbf{w}\in\mathcal{C}_{best}} w_i, \qquad w_i^{max} = \max_{\mathbf{w}\in\mathcal{C}_{best}} w_i. \tag{1.18}$$

Якщо досягнуто максимальну кількість обчислень – повернути $\mathbf{w}^*$ з мінімальним COE; інакше перейти до кроку 1.

Для рівномірного покриття простору пошуку використовується метод динамічного дискретизованого простору [2], що контролює розподіл вузлів

та генерацію кандидатів:

$$w_i(k) = w_i(k-1) + r_g \left[w_i(k-1) - w_i(k-2) \right], \tag{1.19}$$

де $w_i(1) = w_i^{min}$, $w_i(2) = w_i^{min} + \dfrac{w_i^{max} - w_i^{min}}{S}$, $r_g > 0$ – фактор росту інтервалу:

$$S = \begin{cases} \dfrac{1 - r_g^{N_{node}}}{1 - r_g}, & r_g \neq 1, \\ N_{node}, & r_g = 1, \end{cases}$$

де N_{node} – кількість вузлів дискретизації.

Алгоритм RCA може мати обмеження при наявності різких стрибків цільової функції або точок з близькими екстремумами. Однак фізичні умови задачі оптимізації IES таких особливостей не мають, що забезпечує надійну збіжність методу [2].

1.1.6 Аналіз результатів моделювання

Чисельне дослідження [2] проводилось для будівлі загальною площею 3692 м^2 у м. Сіань (Китай) з кроком $\Delta t = 1$ год на горизонті $M = 5 \times 24 = 120$ кроків, що відповідає п'яти репрезентативним робочим добам зимового сезону (сонячні, похмурі, частково хмарні та дощові дні) із варіацією навантаження 20%.

Результати оптимального підбору розмірів компонентів наведено у таблиці 1.1. Алгоритм RCA виключив ЕН та АНР з оптимальної конфігурації через їхню низьку економічну ефективність – їхні розміри прямують до нуля.

Оптимізована IES повністю задовольняє погодинний електричний, тепловий та водневий попит у обох сценаріях. Порівняльний аналіз ключових техніко-економічних показників наведено у таблиці 1.2.

СОЕ обох сценаріїв нижче за довідкову ціну: Сценарій 1 забезпечує економію 32,8%, Сценарій 2 – 51,1%. Водночас застосування технології

Таблиця 1.1 – Оптимальні розміри компонентів IES [2]

Компонент	**Одиниця**	**Сценарій 1**	**Сценарій 2**
Solar panel	м^2	1220,5	0
PT	м^2	0	632,2
PH	м^2	0	1215,3
PV	м^2	0	722,0
EH	кВт	0	0,3
CHP	кВт	56,8	48,0
AHP	кВт	0	0
SOFC	кВт	51,8	62,6
MHR	кВт·год	763,1	823,3
HSV	кг	66,7	65,5
BT	кВт·год	93,5	93,5

Таблиця 1.2 – Порівняння COE оптимізованої IES та традиційного енергопостачання [2]

Показник	**Традиційне**	**Сценарій 1 (SBS)**	**Сценарій 2**
COE [¥/кВт·год]	0,3948	0,2655	0,1930
Загальна ефективність [%]	—	45,32	23,58
Площа сонячного збору [м^2]	—	1220,5	2569,5

розщеплення підвищує загальну енергоефективність на 21,74 в.п. та скорочує площу для сонячного збору на 52,5%, проте збільшує COE на 27,3% через значну капітальну вартість оптичного дільника спектра (47,13% від сумарних витрат). Таким чином, вибір між сценаріями визначається пріоритетами проєкту: Сценарій 1 доцільний за жорстких обмежень на земельні ресурси, Сценарій 2 – за умов мінімізації операційних витрат.

1.2 Оптимізаційна математична модель роботи пластинчасто-ребристих теплообмінників із зубчастими зміщеними ребрами

У цьому підрозділі розглядається задача багатокритеріальної оптимізації пластинчасто-ребристого теплообмінника (англ. Plate-Fin Heat Exchanger , PFHE) із зубчастими зміщеними ребрами, призначеного

для системи охолодження двигуна гібридного транспортного засобу з розширеним діапазоном ходу. Математична модель та оптимізація базуються на дослідженні [25]. Основну увагу приділено формулюванню задачі оптимізації: визначенню змінних проєктування, цільових функцій, обмежень та алгоритмів пошуку оптимальних параметрів.

1.2.1 Огляд методів оптимізації пластинчасто-ребристих теплообмінників

Пластинчасто-ребристі теплообмінники (PFHE) є одним з найбільш поширених об'єктів оптимізаційного проєктування у теплотехніці завдяки своїй компактності, високій теплопередачі та відносно низькій вартості виробництва. Задача оптимального проєктування PFHE формулюється як багатокритеріальна задача нелінійного програмування з обмеженнями, де цільові функції (теплопередача, перепад тиску, вартість) є конфліктуючими між собою [33]. Огляд літератури засвідчує різноманіття застосовуваних оптимізаційних підходів.

Класичні аналітичні методи. Традиційний підхід до проєктування PFHE базується на методах логарифмічної середньої різниці температур та ефективності і кількості одиниць переносу теплоти (ε-NTU). Ці методи, формалізовані у фундаментальній монографії Шаха та Секулича [34], дозволяють отримати аналітичний зв'язок між конструктивними параметрами та термогідравлічними характеристиками апарата. Проте при багатокритеріальній постановці задачі з великою кількістю змінних традиційний підхід до оптимізації на основі цих методів стає неефективним, оскільки аналітичні залежності не здатні самостійно обробляти складні нелінійні та суперечливі обмеження (наприклад, одночасну мінімізацію втрат тиску та максимізацію теплообміну). Вирішенням цієї проблеми став стрімкий розвиток методів обчислювальної гідродинаміки (CFD), які дозволяють оптимізувати

конструкцію теплообмінників з високою ефективністю на основі детального тривимірного моделювання [25].

Однокритеріальні метаевристичні підходи. Рао та Патель [32] вперше застосували алгоритм рою частинок (PSO) до термодинамічної оптимізації перехресно-точних PFHE. Як незалежні цільові функції розглядалися мінімізація числа одиниць генерації ентропії (із забезпеченням заданого теплового навантаження), мінімізація загального об'єму та сукупних річних витрат (TAC). Проведене порівняння засвідчило високу обчислювальну ефективність методу: PSO досягав глобального оптимуму за значно меншу кількість генерацій порівняно з традиційним генетичним алгоритмом (GA), демонструючи високий потенціал ройового інтелекту в задачах проєктування теплообмінників.

Багатокритеріальні генетичні алгоритми. Санає та Хадж-Абдоллагі [33] використали алгоритм NSGA-II для одночасної максимізації ефективності та мінімізації TAC теплообмінника. Отримавши фронт Парето, автори пішли далі та застосували штучну нейронну мережу для створення зручних кореляцій, що дозволяють проєктувальнику швидко знаходити оптимальні параметри як компромісне рішення. Наджафі та ін. [30] розв'язали задачу оптимізації PFHE типу «повітря-повітря» за двома конфліктуючими цілями — максимізацією загального теплового потоку та мінімізацією TAC. Застосуванню багатокритеріального GA у їхній роботі передував детальний аналіз чутливості, який наочно продемонстрував вплив геометрії ребер на конфлікт між тепловою ефективністю та економічними витратами.

Гібридні та новітні алгоритми. Пачпуте та Мор [31] запропонували гібридний алгоритм HGAPSO, що поєднує PSO із GA і демонструє вищу точність за менший обчислювальний час порівняно з окремими PSO та GA. Гюрсес та ін. [15] застосували алгоритм штучних загонів горил (Artificial Gorilla Troops Optimization Algorithm, AGTOA) для мінімізації повної вартості PFHE (початкової та експлуатаційної),

отримавши конкурентоспроможні результати порівняно з іншими алгоритамим.

Підхід, прийнятий у даній роботі. У дослідженні Лі та ін. [25], яке є основою цього підрозділу, застосовано комплексну методику, що поєднує побудову сурогатних моделей на основі регресії опорних векторів (SVR), багатокритеріальну оптимізацію алгоритмом NSGA-II та вибір остаточного рішення методом TOPSIS. Ключовою відмінністю від попередніх робіт є те, що цільові функції не задані аналітично, а побудовані як апроксимації результатів чисельних CFD-симуляцій. Це дозволяє врахувати складні фізичні ефекти (зокрема, турбулентний теплообмін, вплив геометрії ребер на гідродинаміку потоку), зберігаючи при цьому обчислювальну ефективність: сурогатна модель обчислюється на кілька порядків швидше за повну CFD-симуляцію, що робить можливим застосування популяційних алгоритмів оптимізації з тисячами обчислень цільової функції.

1.2.2 Об'єкт дослідження та змінні проєктування

Об'єктом дослідження є теплообмінник із зубчастими зміщеними ребрами типу HPD (High Pressure Direction), що складається з 14 шарів ребер: непарні шари призначені для водяного контуру, парні — для оливного [25]. Конструктивні параметри ребер (зсув, крок, геометрія) суттєво впливають на інтенсивність теплообміну та гідравлічний опір: зміщення ребер руйнує пограничний шар та генерує хвостові вихори, що посилює теплопередачу, проте водночас збільшує втрати тиску [25]. Схематичне зображення моделі наведено на рис. 2.1. Основними *змінними проєктування* обрано три параметри, що визначають термогідравлічні характеристики апарата [25]:

$$\mathbf{x} = (\alpha,\, q_{oil},\, q_{water})^{\top} \in \mathbb{R}^3, \tag{1.20}$$

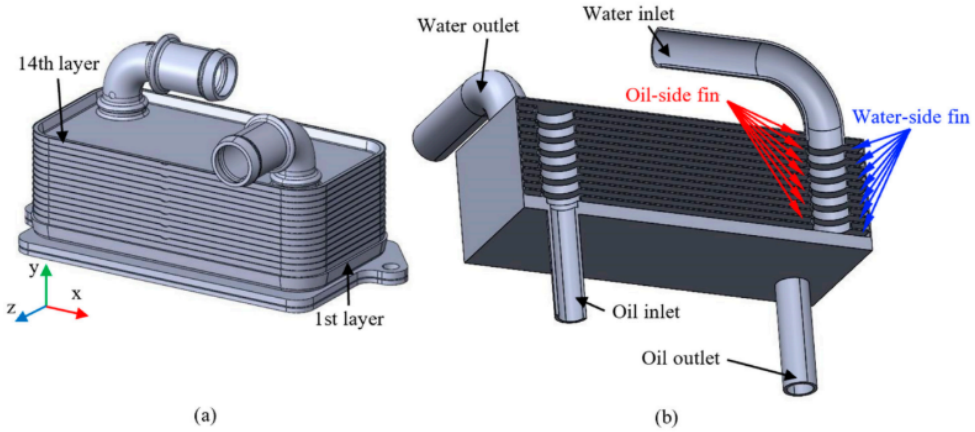

Рисунок 1.2 – Геометрична модель теплообмінника: (a) повна модель, (b) спрощена розрахункова область [25]

де α [°] – кут нахилу зубчастих ребер, що визначає геометрію решітки та інтенсивність турбулізації потоку; q_{oil} [л/хв] – об'ємна витрата мастильної оливи; q_{water} [л/хв] – об'ємна витрата охолоджувальної води.

Кожна змінна обмежена фізично обґрунтованими нижньою та верхньою межами, що утворюють *граничні-обмеження*:

$$x_j^L \le x_j \le x_j^U, \qquad j = 1, 2, 3, \tag{1.21}$$

де x_j^L, x_j^U – нижня та верхня межі j-ї змінної відповідно. Конкретні значення меж наведено у таблиці 1.3.

Таблиця 1.3 – Діапазони зміни змінних проєктування [25]

Змінна проєктування	**Позначення**	x_j^L	x_j^U
Кут нахилу ребер	α	30°	90°
Витрата оливи	q_{oil}	5 л/хв	15 л/хв
Витрата води	q_{water}	5 л/хв	15 л/хв

Таким чином, простір пошуку $\mathcal{X}$ є тривимірним прямокутним паралелепіпедом:

$$\mathcal{X} = \left\{ \mathbf{x} \in \mathbb{R}^3 \;\middle|\; x_j^L \le x_j \le x_j^U,\, j = 1{,}2{,}3 \right\} = [30°, 90°] \times [5, 15] \times [5, 15]. \tag{1.22}$$

Граничні умови CFD-моделі, що використовується для генерації

навчальних даних, зафіксовано: температура оливи на вході – $T_{oil,in} = 100\,^{\circ}\mathrm{C}$, температура води – $T_{water,in} = 65\,^{\circ}\mathrm{C}$, зовнішні стінки адіабатичні [25]. Ці параметри не є змінними оптимізації і залишаються сталими для всіх чисельних експериментів.

1.2.3 Формулювання задачі багатокритеріальної оптимізації

Задача проєктування теплообмінника є типовим прикладом *багатокритеріальної оптимізації*: одночасне покращення теплопередачі, зниження перепаду тиску та забезпечення належного температурного режиму є конфліктуючими цілями, оскільки інтенсифікація теплообміну неминуче призводить до зростання гідравлічного опору [25, 33]. Формально задача записується як *задача багатокритеріальної оптимізації з граничними-обмеженнями* [25]:

$$\boxed{\begin{aligned} &\min_{\mathbf{x}\in\mathcal{X}} \ \mathbf{F}(\mathbf{x}) = \big(-Q(\mathbf{x}),\, \Delta P_{oil}(\mathbf{x}),\, T_{out}(\mathbf{x})\big)^{\top}, \\ &\text{де} \quad \mathcal{X} = \big\{\mathbf{x} \in \mathbb{R}^3 \;\big|\; x_j^L \le x_j \le x_j^U,\, j = 1{,}2{,}3\big\}, \end{aligned}} \tag{1.23}$$

де $\mathbf{F} : \mathcal{X} \to \mathbb{R}^3$ – векторна цільова функція, складена з трьох скалярних компонент, описаних нижче.

Цільова функція 1: тепловий потік $Q(\mathbf{x})$ характеризує кількість теплової енергії, переданої від оливи до охолоджувальної води за одиницю часу [кВт]. Оскільки задача зводиться до мінімізації, максимізація Q еквівалентна мінімізації $-Q(\mathbf{x})$ [25]. Більший тепловий потік означає ефективніше охолодження двигуна, тому $Q(\mathbf{x}) \to \max$.

Цільова функція 2: перепад тиску оливи $\Delta P_{oil}(\mathbf{x})$ [кПа] – різниця статичного тиску між входом та виходом оливного контуру. Мінімізація перепаду тиску є важливою з технічних та економічних міркувань: менший перепад тиску дозволяє використовувати менш потужні (і дешевші) нагнітальні насоси та знижує витрати електроенергії на прокачування рідини [25]. Отже, $\Delta P_{oil}(\mathbf{x}) \to \min$.

Цільова функція 3: температура оливи на виході $T_{out}(\mathbf{x})$ [°C] – температура потоку оливи, що залишає теплообмінник. Чим ближче ця температура до температури охолоджувальної води на вході ($T_{water,in} = 65\,°\text{C}$), тим ефективніше відбувається теплообмін. Отже, $T_{out}(\mathbf{x}) \to \min$.

Важливо зазначити, що конфліктність цільових функцій підтверджується результатами чисельних експериментів [25]: однаковій температурі виходу оливи можуть відповідати суттєво різні значення теплового потоку, а зменшення кута ребер з 90° до 60° знижує перепад тиску при майже сталій теплопередачі, тоді як подальше зменшення до 30° призводить до *зростання* перепаду тиску через надмірну турбулізацію. Така немонотонна поведінка цільових функцій унеможливлює знаходження оптимуму інтуїтивними або однофакторними методами.

Оскільки задача (1.23) має лише граничні-обмеження (відсутні нелінійні функціональні обмеження або обмеження на вихідні характеристики), будь-яка точка $\mathbf{x} \in \mathcal{X}$ є *допустимою*. Це суттєво спрощує реалізацію оптимізатора: не потрібно застосовувати спеціальні методи обробки обмежень (штрафні функції, бар'єрні методи тощо), достатньо обмежити простір пошуку прямокутними межами.

1.2.4 Побудова сурогатних моделей цільових функцій методом SVR

Кожна з трьох цільових функцій у задачі (1.23) не задана аналітично, а визначається через результати чисельного CFD-моделювання теплообмінника. Пряме обчислення цільових функцій на кожній ітерації популяційного оптимізатора є неприйнятним через високу обчислювальну вартість: одна CFD-симуляція потребує від десятків хвилин до кількох годин [25]. Тому між чисельною моделлю та

оптимізатором вводиться проміжний рівень – *сурогатні моделі*, що апроксимують залежності $Q(\mathbf{x})$, $\Delta P_{oil}(\mathbf{x})$, $T_{out}(\mathbf{x})$ як аналітичні функції від вектора $\mathbf{x}$.

Для побудови сурогатних моделей використано метод *регресії опорних векторів* (англ. Support Vector Regression, SVR). Нехай задано навчальну вибірку $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$, де $\mathbf{x}_i \in \mathcal{X}$ – вектор вхідних параметрів i-го чисельного експерименту, $y_i \in \mathbb{R}$ – відповідне значення цільової функції, отримане з CFD-симуляції, m – кількість навчальних зразків ($m = 45$ у даній задачі). SVR будує апроксимуючу функцію у вигляді [25]:

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{m} (\alpha_i - \alpha_i^*)\, K(\mathbf{x}_i, \mathbf{x}) + b, \tag{1.24}$$

де $\alpha_i, \alpha_i^* \geq 0$ – множники Лагранжа, що є розв'язком дуальної задачі квадратичного програмування SVR; $b \in \mathbb{R}$ – зміщення; $K(\mathbf{x}_i, \mathbf{x})$ – ядерна функція, що здійснює неявне відображення вхідних даних у простір ознак вищої розмірності, де побудова лінійної регресії стає можливою.

Ідея SVR полягає у побудові так званої «ε-зони нечутливості» навколо апроксимуючої гіперповерхні: похибки апроксимації, що не перевищують заданого порогу $\varepsilon > 0$, не штрафуються, а лише зразки, що лежать за межами цієї зони, вносять внесок у функцію втрат [36]. Формально, задача SVR зводиться до мінімізації регуляризованого функціоналу:

$$\min_{\mathbf{w}, b, \xi, \xi^*} \ \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{m} (\xi_i + \xi_i^*), \tag{1.25}$$

за умов:

$$\begin{cases} y_i - \langle \mathbf{w}, \varphi(\mathbf{x}_i) \rangle - b \ \leq \ \varepsilon + \xi_i, \\ \langle \mathbf{w}, \varphi(\mathbf{x}_i) \rangle + b - y_i \ \leq \ \varepsilon + \xi_i^*, \\ \xi_i, \xi_i^* \geq 0, \quad i = 1, \ldots, m, \end{cases} \tag{1.26}$$

де $\mathbf{w}$ – вектор вагових коефіцієнтів у просторі ознак; $\varphi(\mathbf{x})$ – нелінійне відображення вхідних даних у простір ознак, що визначається ядерною

функцією; $C > 0$ – параметр регуляризації, що контролює баланс між складністю моделі та точністю апроксимації; $\xi_i, \xi_i^* \geq 0$ – змінні відхилення (slack variables), що вимірюють порушення ε-зони.

У якості ядерної функції обрано Гаусову радіально-базисну функцію (RBF) [25]:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right) = \exp\left(-\gamma_{RBF}\,\|\mathbf{x}_i - \mathbf{x}_j\|^2\right), \tag{1.27}$$

де $\sigma > 0$ – ширина Гаусової функції (визначає радіус впливу кожного опорного вектора); $\gamma_{RBF} = 1/(2\sigma^2)$ – параметр ядра. RBF-ядро обрано завдяки його універсальній апроксимаційній здатності та залежності лише від одного гіперпараметра σ, що спрощує процедуру налаштування моделі [36].

Оскільки задача (1.23) містить три цільові функції, процедура SVR-апроксимації виконується *незалежно для кожної з них*. Для кожної цілі формується власна навчальна вибірка $\mathcal{D}^{(s)} = \{(\mathbf{x}_i, y_i^{(s)})\}_{i=1}^{45}$, де верхній індекс $s \in \{Q, P, T\}$ позначає відповідну цільову функцію, а $y_i^{(s)}$ – значення цієї функції, обчислене CFD-симуляцією для i-го набору параметрів. Розв'язання трьох незалежних задач виду (1.25)–(1.26) дає три сурогатні моделі [25]:

$$\begin{aligned}
\hat{Q}(\mathbf{x}) &= \sum_{i=1}^{45}\left(\alpha_i^{(Q)} - \alpha_i^{*(Q)}\right) K(\mathbf{x}_i, \mathbf{x}) + b^{(Q)}, \\
\widehat{\Delta P}_{oil}(\mathbf{x}) &= \sum_{i=1}^{45}\left(\alpha_i^{(P)} - \alpha_i^{*(P)}\right) K(\mathbf{x}_i, \mathbf{x}) + b^{(P)}, \\
\hat{T}_{out}(\mathbf{x}) &= \sum_{i=1}^{45}\left(\alpha_i^{(T)} - \alpha_i^{*(T)}\right) K(\mathbf{x}_i, \mathbf{x}) + b^{(T)},
\end{aligned} \tag{1.28}$$

де $\alpha_i^{(s)}$, $\alpha_i^{*(s)}$ – множники Лагранжа, отримані при навчанні SVR-моделі для s-ї цільової функції; $b^{(s)}$ – відповідне зміщення; $K(\cdot,\cdot)$ – спільне для всіх моделей RBF-ядро (1.27); $\mathbf{x}_i$ – вектори параметрів з навчальної вибірки (спільні для всіх трьох моделей, оскільки CFD-симуляції

проводились для одних і тих самих 45 комбінацій параметрів). Кожна з функцій (1.28) є неперервною та диференційовною відносно $\mathbf{x}$, а обчислення її значення для довільної точки зводиться до підсумовування 45 зважених ядерних функцій, що виконується за мікросекунди.

Саме ці три функції (1.28) є тим, що безпосередньо оптимізує алгоритм NSGA-II: на кожній ітерації оптимізатор подає вектор параметрів $\mathbf{x}$, обчислює трійку значень $(\hat{Q}, \widehat{\Delta P}_{oil}, \hat{T}_{out})$ і на основі домінування за Парето оцінює якість кожного рішення-кандидата.

Навчальна вибірка $\mathcal{D}$ отримана з 45 чисельних CFD-експериментів (9 комбінацій витрат оливи та води $\times$ 5 значень кута ребер), проведених у програмному пакеті ANSYS Fluent [25]. Побудовані SVR-моделі забезпечили максимальну відносну похибку апроксимації менше 2,5% відносно результатів CFD-симуляцій (див. рис. 1.3), що підтверджує їхню придатність як замінників повної чисельної моделі для подальшої оптимізації [25]. Таким чином, у задачі (1.23) цільові функції

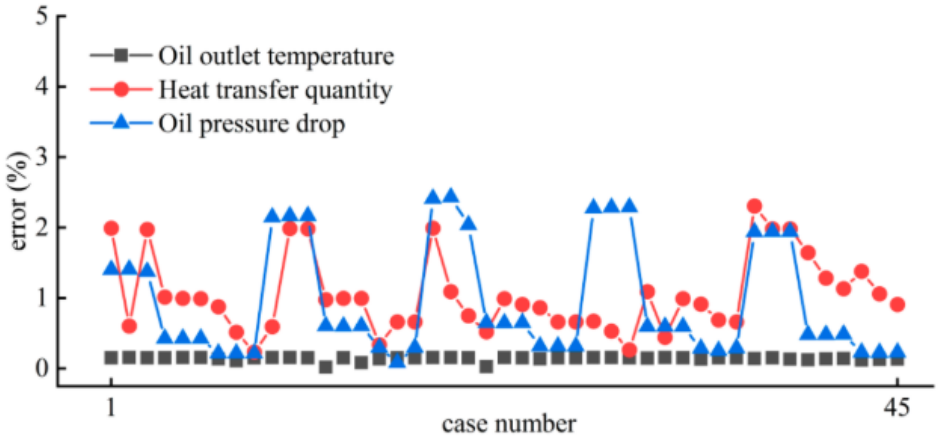

Рисунок 1.3 – Відносна похибка SVR-моделі для трьох цільових функцій [25]

замінюються їх сурогатними апроксимаціями (1.28), і задача оптимізації, що безпосередньо розв'язується алгоритмом NSGA-II, набуває вигляду:

$$\min_{\mathbf{x}\in\mathcal{X}} \hat{\mathbf{F}}(\mathbf{x}) = \left(-\hat{Q}(\mathbf{x}),\, \widehat{\Delta P}_{oil}(\mathbf{x}),\, \hat{T}_{out}(\mathbf{x})\right)^{\top}. \tag{1.29}$$

1.2.5 Багатокритеріальна оптимізація алгоритмом NSGA-II

Для пошуку множини Парето-оптимальних рішень задачі (1.29) застосовано алгоритм NSGA-II (Non-dominated Sorting Genetic Algorithm II) – елітарний генетичний алгоритм, запропонований Дебом та ін. [6].

Ключовим поняттям у багатокритеріальній оптимізації є *домінування за Парето*. Рішення $\mathbf{x}^{(a)}$ *домінує* рішення $\mathbf{x}^{(b)}$ (позначається $\mathbf{x}^{(a)} \prec \mathbf{x}^{(b)}$), якщо воно не гірше за $\mathbf{x}^{(b)}$ за всіма цілями і строго краще хоча б за однією [6]:

$$\mathbf{x}^{(a)} \prec \mathbf{x}^{(b)} \iff \begin{cases} F_k(\mathbf{x}^{(a)}) \le F_k(\mathbf{x}^{(b)}) & \forall\, k \in \{1,2,3\}, \\ \exists\, k_0 \colon\ F_{k_0}(\mathbf{x}^{(a)}) < F_{k_0}(\mathbf{x}^{(b)}). \end{cases} \tag{1.30}$$

Множина усіх недомінованих рішень утворює *фронт Парето* $\mathcal{P}^*$ – множину компромісних варіантів, жоден з яких не може бути покращений за одним критерієм без погіршення за іншим.

NSGA-II оперує популяцією рішень-кандидатів, до яких на кожному поколінні застосовуються оператори схрещування та мутації. Після злиття батьківської та дочірньої популяцій виконується недомінований сортування: рішення розбиваються на фронти $\mathcal{F}_1, \mathcal{F}_2, \ldots$ за рівнем домінування, після чого формується нове покоління шляхом послідовного включення фронтів з пріоритетом за *дистанцією скупчення* (англ. crowding distance) – метрикою, що забезпечує рівномірний розподіл рішень уздовж фронту Парето [6].

Налаштування алгоритму для даної задачі [25]: розмір популяції – $N_{pop} = 2000$ індивідів, максимальна кількість поколінь – $G_{max} = 5000$, ймовірність схрещування – $p_c = 0{,}9$, ймовірність мутації – $p_m = 0{,}2$.

Результат роботи NSGA-II для даної задачі наведено на рис. 1.4: 2000 Парето-оптимальних рішень утворюють тривимірну поверхню у просторі цільових функцій $(\hat{Q}, \widehat{\Delta P}_{oil}, \hat{T}_{out})$. Жодне з 2000 рішень не є

однозначно найкращим за всіма трьома критеріями, що обґрунтовує необхідність застосування методу прийняття рішень для вибору остаточного варіанту.

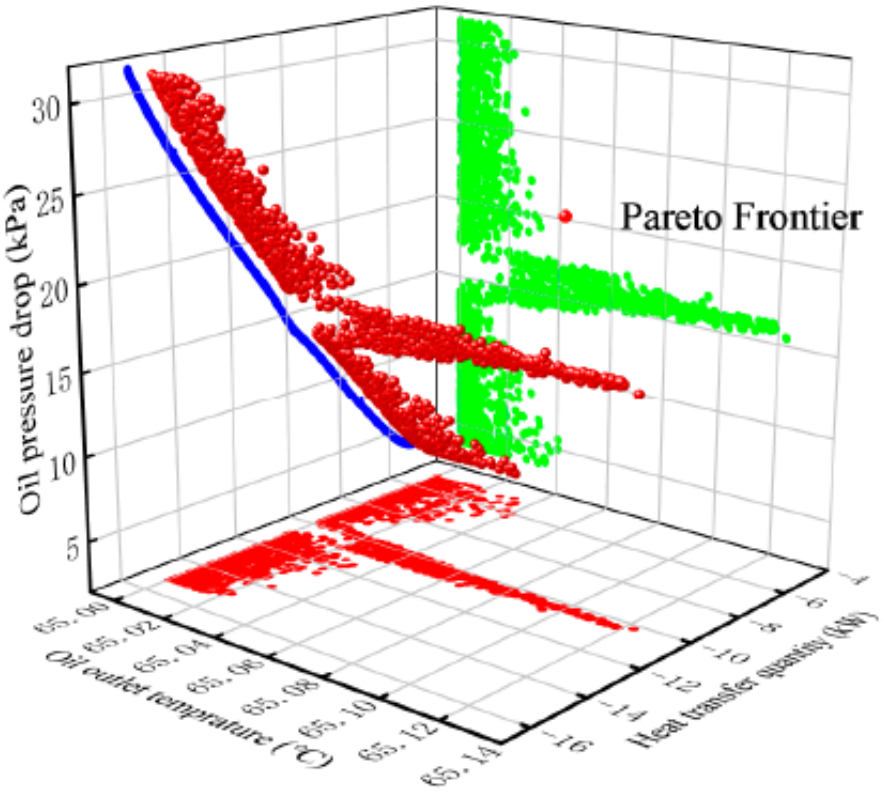

Рисунок 1.4 – Розподіл Парето-оптимальних рішень у просторі трьох цільових функцій [25]

1.2.6 Прийняття рішення методом TOPSIS

Результатом роботи NSGA-II є множина $\mathcal{P}^*$ із 2000 Парето-оптимальних рішень, кожне з яких представляє певний компроміс між трьома цілями. Для вибору єдиного оптимального варіанту застосовано метод *TOPSIS* (Technique for Order Preference by Similarity to Ideal Solution) – метод багатокритеріального прийняття рішень (англ. Multi-Criteria Decision Making, MCDM), що ранжує альтернативи за критерієм відносної близькості до ідеального рішення [25].

Алгоритм TOPSIS складається з таких кроків. Спочатку формується нормалізована зважена матриця рішень. Потім визначаються *позитивне ідеальне рішення* X^+ (найкращі значення за кожним критерієм серед усіх Парето-оптимальних рішень) та *негативне ідеальне рішення* X^- (найгірші значення). Відстані i-ї альтернативи до цих ідеалів визначаються у зваженому евклідовому просторі [25]:

$$D_i^+ = \sqrt{\sum_{j=1}^{n}(X_{ij} - X_j^+)^2}, \qquad D_i^- = \sqrt{\sum_{j=1}^{n}(X_{ij} - X_j^-)^2}, \tag{1.31}$$

де X_{ij} – значення j-ї цільової функції для i-го рішення у зваженій нормалізованій матриці; n – кількість цільових функцій (у даній задачі $n = 3$); X_j^+, X_j^- – компоненти позитивного та негативного ідеалів відповідно.

Критерієм вибору є *відносна близькість* C_i^*, яка має бути максимальною серед усіх альтернатив [25]:

$$C_i^* = \frac{D_i^-}{D_i^- + D_i^+}, \qquad i = 1, \ldots, |\mathcal{P}^*|. \tag{1.32}$$

Значення $C_i^* \in [0, 1]$: чим ближче C_i^* до одиниці, тим ближче i-те рішення до ідеального та далі від найгіршого варіанту. Рішення з максимальним C_i^* обирається як остаточне.

Ваги цільових функцій у даному дослідженні прийнято однаковими [25]:

$$w_Q = w_{\Delta P} = w_{T_{out}} = \frac{1}{3}, \tag{1.33}$$

що відповідає рівнозначності критеріїв теплопередачі, гідравлічних втрат та температурного режиму. У загальному випадку ваги можуть визначатись на основі пріоритетів проєктувальника.

1.2.7 Результати оптимізації

Порівняння знайденого TOPSIS-оптимального рішення з результатами верифікаційної CFD-симуляції наведено в таблиці 1.4.

Таблиця 1.4 – Порівняння результатів оптимізації та перевірочного CFD-розрахунку [25]

Параметр / Функція	**Оптимізація (SVR)**	**CFD-симуляція**	**Похибка (%)**
Кут нахилу ребер (α)	63,01°	63,01°	–
Витрата оливи (q_{oil})	9,7 л/хв	9,7 л/хв	–
Витрата води (q_{water})	6,45 л/хв	6,45 л/хв	–
Теплопередача (Q)	**9,76 кВт**	**9,79 кВт**	**0,31**
Перепад тиску (ΔP_{oil})	**14,00 кПа**	**13,63 кПа**	**2,64**
Т-ра виходу оливи (T_{out})	**65,00°C**	**65,11°C**	**0,17**

Максимальна похибка між прогнозом SVR-моделі та результатом прямого CFD-розрахунку для оптимальних параметрів складає лише 2,64% (для перепаду тиску), що підтверджує високу точність побудованих сурогатних моделей [25].

Знайдене оптимальне рішення $\mathbf{x}^* = (63{,}01°;\ 9{,}7;\ 6{,}45)$ має кілька важливих особливостей з точки зору оптимізації.

По-перше, оптимальний кут нахилу ребер $\alpha^* = 63{,}01°$ є *внутрішньою точкою* простору пошуку ($30° < 63{,}01° < 90°$), а не граничним значенням. Це підтверджує нелінійний характер залежності цільових функцій від α та неможливість знаходження оптимуму інтуїтивним шляхом або однофакторним перебором [25].

По-друге, оптимальні витрати оливи та води суттєво відрізняються ($q^*_{oil} = 9{,}7 > q^*_{water} = 6{,}45$ л/хв), що свідчить про *асиметрію* оптимального режиму відносно двох контурів теплообмінника. Цей результат не є очевидним з фізичних міркувань і може бути отриманий лише шляхом систематичної оптимізації. По-третє, температура оливи на виході $T^*_{out} = 65{,}00\,°\mathrm{C}$ практично збігається з температурою охолоджувальної води на вході ($T_{water,in} = 65\,°\mathrm{C}$), що є фізичною межею

ефективності теплообмінника.

Комплексна методика, що поєднує сурогатне моделювання на основі SVR, багатокритеріальну оптимізацію NSGA-II та вибір рішення методом TOPSIS, дозволила ідентифікувати оптимальні параметри пластинчасто-ребристого теплообмінника. З точки зору методології оптимізації, даний підрозділ ілюструє кілька ключових аспектів: (1) необхідність сурогатного моделювання, коли цільова функція задана неявно через обчислювально дорогу чисельну модель; (2) ефективність алгоритму NSGA-II для побудови фронту Парето у просторі трьох конфліктуючих цілей; (3) застосування MCDM-методу TOPSIS для прийняття остаточного рішення. Водночас обмеження підходу [25] полягає у використанні лише 45 навчальних точок для SVR, що може бути недостатнім для апроксимації складніших залежностей; більш щільна вибірка або адаптивне додавання точок могли б підвищити надійність сурогатної моделі.

1.3 Оптимальне проєктування вентильованого гальмівного диска за допомогою оптимізатора сірого вовка

У цьому підрозділі розглядається задача оптимізації геометричних параметрів вентильованого гальмівного диска за критерієм максимізації довговічності за наявності обмеження на осьовий прогин. На відміну від попередніх підрозділів, де застосовувались класичні алгоритми (RCA, NSGA-II), тут використано більш сучасний ройовий алгоритм – оптимізатор сірого вовка (англ. Grey Wolf Optimizer, GWO), а сурогатну модель побудовано у вигляді поліноміальної регресії другого порядку. Опис моделі базується на дослідженні Каліти, Шінде та Чакраборті [19].

Гальмівний диск під час експлуатації піддається циклічним термомеханічним навантаженням, що виникають внаслідок тертя колодки об робочу поверхню при повторному гальмуванні. Вентильована

конструкція диска складається з двох паралельних пластин (внутрішньої та зовнішньої), з'єднаних системою лопаток, між якими організовано повітряний потік для примусового охолодження. Циклічність навантажень зумовлює руйнування як основний механізм відмови, тому ключовим показником працездатності є *довговічність* (Fatigue Life) – кількість циклів навантаження до появи перших ознак руйнування (мікротріщин, спричинених циклічними термомеханічними навантаженнями).

Прямий пошук оптимальних геометричних параметрів за допомогою FEA-симуляцій є обчислювально недоцільним: кожна симуляція потребує побудови 3D-моделі в SolidWorks та проведення зв'язаного термомеханічного аналізу в ANSYS, що робить неможливим застосування популяційних алгоритмів, які потребують десятків тисяч обчислень цільової функції. Тому у роботі [19] запропоновано підхід на основі *метамоделювання*: за результатами обмеженого числа FEA-симуляцій будуються поліноміальні регресійні моделі, які слугують швидкими аналітичними замінниками для чисельного моделювання.

1.3.1 Огляд методів оптимізації конструкції гальмівного диска

Задача оптимального проєктування вентильованих гальмівних дисків активно досліджується впродовж останніх років. Різні авторські колективи застосовували різні комбінації методів планування експерименту, побудови метамоделей та алгоритмів оптимізації.

Белхосін, Шінде та Патіл [1] побудували *лінійну* регресійну модель довговічності на основі плану Тагучі L27 для п'яти геометричних параметрів і застосували два класичні еволюційні алгоритми – генетичний алгоритм (англ. Genetic Algorithm, GA) та оптимізацію роєм частинок (англ. Particle Swarm Optimization, PSO). Результати показали

покращення довговічності на 12,74% порівняно з параметричним аналізом. Однак лінійна модель не здатна відобразити взаємодії між параметрами та квадратичні ефекти, які є суттєвими у цій задачі [19].

Махешварі та ін. [27] дослідили ту саму конструкцію, але замість метаевристичних алгоритмів застосували чотири методи багатокритеріального прийняття рішень (MCDM-методи): EDAS, COPRAS, TOPSIS та ARAS. Повнофакторний план експерименту ($2^5 = 32$ варіанти) забезпечив рівномірне покриття простору параметрів. Проте MCDM-методи за своєю природою обмежені вибором із заздалегідь заданого дискретного набору альтернатив і не здатні знайти глобальний оптимум у неперервному просторі.

У роботі [19] було запропоновано суттєве розширення методології: замість плану Тагучі використано центральний композиційний план (Central Composite Design, CCD), побудовано поліноміальні регресійні метамоделі *другого порядку* для двох відгуків – довговічності та осьового прогину – і застосовано оптимізатор сірого вовка (GWO). Результати GWO показали приблизно 20% покращення порівняно з базовою конструкцією та 21% покращення порівняно з найкращими MCDM-рішеннями [19]. Детальний аналіз цієї роботи наведено у підрозділах 1.3.2–1.3.6.

Аналіз наведених робіт засвідчує: задача оптимізації гальмівного диска є типовою задачею пошуку глобального екстремуму на сурогатній моделі, побудованій за результатами обчислювально витратних FEA-симуляцій. Перевага підходу [19] полягає у поєднанні більш інформативного плану експерименту (CCD), нелінійної метамоделі та сучасного ройового алгоритму, що разом забезпечують більш точне та ефективне розв'язання задачі.

1.3.2 Змінні проєктування та планування чисельних експериментів

Об'єктом дослідження є вентильований гальмівний диск із сірого чавуну FG 15. Конструкція диска визначається п'ятьма геометричними параметрами, що виступають *змінними проєктування* [19]:

$$\mathbf{x} = (T_i,\, T_o,\, H_v,\, E_o,\, r)^\top \in \mathbb{R}^5, \tag{1.34}$$

де T_i [мм] – товщина внутрішньої пластини; T_o [мм] – товщина зовнішньої пластини ; H_v [мм] – висота вентиляційної лопатки; E_o [мм] – ефективне зміщення диска (відстань від площини кріплення до робочої поверхні); r [мм] – радіус центрального отвору кріплення. Простір пошуку $\mathcal{X}$ задається граничними-обмеженнями:

$$\mathcal{X} = [3, 11] \times [3, 11] \times [4, 8] \times [16, 40] \times [20, 30]. \tag{1.35}$$

Для побудови поліноміальної метамоделі другого порядку необхідно обрати *план експерименту* – набір точок у просторі параметрів, у яких проводитимуться FEA-симуляції. Повний перебір усіх комбінацій п'яти параметрів на п'яти рівнях потребував би $5^5 = 3125$ симуляцій, що є непрактичним. У роботі [1] для цієї ж конструкції використовувався план Тагучі L27, який забезпечує ефективну оцінку *головних ефектів* (тобто впливу кожного параметра окремо), але не дозволяє коректно оцінити попарні взаємодії між параметрами та квадратичні ефекти. Тому в [1] була побудована лише лінійна модель.

У [19] замість плану Тагучі використано **центральний композиційний план** (CCD) – план, спеціально сконструйований для побудови поліноміальних моделей другого порядку. CCD також містить 27 точок, але розміщує їх у просторі параметрів інакше: замість рівномірної сітки він складається з трьох цілеспрямованих груп точок.

Перша група – *факторні точки* ($2^{n-1} = 16$ комбінацій, де кожен параметр набуває значення -1 або $+1$ у кодованих одиницях) – забезпечує оцінку *попарних взаємодій* між параметрами. Друга група – *осьові точки* ($2n = 10$ точок, у кожній з яких лише один параметр відхиляється від центру на ± 2, а решта залишаються на нульовому рівні) – дозволяє оцінити *квадратичні ефекти*. Третя – одна *центральна точка*, де всі параметри дорівнюють середнім значенням [19]. Рівні змінних проєктування для кожного кодованого значення наведено в таблиці 1.5.

Таблиця 1.5 – Рівні змінних проєктування у CCD-плані [19]

Параметр	**Символ**	-2	-1	0	$+1$	$+2$
Товщина внутрішньої пластини (мм)	T_i	3	5	7	9	11
Товщина зовнішньої пластини (мм)	T_o	3	5	7	9	11
Висота лопатки (мм)	H_v	4	5	6	7	8
Ефективний виліт (мм)	E_o	16	22	28	34	40
Радіус центрального отвору (мм)	r	20	22,5	25	27,5	30

Таким чином, ключова відмінність CCD від Тагучі полягає не в кількості дослідів (обидва плани використовують 27 точок), а в їх *розташуванні*: Тагучі дозволяє визначити, *який параметр найважливіший*, тоді як CCD дозволяє побудувати *точну формулу*, що зв'язує параметри з відгуком, включаючи нелінійні ефекти та взаємодії [19].

Для кожної з 27 комбінацій параметрів проведено термомеханічне моделювання в ANSYS для оцінки довговічності. Результатом кожної симуляції є дві цільові характеристики: мінімальна довговічність $\mathrm{Life}(\mathbf{x})$ [цикли] та максимальний осьовий прогин $\delta(\mathbf{x})$ [мм]. Повні результати 27 симуляцій наведено в таблиці 1.6. Дані таблиці 1.6 формують навчальну вибірку $\mathcal{D} = \{(\mathbf{x}_i, y_i^{(1)}, y_i^{(2)})\}_{i=1}^{27}$, де $y_i^{(1)}$ – довговічність, $y_i^{(2)}$ – осьовий прогин i-го досліду. Ця вибірка використовується для побудови поліноміальних регресійних метамоделей у наступному підрозділі.

Таблиця 1.6 – CCD-план та результати FEA-симуляцій [19]

№	Кодовані значення					Натуральні значення					Довговічність	Прогин
	T_i	T_o	H_v	E_o	r	T_i	T_o	H_v	E_o	r	(цикли)	(мм)
1	0	2	0	0	0	7	11	6	28	25	708 090	0,00839
2	0	0	0	0	2	7	7	6	28	25	685 450	0,00799
3	−1	1	1	1	−1	5	9	7	22	22,5	706 310	0,01128
4	0	0	0	0	0	7	7	6	28	25	683 960	0,00807
5	1	1	1	1	1	9	9	7	34	27,5	853 910	0,00636
6	1	1	−1	1	−1	9	9	5	34	22,5	762 120	0,00622
7	0	0	0	0	−2	7	7	6	28	16	621 510	0,00817
8	−1	−1	1	1	1	5	5	7	34	27,5	607 820	0,01114
9	1	−1	1	1	−1	9	5	7	34	22,5	696 100	0,00635
10	−1	−1	−1	−1	1	5	5	5	22	27,5	491 910	0,01175
11	1	1	1	−1	−1	9	9	7	22	22,5	779 220	0,00737
12	0	−2	0	0	0	7	3	6	28	25	351 810	0,00856
13	0	0	2	0	0	7	7	8	28	25	763 460	0,00825
14	2	0	0	0	0	11	7	6	28	25	712 310	0,00562
15	−2	0	0	0	0	3	7	6	28	25	351 080	0,02190
16	1	−1	−1	−1	−1	9	5	5	22	22,5	620 650	0,00703
17	−1	1	−1	−1	−1	5	9	5	22	22,5	546 400	0,01224
18	1	−1	−1	1	1	9	5	5	34	27,5	488 460	0,00618
19	−1	1	−1	1	1	5	9	5	34	27,5	601 280	0,01105
20	1	1	−1	−1	1	9	9	5	22	27,5	776 830	0,00720
21	−1	−1	1	−1	−1	5	5	7	22	22,5	563 040	0,01213
22	1	−1	1	−1	1	9	5	7	22	27,5	715 420	0,00697
23	0	0	0	2	0	7	7	6	40	25	691 200	0,00754
24	−1	1	1	−1	1	5	9	7	22	27,5	641 660	0,01244
25	0	0	−2	0	0	7	7	4	28	25	793 720	0,00789
26	−1	−1	−1	1	−1	5	5	5	34	22,5	484 460	0,01112
27	0	0	0	−2	0	7	7	6	16	25	601 130	0,00934

1.3.3 Побудова поліноміальних сурогатних моделей

У роботі [19] побудовано дві поліноміальні регресійні моделі другого порядку, кожна з яких містить лінійні доданки, попарні взаємодії та квадратичні ефекти. Загальна форма такої моделі для відгуку $\hat{y}$:

$$\hat{y}(\mathbf{x}) = \beta_0 + \sum_{j=1}^{5} \beta_j \, x_j + \sum_{j=1}^{5} \sum_{k>j}^{5} \beta_{jk} \, x_j x_k + \sum_{j=1}^{5} \beta_{jj} \, x_j^2, \tag{1.36}$$

де β_0 – вільний член; β_j – коефіцієнти головних ефектів; β_{jk} – коефіцієнти попарних взаємодій; β_{jj} – коефіцієнти квадратичних ефектів.

Для довговічності конкретні значення коефіцієнтів мають вигляд [19]:

$$\begin{aligned}\widehat{\mathrm{Life}}(\mathbf{x}) = {} & 678\,143 + 73\,845{,}42\,T_i + 71\,351{,}25\,T_o + 30\,452{,}08\,H_v + \\ & + 10\,227{,}92\,E_o + 6\,119{,}58\,r + 18\,939{,}38\,T_iT_o + 113{,}13\,T_iH_v - \\ & - 15\,524{,}38\,T_iE_o - 4\,120{,}63\,T_ir - 12\,651{,}88\,T_oH_v + \\ & + 18\,355{,}63\,T_oE_o + 8\,766{,}88\,T_or + 16\,516{,}88\,H_vE_o + \\ & + 8\,080{,}63\,H_vr - 13\,376{,}88\,E_or - 32\,249{,}10\,T_i^2 - \\ & - 32\,685{,}34\,T_o^2 + 29\,474{,}66\,H_v^2 - 3\,631{,}6\,E_o^2 - 1\,802{,}85\,r^2.\end{aligned} \tag{1.37}$$

Ця модель містить 21 коефіцієнт (1 вільний + 5 лінійних + 10 взаємодій + 5 квадратичних), що суттєво відрізняється від лінійної моделі Белхосіна та ін. [1], яка мала лише 5 коефіцієнтів і не враховувала ні взаємодій, ні нелінійних ефектів. Аналіз коефіцієнтів (1.37) виявляє кілька важливих закономірностей. По-перше, найбільші лінійні коефіцієнти мають товщини пластин ($\beta_{T_i} = 73\,845$ та $\beta_{T_o} = 71\,352$), що підтверджує їх домінуючий вплив на довговічність. Аналіз Парето у [19] показав, що товщини внутрішньої та зовнішньої пластин разом пояснюють приблизно 80% варіації відгуку. По-друге, від'ємні квадратичні коефіцієнти при T_i^2 та T_o^2 ($-32\,249$ та $-32\,685$) свідчать про наявність *оптимуму*: занадто тонкі пластини мають недостатню структурну жорсткість, а занадто товсті – збільшену масу та перерозподіл термічних напружень. Ця нелінійна поведінка не може бути відображена лінійною моделлю [1]. По-третє, значущі коефіцієнти взаємодій (зокрема $\beta_{T_iT_o} = 18\,940$ та $\beta_{T_oE_o} = 18\,356$) означають, що оптимальне значення одного параметра залежить від значень інших, що унеможливлює знаходження оптимуму шляхом послідовної оптимізації кожного параметра окремо. Для осьового прогину метамодель має

вигляд [19]:

$$\begin{aligned}\hat{\delta}(\mathbf{x}) = {} & 0{,}008161 - 0{,}003002\,T_i + 0{,}000049\,T_o + 0{,}000082\,H_v - \\ & - 0{,}000459\,E_o - 0{,}000042\,r - 0{,}000016\,T_iT_o - \\ & - 0{,}000026\,T_iH_v - 0{,}000031\,T_iE_o - 0{,}000069\,T_ir + \\ & + 0{,}000015\,T_oH_v - 0{,}000077\,T_oE_o + 0{,}000032\,T_or - \\ & - 0{,}000007\,H_vE_o + 0{,}000013\,H_vr + 0{,}000011\,E_or + \\ & + 0{,}001330\,T_i^2 + 0{,}000009\,T_o^2 - 0{,}000093\,H_v^2 - \\ & - 0{,}0000006\,E_o^2 - 0{,}000091\,r^2,\end{aligned} \tag{1.38}$$

де $\hat{\delta}(\mathbf{x})$ [мм] – прогнозований осьовий прогин диска. Домінуючий коефіцієнт $-0{,}003002$ при T_i означає, що збільшення товщини внутрішньої пластини суттєво зменшує прогин, а додатний квадратичний коефіцієнт $+0{,}001330$ при T_i^2 свідчить про нелінійне «насичення» цього ефекту. Саме ця метамодель задає функціональне обмеження $\hat{\delta}(\mathbf{x}) \leq 0{,}05$ мм у задачі (1.39). Якість обох метамоделей оцінено за допомогою стандартних статистичних показників (табл. 1.7).

Таблиця 1.7 – Показники якості поліноміальних метамоделей [19]

Метрика	**Довговічність**	**Осьовий прогин**
R^2	92,61%	94,86%
R^2_{adj}	79,17%	85,52%
R^2_{pred}	94,17%	76,27%
Q^2_{F3} (незалежний тест)	96,34%	99,25%

$$Q^2_{F3} = 1 - \frac{\sum_{i=1}^{n_{test}} (y_i - \hat{y}_i)^2 / n_{test}}{\sum_{i=1}^{n_{train}} (y_i - \bar{y}_{train})^2 / n_{train}},$$

де: y_i та $\hat{y}_i$ – фактичне та прогнозоване значення відгуку для тестової вибірки; $\bar{y}_{train}$ – середнє значення відгуку у навчальній вибірці; n_{test} та n_{train} – кількість точок у тестовій та навчальній вибірках відповідно.

Особливо важливим є показник Q^2_{F3} – метрика якості метамоделі на *незалежному* тестовому датасеті (32 точки з FFD-плану [27]), що не

використовувався при побудові моделі. Значення $Q^2_{F3} = 96{,}34\%$ для довговічності підтверджує високу прогностичну здатність метамоделі [19].

На відміну від сурогатних моделей на базі методів машинного навчання (наприклад, SVR з підрозділу 1.2.4, що діє як «чорна скринька»), поліноміальна регресія (1.37) є повністю прозорою аналітичною моделлю. Вона дозволяє безпосередньо оцінити вагу кожного геометричного параметра та їхніх парних взаємодій, що значно полегшує математичний та інженерний аналіз системи.

1.3.4 Формулювання оптимізаційної задачі

На основі побудованих метамоделей задача проєктування вентильованого гальмівного диска формулюється як *задача однокритеріальної оптимізації з граничними-обмеженнями та функціональним обмеженням* [19]:

$$\boxed{\begin{aligned} &\max_{\mathbf{x}\in\mathcal{X}} \; f(\mathbf{x}) = \widehat{\text{Life}}(\mathbf{x}), \\ &\text{за обмежень:} \\ &\quad \hat{\delta}(\mathbf{x}) \le 0{,}05 \text{ мм}, \\ &\quad x_j^L \le x_j \le x_j^U,\; j = \overline{1,5}. \end{aligned}} \tag{1.39}$$

Ця постановка має кілька важливих відмінностей від задач, розглянутих у попередніх підрозділах. По-перше, задача (1.39) є *однокритеріальною з обмеженням*: осьовий прогин $\hat{\delta}(\mathbf{x})$ не є другим критерієм оптимізації (як, наприклад, у багатокритеріальній постановці (1.23)), а виступає *функціональним обмеженням* $g(\mathbf{x}) = \hat{\delta}(\mathbf{x}) - 0{,}05 \le 0$. Значення 0,05 мм обрано як гранично допустимий прогин для безпечної експлуатації [19].

По-друге, цільова функція (1.37) є *нелінійною* (квадратичною) відносно змінних проєктування, на відміну від лінійної моделі [1]. Наявність від'ємних квадратичних коефіцієнтів та значущих взаємодій

робить задачу (1.39) задачею *нелінійного програмування*, для якої градієнтні методи можуть потрапити до локальних екстремумів. Це обґрунтовує застосування глобального метаевристичного алгоритму.

По-третє, розмірність задачі є компактною ($n = 5$ змінних), що дозволяє популяційним алгоритмам ефективно обстежувати увесь допустимий простір за помірну кількість ітерацій.

1.3.5 Оптимізатор сірого вовка

Для розв'язання задачі (1.39) у [19] застосовано **оптимізатор сірого вовка** (Grey Wolf Optimizer, GWO) – метаевристичний популяційний алгоритм, запропонований Мірджалілі та ін. [29]. GWO моделює соціальну ієрархію та мисливську поведінку зграї сірих вовків.

Соціальна ієрархія. У зграї сірих вовків існує чітка ієрархія з чотирьох рівнів [29]: вовк-*альфа* (α) – лідер зграї, найкраще поточне рішення; *бета* (β) – другий за якістю розв'язок; *дельта* (δ) – третій; решта вовків належать до категорії *омега* (ω) і оновлюють свої позиції, орієнтуючись на трьох лідерів. Така структура забезпечує баланс між розвідкою нових ділянок простору та експлуатацією вже знайдених перспективних зон.

Оточення здобичі. На кожній ітерації t положення кожного вовка (розв'язку) $\vec{X}(t)$ оновлюється відносно положення здобичі $\vec{X}_p(t)$ за формулами [29]:

$$\vec{D} = \left|\vec{C} \cdot \vec{X}_p(t) - \vec{X}(t)\right|, \tag{1.40}$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} \cdot \vec{D}, \tag{1.41}$$

де $\vec{D}$ – вектор відстані до здобичі; $\vec{A}$ та $\vec{C}$ – коефіцієнтні вектори, що обчислюються як:

$$\vec{A} = 2\vec{a} \cdot \vec{r}_1 - \vec{a}, \qquad \vec{C} = 2 \cdot \vec{r}_2, \tag{1.42}$$

де $\vec{r}_1, \vec{r}_2 \sim U(0,1)$ – незалежні випадкові вектори з рівномірного розподілу; компоненти вектора $\vec{a}$ *лінійно зменшуються* від 2 до 0 протягом ітерацій [29]. Це зменшення є ключовим механізмом переходу від глобальної розвідки (при $|\vec{A}| > 1$ вовки віддаляються від здобичі) до локальної експлуатації (при $|\vec{A}| < 1$ вовки наближаються до здобичі).

Полювання. Оскільки справжнє положення глобального оптимуму (здобичі) невідоме, GWO використовує трьох найкращих агентів (α, β, δ) як орієнтири. Кожен ω-вовк оновлює свою позицію як середнє зважене трьох напрямних векторів [29]:

$$\vec{D}_\alpha = |\vec{C}_1 \cdot \vec{X}_\alpha - \vec{X}|, \quad \vec{D}_\beta = |\vec{C}_2 \cdot \vec{X}_\beta - \vec{X}|, \quad \vec{D}_\delta = |\vec{C}_3 \cdot \vec{X}_\delta - \vec{X}|, \quad (1.43)$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 \vec{D}_\alpha, \quad \vec{X}_2 = \vec{X}_\beta - \vec{A}_2 \vec{D}_\beta, \quad \vec{X}_3 = \vec{X}_\delta - \vec{A}_3 \vec{D}_\delta, \quad (1.44)$$

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3}. \quad (1.45)$$

Формула (1.45) є центральним механізмом GWO: оновлена позиція кожного вовка визначається як середнє трьох позицій, обчислених відносно α, β та δ. Це забезпечує одночасне врахування інформації від трьох найкращих розв'язків, що суттєво відрізняє GWO від PSO, де використовується лише один глобальний найкращий розв'язок.

У [19] проведено попередній аналіз впливу кількості пошукових агентів на якість оптимізації. Кількість агентів варіювалась від 50 до 1000, при цьому загальне число обчислень цільової функції фіксувалось на рівні $\approx 100\,000$. За результатами 30 незалежних запусків обрано $N_{pop} = 850$ агентів та 118 ітерацій як оптимальну конфігурацію [19].

1.3.6 Результати оптимізації

Результати оптимізації GWO для 30 незалежних запусків наведено в таблиці 1.8. Мале стандартне квадратичне відхилення (SD) оптимального

Таблиця 1.8 – Результати GWO-оптимізації для 30 незалежних запусків [19]

	T_i (мм)	T_o (мм)	H_v (мм)	E_o (мм)	r (мм)	**Довговічність** (цикли)	**Прогин** (мм)
Best	9,1420	10,6617	7,9999	39,9925	29,6181	1 024 160	0,00473
Mean	9,1975	10,5484	7,997	39,9884	28,5304	1 023 297	0,00483
SD	0,0776	0,1539	0,0003	0,0076	1,3890	1 444	0,00011

значення цільової функції (СКВ = 1 444 циклів при середньому 1 023 297, тобто відносна варіація лише 0,14%) свідчить про високу надійність та відтворюваність результатів GWO [19]. Аналіз збіжності показав, що GWO досягає 11% покращення вже на першій ітерації та виходить на рівень 20% покращення до 107-ї ітерації [19]. Кожен із 30 запусків GWO виконувався приблизно 45 секунд у MATLAB. Порівняння результатів GWO з MCDM-рішеннями наведено в таблиці 1.9.

Таблиця 1.9 – Порівняння GWO з MCDM-методами оптимізації [19, 27]

Метод	**Довговічність (цикли)**	**Покращення GWO**	**Прогин (мм)**
GWO [19]	1 024 160	—	0,00473
EDAS / TOPSIS / ARAS [27]	843 404	21%	0,00590
COPRAS [27]	769 322	33%	0,00563

Проведений аналіз дозволяє сформулювати кілька важливих спостережень щодо ідентифікації оптимальних параметрів.

По-перше, оптимальна конструкція гальмівного диска вимагає *майже максимальних значень* усіх параметрів, крім внутрішньої пластини (T_i = 9,1420 мм замість верхньої межі 11 мм) та радіуса отвору (r = 29,6281 мм). Висота лопатки та ефективний виліт знаходяться на верхніх межах (H_v = 7,9999 мм, E_o = 39,9984 мм). Цей результат не міг бути отриманий за допомогою MCDM-методів, обмежених дискретним набором заздалегідь обчислених альтернатив [19].

По-друге, перехід від лінійної регресії [1] до поліноміальної моделі

другого порядку (1.37) дозволив виявити нелінійні залежності (квадратичні ефекти та взаємодії), що є критично важливими для точної ідентифікації оптимуму.

По-третє, GWO як ройовий алгоритм з ієрархічною структурою (α–β–δ–ω) продемонстрував високу ефективність: 20% покращення порівняно з базовою конструкцією та 21% порівняно з MCDM-рішеннями.

Висновки до розділу 1

У першому розділі продемонстровано, що основою сучасного проєктування складних технічних систем є строга математична формалізація. Аналіз трьох різнорідних інженерних об'єктів підтвердив: незалежно від фізичної природи процесів, ключем до знаходження ефективних параметрів є коректна постановка задачі у вигляді оптимізаційної моделі з чітко визначеними цільовими функціями та обмеженнями. Для енергетичних комплексів вдалося визначити потужності компонентів та оптимальні сценарії їхньої роботи, що забезпечують мінімальну питому вартість енергії. У задачах проєктування теплообмінників знайшли баланс між суперечливими технічними вимогами, використовуючи багатокритеріальні методи пошуку та математичні інструменти прийняття рішень. Для гальмівних систем показано, що сучасні метаевристики дозволяють ідентифікувати таку геометрію диска, яка максимізує його довговічність за умови дотримання жорстких експлуатаційних обмежень.

Системний пошук оптимальних характеристик дозволяє вийти за межі стандартних конструкторських шаблонів. Це дає змогу знаходити глобальні оптимуми та виявляти неочевидні закономірності, наприклад, асиметричні режими роботи, які суттєво підвищують загальну ефективність системи.

2 СУЧАСНІ ОПТИМІЗАТОРИ ПОШУКУ РОЗВ'ЯЗКІВ ІНЖЕНЕРНИХ ЗАДАЧ, ЯКІ МАЮТЬ ОПТИМІЗАЦІЙНЕ ФОРМУЛЮВАННЯ

У другому розділі роботи проведено аналіз сучасних метаевристичних алгоритмів оптимізації, що застосовуються для пошуку глобальних розв'язків інженерних задач в оптимізаційному формулюванні. Спочатку надано загальну характеристику метаевристичних методів, після чого детально розглянуто чотири алгоритми ройового інтелекту: алгоритм пошуку горобців, оптимізатор гнойового жука, алгоритм рою тунікатівта алгоритм птаха-секретаря.

2.1 Загальна характеристика метаевристичних методів оптимізації

Як було показано у першому розділі, складні інженерні задачі оптимізації відзначаються нелінійністю цільових функцій, високою розмірністю простору параметрів і наявністю жорстких обмежень. Традиційні методи розв'язання таких задач концептуально поділяються на детерміністичні та стохастичні. Детерміністичні методи, що спираються на обчислення градієнта, наприклад, метод найшвидшого спуску чи метод Ньютона, забезпечують швидку збіжність, але вимагають неперервності та диференційовності цільової функції.

Однак значна частина реальних інженерних моделей описується *недиференційовними* (негладкими) функціями. У точках зламу, розриву або на «ступінчастих» ландшафтах градієнт просто не визначений, що робить використання градієнтних стратегій принципово неможливим. Крім того, такі методи вкрай чутливі до вибору початкової точки і схильні до застрягання у локальних оптимумах.

Для задач із дискретними змінними або випадків, коли цільова функція оцінюється неявно через чисельне моделювання (такі як FEA, CFD), аналітичний розрахунок похідних також є неможливим. У таких ситуаціях можуть застосовуватись безпохідні детерміністичні методи, зокрема метод Нелдера–Міда або метод Гука–Дживса [44]. Хоча вони здатні працювати з функціями типу «чорний ящик», їхня жорстко детермінована природа зберігає головний недолік: за відсутності механізмів глобального дослідження ці алгоритми так само легко застрягають у локальних мінімумах.

Ці обмеження зумовили розвиток *метаевристичних* алгоритмів – стохастичних стратегій пошуку, що не потребують інформації про градієнт і працюють лише зі значеннями цільової функції. Їхня ефективність базується на трьох фундаментальних рисах:

– **популяційний підхід**: одночасна робота з множиною розв'язків забезпечує паралельне дослідження різних ділянок простору параметрів;

– **стохастичність**: використання випадкових компонентів дозволяє агентам «вистрибувати» з локальних пасток;

– **кооперативність**: ітераційне оновлення позицій відбувається на основі обміну інформацією між агентами, що спрямовує пошук до найбільш перспективних зон.

Ключовою концепцією, що визначає ефективність метаевристичного алгоритму, є баланс між *розвідкою* (exploration) та *експлуатацією* (exploitation) [42]. Розвідка – здатність алгоритму досліджувати нові ділянки простору пошуку – домінує на ранніх ітераціях та запобігає передчасній збіжності до локального мінімуму. Експлуатація – детальне обстеження околиць вже знайдених перспективних рішень – стає визначальною на пізніх ітераціях для уточнення глобального оптимуму. Надмірна розвідка уповільнює збіжність, тоді як надмірна експлуатація призводить до *передчасної збіжності* – застрягання алгоритму у локальному мінімумі. Кожен із розглянутих далі алгоритмів реалізує власні механізми динамічного управління цим балансом.

Фундаментальним теоретичним обґрунтуванням доцільності розробки та порівняння різних алгоритмів є *теорема про відсутність безкоштовних обідів* (англ. No Free Lunch theorem, NFL), доведена Вольпертом та Макреді [40]. Теорема стверджує, що усереднена продуктивність будь-якого алгоритму по множині *всіх можливих* задач оптимізації є однаковою. Практичний наслідок полягає в тому, що не існує єдиного універсального оптимізатора: алгоритм, ефективний на одному класі задач, може виявитися непридатним для іншого. Цей результат безпосередньо мотивує структуру третього розділу даної роботи, де кілька алгоритмів порівнюються на конкретних інженерних задачах.

2.2 Ройовий алгоритм оптимізації, натхненний поведінкою горобців

Алгоритм пошуку горобців (англ. Sparrow Search Algorithm, SSA) – метаевристичний оптимізаційний алгоритм ройового інтелекту, запропонований Сюе та Шенем у 2020 році [41]. Алгоритм моделює колективну кормову поведінку та антихижацькі стратегії зграї горобців, поєднуючи механізми розвідки та експлуатації через розподіл ролей між агентами популяції.

2.2.1 Біологічне підґрунтя алгоритму

Горобці – суспільні птахи, поширені на більшості континентів, що утворюють стабільні зграї з ієрархічною організацією. В основу алгоритму покладено модель «виробник–нахлібник». Згідно з цією моделлю, в будь-якій зграї можна виділити два типи поведінки: *виробники* активно шукають нові джерела їжі, тоді як *нахлібники* слідкують за виробниками та намагаються скористатися знайденими

ними ресурсами. Етологічні дослідження засвідчують, що окремі особини гнучко переключаються між цими двома стратегіями залежно від рівня власних енергетичних резервів та успішності кормодобування [41].

Крім кормових взаємодій, поведінка горобців визначається антихижацькими реакціями. Особини, розташовані на периферії зграї, більш вразливі до нападу хижаків і прагнуть зайняти безпечнішу позицію ближче до центру. Коли один або кілька горобців виявляють хижака, вони видають характерний сигнал тривоги, після чого вся зграя здійснює колективну втечу [41]. Ця трирівнева модель поведінки – пошук їжі виробниками, конкурентне відстежування нахлібниками та антихижацька реакція дозорних – формує математичну основу SSA.

2.2.2 Математична модель алгоритму

Розглянемо задачу мінімізації цільової функції $f(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^d$, де d — розмірність простору пошуку. Нехай популяція складається з n горобців. Позиція i-го горобця на ітерації t є d-вимірним вектором $\mathbf{X}_i^t = (X_{i,1}^t, X_{i,2}^t, \ldots, X_{i,d}^t)$, а матриця позицій усієї популяції — матрицею розміру $n \times d$ [41]:

$$\mathbf{X} = \begin{pmatrix} X_{1,1} & X_{1,2} & \cdots & X_{1,d} \\ X_{2,1} & X_{2,2} & \cdots & X_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ X_{n,1} & X_{n,2} & \cdots & X_{n,d} \end{pmatrix}. \tag{2.1}$$

Кожному рядку матриці (2.1) відповідає значення цільової функції

$f_i = f(\mathbf{X}_i)$. Вектор значень пристосованості записується як [41]:

$$\mathbf{F}_X = \begin{pmatrix} f(\mathbf{X}_1) \\ f(\mathbf{X}_2) \\ \vdots \\ f(\mathbf{X}_n) \end{pmatrix}. \tag{2.2}$$

На кожній ітерації агенти ранжуються за значеннями f_i, після чого верхні P_D агентів (з найкращою пристосованістю) призначаються виробниками, а решта $n - P_D$ – нахлібниками. Крім того, S_D агентів випадково обираються як дозорні, що виконують антихижацьку функцію.

Оновлення позицій виробників. Виробники відповідають за розвідку простору пошуку і спрямовують рух усієї зграї. Їхні позиції оновлюються за правилом [41]:

$$X_{i,j}^{t+1} = \begin{cases} X_{i,j}^{t} \cdot \exp\left(\dfrac{-i}{\alpha \cdot T_{\max}}\right), & \text{якщо } R_2 < ST, \\ X_{i,j}^{t} + Q \cdot L, & \text{якщо } R_2 \geq ST, \end{cases} \tag{2.3}$$

де $j = 1, 2, \ldots, d$ – індекс виміру; $T_{\max}$ – максимальна кількість ітерацій; $\alpha \in (0, 1]$ – рівномірно розподілене випадкове число; $R_2 \in [0, 1]$ – значення тривоги, що моделює ступінь загрози від хижаків; $ST \in [0{,}5;\ 1{,}0]$ – поріг безпеки; Q – випадкова величина з нормальним розподілом $\mathcal{N}(0, 1)$; L – вектор розміру $1 \times d$, усі елементи якого дорівнюють одиниці.

Фізичний зміст рівняння (2.3) полягає в наступному. Коли $R_2 < ST$, тобто рівень загрози є низьким, виробник здійснює широкий пошук: множник $\exp(-i/(\alpha \cdot T_{\max}))$ забезпечує поступове зменшення амплітуди кроку з номером ітерації та рангом агента i, що реалізує перехід від розвідки до експлуатації. Коли $R_2 \geq ST$, один або кілька дозорних виявили хижака, і всі виробники здійснюють випадковий стрибок $Q \cdot L$, що відповідає негайній зміні напрямку руху для уникнення небезпеки [41].

Оновлення позицій нахлібників. Нахлібники ($i = P_D + 1, \ldots, n$)

слідкують за виробниками і конкурують за знайдені ресурси. Їхні позиції оновлюються за формулою [41]:

$$X_{i,j}^{t+1} = \begin{cases} Q \cdot \exp\left(\dfrac{X_{worst,j}^{t} - X_{i,j}^{t}}{i^2}\right), & \text{якщо } i > n/2, \\ X_{P,j}^{t+1} + \left|X_{i,j}^{t} - X_{P,j}^{t+1}\right| \cdot A^{+} \cdot L, & \text{інакше,} \end{cases} \tag{2.4}$$

де $\mathbf{X}_{worst}^{t}$ – позиція агента з найгіршим значенням пристосованості на ітерації t; $\mathbf{X}_{P}^{t+1}$ – оновлена позиція найкращого виробника; A – матриця розміру $1 \times d$, кожен елемент якої випадково набуває значення $+1$ або -1; $A^{+} = A^{\top}(AA^{\top})^{-1}$ – псевдообернена матриця Мура–Пенроуза.

У першому випадку ($i > n/2$) нахлібник із низьким рангом (тобто поганою пристосованістю) здійснює великий стрибок у бік, протилежний найгіршому розв'язку, що моделює поведінку голодного горобця, який залишає неперспективну ділянку в пошуках кращих ресурсів. У другому випадку нахлібник із відносно високим рангом рухається в околиці найкращого виробника, що відповідає конкурентному відстежуванню та спробі перехопити знайдену їжу [41].

Антихижацький механізм. Для моделювання антихижацької поведінки S_D агентів (зазвичай 10–20% від загальної чисельності) обираються випадково та оновлюють свої позиції за правилом [41]:

$$X_{i,j}^{t+1} = \begin{cases} X_{best,j}^{t} + \beta \cdot \left|X_{i,j}^{t} - X_{best,j}^{t}\right|, & \text{якщо } f_i > f_g, \\ X_{i,j}^{t} + K \cdot \dfrac{\left|X_{i,j}^{t} - X_{worst,j}^{t}\right|}{(f_i - f_w) + \varepsilon}, & \text{якщо } f_i = f_g, \end{cases} \tag{2.5}$$

де $\mathbf{X}_{best}^{t}$ – глобально найкраща позиція на ітерації t; $\beta \sim \mathcal{N}(0,1)$ – параметр контролю розміру кроку; $K \in [-1,1]$ – випадкове число, що визначає напрямок руху; f_g та f_w – відповідно найкраще та найгірше значення пристосованості у поточній популяції; ε – мала додатна константа для запобігання діленню на нуль.

Умова $f_i > f_g$ (для задачі мінімізації це означає, що агент має гіршу

пристосованість, ніж глобальний лідер) відповідає ситуації, коли горобець перебуває на периферії зграї та рухається до центру – найбезпечнішої позиції $\mathbf{X}_{best}^{t}$. Умова $f_i = f_g$ моделює ситуацію, коли горобець, що знаходиться поблизу центру, усвідомлює загрозу та здійснює випадкове переміщення для зменшення області небезпеки. Узагальнену процедуру SSA подано в алгоритмі 2.1:

Алгоритм 2.1 Алгоритм пошуку горобців (SSA)

Вхід: $T_{\max}$ — макс. кількість ітерацій; n — розмір популяції; P_D — кількість виробників; S_D — кількість дозорних

Вихід: $\mathbf{X}_{best}$ — найкраща знайдена позиція; f_g — відповідне значення ЦФ

Ініціалізувати популяцію $\{\mathbf{X}_i\}_{i=1}^{n}$ випадково у допустимій області
$t \leftarrow 0$
while $t < T_{\max}$ **do**
    Обчислити $f_i = f(\mathbf{X}_i)$ для кожного i; визначити $\mathbf{X}_{best}$, $\mathbf{X}_{worst}$
    Згенерувати $R_2 = \mathrm{rand}(0, 1)$
    for $i = 1, \ldots, P_D$ **do**
        Оновити $\mathbf{X}_i$ за рівнянням (2.3) ▷ Виробники
    end for
    for $i = P_D + 1, \ldots, n$ **do**
        Оновити $\mathbf{X}_i$ за рівнянням (2.4) ▷ Нахлібники
    end for
    for $l = 1, \ldots, S_D$ **do**
        Оновити $\mathbf{X}_l$ за рівнянням (2.5) ▷ Дозорні
    end for
    Перевірити межі допустимої області; зберегти нову позицію, якщо $f(\mathbf{X}_i^{t+1}) < f(\mathbf{X}_i^{t})$
    $t \leftarrow t + 1$
end while
return $\mathbf{X}_{best}$, f_g

2.2.3 Результати тестування алгоритму

Ефективність SSA верифіковано авторами алгоритму на 19 стандартних тестових функціях (7 унімодальних F1–F7 та

12 мультимодальних F8–F19) у порівнянні з трьома відомими метаевристиками: оптимізатором сірого вовка, оптимізацією роєм частинок та гравітаційним пошуковим алгоритмом. Параметри експерименту: розмір популяції $n = 100$, максимальна кількість ітерацій $T_{\max} = 1000$, кількість незалежних запусків – 30. Розмірність простору пошуку для унімодальних та мультимодальних функцій складала $d = 30$, для функцій із фіксованою розмірністю – від 2 до 6 [41].

Узагальнені результати бенчмаркінгу наведено у табл. 2.1:

Таблиця 2.1 – Узагальнені результати порівняння SSA з GWO, PSO та GSA на 19 тестових функціях [41]

Критерій	**SSA**	**GWO**	**PSO**	**GSA**
Кількість функцій із найкращим середнім значенням (з 19)	15	2	0	2
Кількість функцій із найменшим стандартним відхиленням (з 19)	14	2	1	2

На унімодальних функціях (F1–F7), що характеризують здатність алгоритму до експлуатації, SSA продемонстрував перевагу: на функціях F1–F4 алгоритм досяг глобального мінімуму (0,0) з нульовим стандартним відхиленням, тоді як найближчий конкурент GWO – лише порядку 10^{-88} на F1. На мультимодальних функціях (F8–F19), що тестують здатність до розвідки та уникнення локальних пасток, SSA отримав найкраще середнє значення на 9 з 12 функцій та продемонстрував найвищу стабільність (найменше стандартне відхилення) також на 9 з 12 [41].

Окрім тестових функцій, авторами проведено верифікацію на двох практичних інженерних задачах оптимізації з обмеженнями: задачі нелінійної оптимізації Гіммельблау (5 змінних, 6 нелінійних обмежень) та задачі проєктування редуктора швидкості (7 змінних, 11 обмежень). Для обробки обмежень-нерівностей використовувався метод штрафних функцій. В обох випадках SSA знайшов допустимі розв'язки з високою

повторюваністю, що підтверджує ефективність алгоритму при розв'язанні задач інженерного проєктування зі змішаними змінними та нелінійними обмеженнями [41].

Таблиця 2.2 – Результати оптимізації для практичних інженерних задач за допомогою алгоритму SSA [41]

Задача	**Best**	**Worst**	**Mean**	**Std**
Нелінійна оптимізація Гіммельблау	−30665,5387	−30662,8505	−30665,3808	0,5713
Проєктування редуктора швидкості	2996,7077	3008,1638	2997,7101	2,8209

2.2.4 Аналіз балансу розвідки та експлуатації

Розвідка простору пошуку реалізується переважно через виробників: при низькому рівні тривоги ($R_2 < ST$) вони здійснюють широкомасштабний пошук із поступовим зменшенням амплітуди кроку завдяки експоненціальному загасанню в (2.3). Додатковий внесок у розвідку забезпечують нахлібники з низьким рангом ($i > n/2$), що здійснюють великі стрибки згідно з першою гілкою рівняння (2.4).

Експлуатація перспективних ділянок забезпечується нахлібниками з високим рангом ($i \leq n/2$), що рухаються в околиці найкращого виробника за другою гілкою (2.4), та дозорними, що концентруються навколо глобального оптимуму $\mathbf{X}_{best}$ за рівнянням (2.5).

Антихижацький механізм виконує подвійну функцію: з одного боку, він запобігає передчасній збіжності, вносячи стохастичні збурення у позиції дозорних; з іншого – сприяє уточненню глобального оптимуму, спрямовуючи периферійних агентів до центру зграї.

Водночас SSA має певні обмеження. Фіксовані частки виробників та нахлібників не адаптуються до ландшафту цільової функції, що може знижувати ефективність на задачах зі складною топологією. Ці обмеження мотивують численні модифікації SSA, серед яких варто

відзначити інтеграцію блукань Леві для покращення розвідки, хаотичних послідовностей для ініціалізації популяції, а також адаптивне управління часткою виробників [39].

2.3 Ройовий оптимізатор гнойового жука

Оптимізатор гнойового жука (англ. Dung Beetle Optimizer, DBO) – метаевристичний алгоритм ройового інтелекту, запропонований Сюе та Шенем у 2022 році [42]. Алгоритм моделює п'ять типів поведінки гнойових жуків: кочення кулі, танець, розмноження, годування та крадіжку. Кожному типу поведінки відповідає окрема стратегія оновлення позицій агентів, що забезпечує ефективний баланс між розвідкою та експлуатацією простору пошуку.

2.3.1 Біологічне підґрунтя алгоритму

Гнойові жуки – комахи, поширені на більшості континентів, що відіграють ключову роль у екосистемі як деструктори. Їхня характерна особливість – формування сфери з гною тварин та транспортування її до місця гніздування. При коченні кулі жук орієнтується за небесними тілами (Сонцем, Місяцем, поляризованим світлом), що дозволяє рухатися по прямолінійній траєкторії. Якщо джерело світла відсутнє, траєкторія жука стає криволінійною або навіть кільцевою. У разі зустрічі з перешкодою жук піднімається на вершину кулі та виконує характерний «танець» – серію обертів та пауз, що дозволяє переорієнтуватися та обрати новий напрямок.

Друга важлива поведінка – розмноження. Самка закопує кулю у ґрунт та відкладає в неї яйця; куля слугує одночасно місцем розвитку та джерелом живлення для личинок. Після дорослішання молоді жуки виходять на поверхню та здійснюють кормовий пошук в околі

оптимальних ресурсів. Окрім того, частина популяції демонструє клептопаразитичну поведінку – крадіжку куль, виготовлених іншими особинами. Ця п'ятирівнева модель поведінки – кочення, танець, розмноження, годування та крадіжка – формує математичну основу алгоритму DBO.

2.3.2 Математична модель алгоритму

Розглянемо задачу мінімізації цільової функції $f(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^D$, де D – розмірність простору пошуку. Нехай популяція складається з N агентів. Позиція i-го агента на ітерації t є D-вимірним вектором $\mathbf{x}_i(t) = (x_{i1}(t), x_{i2}(t), \ldots, x_{iD}(t))$. Популяція поділяється на чотири функціональні групи: жуки-котили (англ. ball-rolling beetles), виводкові кулі (англ. brood balls), молоді жуки (small beetles) та крадії (англ. thieves). За замовчуванням рекомендовано пропорцію $20:20:25:35$ [18].

Оновлення позицій жуків-котил. Жук-котила переміщує сферу з гною у заданому напрямку, орієнтуючись за небесним джерелом світла. Математична модель цього процесу має вигляд [42]:

$$x_i(t+1) = x_i(t) + \alpha \cdot k \cdot x_i(t-1) + b \cdot \Delta x, \tag{2.6}$$

де $\Delta x = |x_i(t) - X^w|$ – міра флуктуації інтенсивності світла; X^w – позиція глобально найгіршого розв'язку в поточній популяції; $k \in (0;\, 0{,}2]$ – коефіцієнт відхилення, що моделює вплив природних факторів (вітер, нерівності ґрунту); $b \in (0;\, 1)$ – вагова стала; α – натуральний коефіцієнт, що приймає значення $+1$ або -1 за стохастичним правилом [42]:

$$\alpha = \begin{cases} 1, & \text{якщо } \eta > \lambda, \\ -1, & \text{інакше,} \end{cases} \tag{2.7}$$

де $\eta = \mathrm{rand}(0{,}1)$ – рівномірно розподілена випадкова величина; λ – поріг

імовірності. Значення $\alpha = 1$ відповідає руху без відхилення від початкового напрямку, $\alpha = -1$ – відхиленню [42]. Більше значення Δx відповідає слабшому джерелу світла: чим нижча якість найгіршого розв'язку, тим більшим стає крок пошуку, що сприяє розвідці нових ділянок простору [42]. Коли жук зустрічає перешкоду, він переорієнтується за допомогою танцю. Позиція оновлюється за формулою [42]:

$$x_i(t+1) = x_i(t) + \tan(\theta) \cdot |x_i(t) - x_i(t-1)|, \tag{2.8}$$

де $\theta \in [0; \pi]$ – випадковий кут повороту. При $\theta = 0$, $\theta = \pi/2$ або $\theta = \pi$ позиція жука залишається незмінною [42].

Перемикання між коченням (2.6) та танцем (2.8) здійснюється стохастично: генерується $\delta = \text{rand}(0,1)$; якщо $\delta < 0{,}9$, застосовується правило кочення, інакше – правило танцю [42].

Оновлення позицій виводкових куль. Самки гнойових жуків обирають захищену ділянку для відкладання яєць. Область нересту визначається динамічно навколо поточного локального найкращого розв'язку X^* [42]:

$$\begin{aligned} Lb^* &= \max\big(X^* \cdot (1-R),\, Lb\big), \\ Ub^* &= \min\big(X^* \cdot (1+R),\, Ub\big), \end{aligned} \tag{2.9}$$

де Lb, Ub – нижня та верхня межі простору пошуку відповідно; $R = 1 - t/T_{\max}$ – параметр звуження, що лінійно зменшується від 1 до 0 протягом $T_{\max}$ ітерацій. Динамічна зміна R забезпечує перехід від широкої розвідки (великий R на початку) до точної експлуатації (малий R наприкінці) [42].

Позиція i-ї виводкової кулі оновлюється за формулою [42]:

$$B_i(t+1) = X^* + b_1 \cdot \big(B_i(t) - Lb^*\big) + b_2 \cdot \big(B_i(t) - Ub^*\big), \tag{2.10}$$

де $B_i(t)$ – позиція i-ї виводкової кулі на ітерації t; b_1, b_2 – два незалежні випадкові вектори розмірності $1 \times D$. Позиція кулі суворо обмежується

рамками області нересту: якщо будь-яка компонента виходить за межі $[Lb^*, Ub^*]$, вона замінюється відповідною граничною величиною.

Оновлення позицій молодих жуків. Молоді жуки, що досягли зрілості, виходять з кулі та починають кормовий пошук. Область пошуку формується аналогічно до області нересту, але центрується навколо глобально найкращого розв'язку X^b:

$$\begin{aligned} Lb^b &= \max\big(X^b \cdot (1 - R),\, Lb\big), \\ Ub^b &= \min\big(X^b \cdot (1 + R),\, Ub\big). \end{aligned} \tag{2.11}$$

Позиція i-го молодого жука оновлюється за правилом:

$$x_i(t+1) = x_i(t) + C_1 \cdot \big(x_i(t) - Lb^b\big) + C_2 \cdot \big(x_i(t) - Ub^b\big), \tag{2.12}$$

де C_1 – випадкова величина з нормальним розподілом $\mathcal{N}(0{,}1)$; C_2 – випадковий вектор з рівномірного розподілу на (0; 1).

Оновлення позицій крадіїв. Частина популяції поводиться як клептопаразити, прямуючи до оптимального джерела ресурсів X^b:

$$x_i(t+1) = X^b + S \cdot g \cdot \big(|x_i(t) - X^*| + |x_i(t) - X^b|\big), \tag{2.13}$$

де g – випадковий вектор розмірності $1 \times D$ з нормальним розподілом $\mathcal{N}(0{,}1)$; S – стала величина. Присутність обох доданків $|x_i(t) - X^*|$ та $|x_i(t) - X^b|$ дозволяє крадієві враховувати як локальну, так і глобальну інформацію, підтримуючи різноманітність популяції навіть при наближенні до оптимуму [42].

Узагальнену процедуру DBO подано в алгоритмі 2.2.

Алгоритм 2.2 Оптимізатор гнойового жука (DBO)

Вхід: $T_{\max}$ – макс. кількість ітерацій; N – розмір популяції

Вихід: X^b – найкраща знайдена позиція; f_b – відповідне значення ЦФ

Ініціалізувати популяцію $\{\mathbf{x}_i\}_{i=1}^{N}$ випадково у $[Lb, Ub]^D$
Розподілити агентів на 4 групи: жуки-котили, виводкові кулі, молоді жуки, крадії
$t \leftarrow 1$
while $t \leq T_{\max}$ **do**
    Обчислити $f_i = f(\mathbf{x}_i)$ для кожного i; визначити X^b, X^*, X^w
    for $i \in$ жуки-котили **do**
        $\delta \leftarrow \text{rand}(0,1)$
        if $\delta < 0{,}9$ **then**
            Обрати α за (2.7); оновити $\mathbf{x}_i$ за (2.6) ▷ Кочення
        else
            Оновити $\mathbf{x}_i$ за (2.8) ▷ Танець
        end if
    end for
    for $i \in$ виводкові кулі **do**
        Обчислити межі (2.9); оновити B_i за (2.10)
        Обрізати B_i до $[Lb^*, Ub^*]$
    end for
    for $i \in$ молоді жуки **do**
        Обчислити межі (2.11); оновити $\mathbf{x}_i$ за (2.12)
    end for
    for $i \in$ крадії **do**
        Оновити $\mathbf{x}_i$ за (2.13)
    end for
    Перевірити допустимість позицій; оновити X^b, X^*, X^w
    $t \leftarrow t + 1$
end while
return X^b, f_b

2.3.3 Результати тестування алгоритму

Ефективність DBO було верифіковано авторами на двох наборах тестових функцій: 23 класичних бенчмарках (унімодальні F1–F7,

мультимодальні F8–F13, функції з фіксованою розмірністю F14–F23) та 29 функціях CEC-BC-2017 [42]. Порівняння проводилося із сімома метаевристиками: HHO, PSO, WOA, MVO, SCA, SSA та GWO. Узагальнені результати наведено у табл. 2.3 та 2.4.

Таблиця 2.3 – Узагальнені результати порівняння DBO з сімома алгоритмами на 23 класичних тестових функціях (тест Вілкоксона та тест Фрідмана) [42]

Критерій	**DBO**	**HHO**	**GWO**	**PSO**	**WOA**	**SSA**	**MVO**	**SCA**
+/-/ = (DBO vs)	–	14/5/4	16/3/4	13/4/6	14/1/8	15/1/7	16/2/5	23/0/0
Ранг Фрідмана	**2,370**	3,109	4,087	4,565	4,761	4,935	4,978	7,065
Позиція	**1**	2	3	4	5	6	7	8

Таблиця 2.4 – Узагальнені результати порівняння DBO з сімома алгоритмами на 29 функціях CEC-BC-2017 (тест Фрідмана) [42]

Критерій	**DBO**	**SSA**	**MVO**	**PSO**	**GWO**	**SCA**	**HHO**	**WOA**
+/-/ = (DBO vs)	–	10/5/14	13/9/7	14/9/6	13/7/9	21/1/7	20/1/8	23/0/6
Ранг Фрідмана	**2,672**	3,466	3,724	4,207	4,466	5,534	5,586	6,345
Позиція	**1**	2	3	4	5	6	7	8

У табл. 2.3 рядок «+/-/ =» показує, на скількох функціях DBO *статистично значущо* перевищив (+), поступився (−) або не відрізнявся (=) від конкурента за тестом Вілкоксона зі знаком рангу на рівні значущості 5% [42]. DBO є статистично кращим за SCA на всіх 23 функціях та перевищує GWO, MVO, SSA на 15–16 функціях. На обох наборах DBO посів перше місце за тестом Фрідмана (ранги 2,370 та 2,672 відповідно), що підтверджує збалансованість його пошукових стратегій [42].

На унімодальних функціях F1–F4 DBO досяг середніх значень порядку 10^{-56}–10^{-104}, суттєво перевершивши найближчого конкурента HHO (10^{-47}–10^{-97}). На мультимодальних функціях F9 та F11 DBO

знайшов точний глобальний оптимум (0,0). На наборі CEC-BC-2017 DBO посів перше місце на гібридних функціях f13, f17 та f20, хоча відносна перевага скоротилася порівняно з класичними бенчмарками [42].

Крім бенчмаркових функцій, DBO було верифіковано на трьох класичних інженерних задачах з обмеженнями (трьохстержнева ферма, посудина під тиском, зварна балка), де алгоритм знайшов конкурентоспроможні або найкращі розв'язки серед усіх порівнюваних метаевристик [42].

2.3.4 Аналіз балансу розвідки та експлуатації та обмеження базового DBO

Ефективність DBO зумовлена взаємодією чотирьох стратегій оновлення, що забезпечують динамічний баланс між розвідкою та експлуатацією [42]. Розвідку простору пошуку реалізують переважно жуки-котили та крадії: правило кочення (2.6) генерує великі кроки пошуку завдяки залежності від найгіршого розв'язку X^w, а правило танцю (2.8) забезпечує стохастичну переорієнтацію агента. Експлуатацію перспективних ділянок здійснюють виводкові кулі та молоді жуки, чиї області пошуку (2.9), (2.11) звужуються з кожною ітерацією завдяки монотонному спаданню параметра $R = 1 - t/T_{\max}$ [42].

Водночас подальші дослідження виявили три суттєвих обмеження базового DBO [18]: (1) рівномірна випадкова ініціалізація не гарантує рівномірного покриття простору пошуку, що знижує глобальну пошукову здатність на ранніх ітераціях; (2) модель кочення (2.6) передбачає лише прямолінійний рух під керівництвом джерела світла, ігноруючи криволінійну поведінку жука за відсутності світлових орієнтирів, що обмежує різноманітність стратегій пошуку; (3) фіксовані пропорції розподілу агентів між групами $(20:20:25:35)$ не дозволяють адаптивно перерозподіляти ресурси між розвідкою та експлуатацією впродовж

оптимізаційного процесу. Ці обмеження мотивували розробку вдосконаленої версії – алгоритму DPDBO, що детально розглядається у наступному розділі.

2.4 Вдосконалений алгоритм DBO з динамічною популяцією (DPDBO)

Алгоритм Dynamic Population Dung Beetle Optimization (скор. DPDBO) запропоновано у [18] у 2025 році як багатостратегійну гібридну модифікацію базового DBO. DPDBO інтегрує три стратегії вдосконалення, кожна з яких спрямована на усунення одного з виявлених обмежень базового алгоритму [18].

2.4.1 Ініціалізація популяції на основі PWLCM

Якість початкової популяції суттєво впливає на швидкість збіжності та глобальну пошукову здатність метаевристичного алгоритму. Базовий DBO використовує рівномірну випадкову ініціалізацію, що може призводити до нерівномірного покриття простору пошуку. Для подолання цього обмеження у DPDBO застосовано кусково-лінійне хаотичне відображення (англ. Piecewise Linear Chaotic Map, PWLCM), яке характеризується вищою випадковістю та ергодичністю порівняно з іншими хаотичними відображеннями (Tent, Logistic, Sine) завдяки механізму сегментації простору відображення [18]. PWLCM визначається

формулою [18]:

$$f(i+1) = \begin{cases} \dfrac{f(i)}{p}, & 0 \le f(i) < p, \\ \dfrac{f(i)-p}{0{,}5-p}, & p \le f(i) < 0{,}5, \\ \dfrac{1-p-f(i)}{0{,}5-p}, & 0{,}5 \le f(i) < 1-p, \\ \dfrac{1-f(i)}{p}, & 1-p \le f(i) < 1, \end{cases} \tag{2.14}$$

де $p \in [0;\, 0{,}5]$ – керуючий параметр хаотичного відображення, що визначає структуру сегментації; $f(i)$ – значення хаотичної послідовності на i-му кроці.

Процедура ініціалізації складається з трьох етапів: задання початкових значень p та $f(0) \in [0;\, 1]$; генерація хаотичної послідовності z_{ij} $(i = 1, 2, \ldots, N;\ j = 1, 2, \ldots, D)$ за допомогою ітерацій (2.14); перетворення отриманої послідовності у простір розв'язків за лінійним масштабуванням [18]:

$$x_{ij} = lb_j + (ub_j - lb_j) \cdot z_{ij}, \tag{2.15}$$

де x_{ij} – координата i-го жука у j-му вимірі; lb_j, ub_j – нижня та верхня межі j-го виміру.

2.4.2 Моделювання дезорієнтаційної поведінки жуків-котил

У базовому DBO жук-котила рухається прямолінійно, орієнтуючись за джерелом світла (2.6). Проте етологічні дослідження засвідчують, що за відсутності світлових орієнтирів (повна темрява) жуки втрачають здатність до точної навігації та рухаються по криволінійних або навіть еліптичних траєкторіях [18]. Для моделювання такої *дезорієнтаційної*

поведінки у DPDBO введено спіральну стратегію пошуку [18]:

$$x_i(t+1) = x_i(t) + b \cdot l \cdot \cos(2\pi l) \cdot |x_i(t) - X^w| + \alpha \cdot k \cdot x_i(t-1), \quad (2.16)$$

де $b \cdot l \cdot \cos(2\pi l)$ – спіральний множник, що імітує гвинтоподібний рух дезорієнтованого жука; b – стала, що визначає крок спіралі (більше значення b відповідає ширшому пошуку, але нижчій точності); $l \in [-1; 1]$ – випадковий параметр; α, k – параметри, визначені аналогічно до базового DBO (2.6).

Перемикання між прямолінійною (2.6)–(2.8) та спіральною (2.16) стратегіями здійснюється стохастично за допомогою ймовірності p_1 [18]:

$$\begin{cases} \text{рівняння (2.6)–(2.8)}, & \text{якщо } \alpha > p_1, \\ \text{рівняння (2.16)}, & \text{якщо } \alpha \leq p_1, \end{cases} \quad (2.17)$$

де $\alpha \in [0; 1]$ – рівномірно розподілена випадкова величина. З огляду на те, що дезорієнтація є нечастим явищем у природному середовищі, значення p_1 встановлено рівним 0,3 [18]. Інтеграція спіральної стратегії збагачує арсенал пошукових механізмів DBO, підвищує ймовірність знаходження глобального оптимуму та зменшує ризик застрягання у локальних мінімумах.

2.4.3 Стратегія динамічної популяції

У природному середовищі гнойові жуки демонструють колективну поведінку: при знаходженні їжі вони виділяють феромони для залучення інших особин. Зі збільшенням обсягу ресурсів у місці гніздування дедалі більше жуків переходять до стратегії крадіжки замість активного пошуку [18]. Для відтворення цієї поведінки у DPDBO запропоновано стратегію динамічної популяції, що адаптивно перерозподіляє чисельність жуків-котил та крадіїв протягом оптимізаційного процесу.

Сумарна частка жуків-котил та крадіїв фіксується на рівні 55% від загальної популяції, але їхнє індивідуальне співвідношення змінюється від ітерації до ітерації за нелінійним законом [18]:

$$\begin{cases} p_S = p_{TS} - p_T, \\ p_{TS} = \text{pop} \cdot 55\%, \\ p_T = \text{Round}(w), \\ w = w_1 + c - \dfrac{w_1 - w_2}{0{,}75 + e^{-2(2t-M)/M}}, \end{cases} \tag{2.18}$$

де p_S – кількість крадіїв; p_{TS} – сумарна кількість жуків-котил та крадіїв; p_T – кількість жуків-котил; w – адаптивний фактор; t – номер поточної ітерації; M – максимальна кількість ітерацій; c – коригувальний коефіцієнт; $w_1 = p_{TS} \cdot Q_1$ та $w_2 = p_{TS} \cdot Q_2$ – початкове та кінцеве значення кількості жуків-котил, де Q_1 та Q_2 – початкова та кінцева частки жуків-котил відповідно [18].

За рекомендованих значень $Q_1 = 80\%$, $Q_2 = 20\%$ та $c = 1{,}25$ кількість жуків-котил монотонно зменшується від $p_{TS} \cdot Q_1$ до $p_{TS} \cdot Q_2$ з прискоренням у середній частині оптимізаційного процесу та стабілізацією на пізніх ітераціях [18]. Логіка такого перерозподілу полягає в тому, що жуки-котили, завдяки необмеженому діапазону руху та великому кроку, забезпечують глобальну розвідку на ранніх ітераціях, тоді як крадії, що зосереджують пошук навколо глобального оптимуму X^b, посилюють локальну експлуатацію на пізніх ітераціях [18].

2.4.4 Результати тестування DPDBO

Ефективність DPDBO верифіковано на наборах CEC2017 (29 функція, розмірність 10, 1000 ітерацій) та CEC2022 (12 функцій, розмірність 20, 500 ітерацій) у порівнянні з DBO, SSA, HHO, WOA, GA та SCA. Кожен експеримент виконувався 50 разів незалежно з розміром

популяції $N = 30$ [18]. Узагальнені результати за тестом Фрідмана наведено у табл. 2.5. На наборі CEC2017 DPDBO продемонстрував

Таблиця 2.5 – Ранг Фрідмана DPDBO та порівнюваних алгоритмів на наборах CEC2017 та CEC2022 [18]

Набір	**DPDBO**	**DBO**	**SSA**	**HHO**	**WOA**	**GA**	**SCA**
CEC2017	**1,414**	2,724	3,259	4,690	5,207	6,448	4,259
CEC2022	**1,167**	4,667	6,417	2,750	3,833	6,250	2,917

найкращі результати на 21 з 29 тестових функцій (із урахуванням статистики +/ = /-) та отримав найнижчий ранг Фрідмана 1,414. Примітно, що стандартне відхилення результатів DPDBO є суттєво нижчим, ніж у конкурентів, на більшості тестових функцій, що засвідчує вищу стабільність алгоритму [18].

На наборі CEC2022 DPDBO також посів перше місце (ранг 1,167), досягнувши оптимальних результатів на 11 з 12 функцій. Варто зауважити, що базовий DBO на цьому наборі показав значно слабші результати (ранг 4,667), поступившись HHO, WOA та SCA, тоді як DPDBO завдяки трьом стратегіям вдосконалення суттєво покращив ефективність порівняно з базовою версією [18].

Для оцінки індивідуального внеску кожної стратегії проведено абляційне дослідження на 6 функціях CEC2005, де DBO1 – DBO з PWLCM-ініціалізацією, DBO2 – DBO з дезорієнтаційною поведінкою, DBO3 – DBO з динамічною популяцією. Результати засвідчили, що кожна стратегія окремо покращує базовий DBO, а їхнє комбіноване застосування у DPDBO дає найвищу точність та найшвидшу збіжність [18].

Окрім бенчмаркових функцій, DPDBO верифіковано на інженерних задачах оптимізації (проєктування пружини розтягу/стиску, проєктування зварної балки, оптимізація покриття сенсорної мережі), де алгоритм стабільно перевершив базовий DBO та інші метаевристики за найкращими та середніми значеннями цільових функцій [18]. Ці

результати обґрунтовують вибір DPDBO як одного з алгоритмів для чисельних експериментів у третьому розділі даної роботи.

2.5 Алгоритм рою тунікатів та його модифікації

Алгоритм рою тунікатів (Tunicate Swarm Algorithm, TSA) – метаевристичний оптимізаційний алгоритм ройового інтелекту, запропонований Каур та ін. у 2020 році [20]. Алгоритм моделює два ключові аспекти поведінки морських тунікатів: реактивний рух, що забезпечує переміщення окремих особин, та колективну ройову поведінку, що координує дії популяції під час пошуку їжі.

2.5.1 Біологічне підґрунтя алгоритму

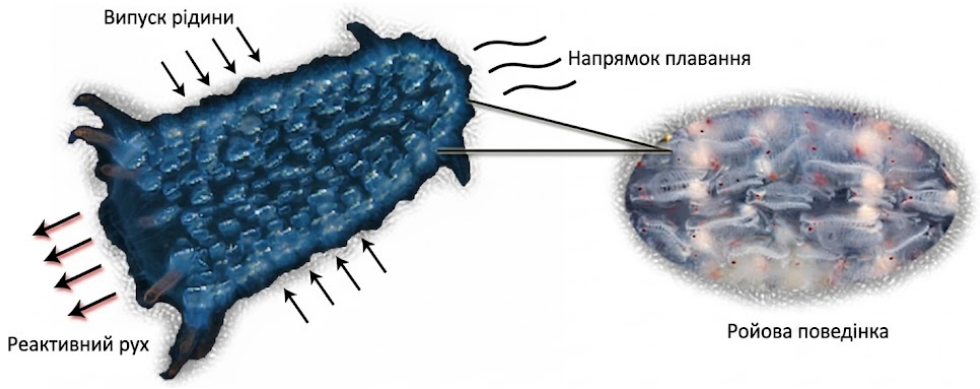

Рисунок 2.1 – Ройова поведінка туніката в океані [20]

Тунікати – підтип хордових тварин, що мешкають у глибинах океану та здатні випромінювати м'яке синьо-зелене біолюмінесцентне світло. Кожна особина поглинає воду з навколишнього середовища через оральний сифон та виштовхує її через атріальний сифон, створюючи реактивну тягу. Цей унікальний механізм забезпечує тунікатам рухливість, достатню для вертикальної міграції у водному стовпі.

Незважаючи на те, що розміри окремих тунікатів становлять лише

кілька міліметрів, вони об'єднуються спільною желатиновою оболонкою (туніка), що дозволяє їм демонструвати координовану ройову поведінку. Під час пошуку їжі тунікати утворюють ланцюгові структури, де лідер (перша особина ланцюга) рухається у напрямку джерела їжі, а послідовники оновлюють свої позиції відносно лідера та сусідніх особин. Саме ця двофазна поведінка – індивідуальний реактивний рух та колективна координація – формує математичну основу алгоритму TSA [20].

2.5.2 Математична модель алгоритму

Розглянемо задачу мінімізації цільової функції $f(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^d$, де d – розмірність простору пошуку. Нехай популяція складається з N тунікатів. Позиція i-го тунікату на ітерації t позначається $\vec{P}_p^{(i)}(t) \in \mathbb{R}^d$, а найкраще знайдене рішення (джерело їжі) – $\vec{FS}$. Математична модель TSA складається з двох фаз: фази реактивного руху та фази ройового інтелекту [20].

Фаза реактивного руху. Ця фаза моделює індивідуальне переміщення тунікатів та включає три послідовні механізми: уникнення конфліктів між особинами, рух у напрямку найкращого сусіда та рух у напрямку найкращого тунікату.

Уникнення конфліктів між особинами. Для запобігання зіткненням між тунікатами під час пошуку обчислюється вектор $\vec{A}$, що визначає нове положення кожної особини:

$$\vec{A} = \frac{\vec{G}}{\vec{M}}, \tag{2.19}$$

де $\vec{G}$ – вектор, що моделює сили гравітації та потоку води; $\vec{M}$ – вектор соціальних сил між тунікатами. Складові обчислюються як:

$$\vec{G} = \vec{c}_2 + \vec{c}_3 - \vec{F}, \qquad \vec{F} = 2 \cdot c_1, \tag{2.20}$$

де $c_1 \in [0,1]$ – випадкове число з рівномірного розподілу; $\vec{c}_2$ та $\vec{c}_3$ – випадкові вектори з елементами у діапазоні $[0,1]$; $\vec{F}$ – адвекція водного потоку, що моделює зовнішній вплив течії на тунікатів. Зауважимо, що коли $c_1 > 0{,}5$, значення $\vec{F} > 1$, що сприяє розвідці; при $c_1 < 0{,}5$ значення $\vec{F} < 1$, що посилює експлуатацію.

Вектор соціальних сил $\vec{M}$ визначає швидкість соціальної взаємодії між особинами:

$$\vec{M} = \lfloor P_{\min} + c_1 \cdot (P_{\max} - P_{\min}) \rfloor, \tag{2.21}$$

де $P_{\min}$ та $P_{\max}$ – початкова та кінцева швидкості соціальної взаємодії (рекомендовані значення: $P_{\min} = 1$, $P_{\max} = 4$).

Рух у напрямку найкращого сусіда. Після усунення конфліктів тунікат рухається у напрямку найкращої знайденої позиції (джерела їжі). Відстань між тунікатом та джерелом їжі обчислюється за формулою:

$$\vec{PD} = \left| \vec{FS} - \text{rand} \cdot \vec{P}_p(\mathbf{x}) \right|, \tag{2.22}$$

де $\vec{PD}$ – відстань між джерелом їжі та тунікатом; $\vec{FS}$ – позиція джерела їжі (поточний глобальний оптимум); $\vec{P}_p(\mathbf{x})$ – поточна позиція тунікату; $\text{rand} \in [0,1]$ – випадкове число.

Рух у напрямку найкращого тунікату. Після визначення відстані $\vec{PD}$ тунікат оновлює свою позицію відносно найкращого тунікату за правилом:

$$\vec{P}_p(\mathbf{x}') = \begin{cases} \vec{FS} + \vec{A} \cdot \vec{PD}, & \text{якщо rand} \geq 0{,}5, \\ \vec{FS} - \vec{A} \cdot \vec{PD}, & \text{якщо rand} < 0{,}5, \end{cases} \tag{2.23}$$

де $\vec{P}_p(\mathbf{x}')$ – оновлена позиція тунікату. Стохастичний вибір між додаванням та відніманням забезпечує розвідку простору пошуку з обох боків від поточного оптимуму.

Фаза ройового інтелекту. Ця фаза моделює колективну поведінку ланцюга тунікатів. Алгоритм зберігає два найкращих знайдених розв'язки

та оновлює позиції інших агентів відповідно до позицій цих лідерів. Позиції послідовників оновлюються за формулою:

$$\vec{P}_p(\mathbf{x}+1) = \frac{\vec{P}_p(\mathbf{x}) + \vec{P}_p(\mathbf{x}+1)}{2+c_1}, \tag{2.24}$$

де $\vec{P}_p(\mathbf{x})$ – поточна позиція тунікату; $\vec{P}_p(\mathbf{x}+1)$ – позиція, розрахована за формулою (2.23); $c_1 \in [0,1]$ – випадкове число. Ділення на $(2+c_1)$ замість простого усереднення забезпечує стохастичну регуляризацію оновлення, що запобігає надмірному зближенню особин та підтримує різноманітність популяції. Кінцеве положення кожного тунікату випадковим чином розташовується у межах циліндричної або конічної області, визначеної позицією лідера ланцюга. Узагальнену процедуру TSA подано в алгоритмі 2.3:

Алгоритм 2.3 Алгоритм рою тунікатів (TSA)

Вхід: Популяція тунікатів $\vec{P}_p$
Вихід: Оптимальне значення пристосованості $\vec{FS}$

procedure TSA
  Ініціалізувати параметри $\vec{A}$, $\vec{G}$, $\vec{F}$, $\vec{M}$ та $Max_{iterations}$
  $P_{\min} \leftarrow 1$; $P_{\max} \leftarrow 4$
  $Swarm \leftarrow 0$
  while $x < Max_{iterations}$ **do**
    for $i \leftarrow 1$ **to** 2 **do** ▷ Цикл обчислення ройової поведінки
      $\vec{FS} \leftarrow$ обчислити пристосованість кожного агента $\vec{P}_p$ ▷ Реактивний рух
      $c_1, c_2, c_3, r \leftarrow$ Rand() ▷ Випадкові числа з $[0,1]$
      $\vec{M} \leftarrow \lfloor P_{\min} + c_1 \times P_{\max} - P_{\min} \rfloor$
      $\vec{F} \leftarrow 2 \times c_1$
      $\vec{G} \leftarrow c_2 + c_3 - \vec{F}$
      $\vec{A} \leftarrow \vec{G} \,/\, \vec{M}$ ▷ Рівняння (2.19)
      $\vec{PD} \leftarrow |\vec{FS} - r \times \vec{P}_p(x)|$ ▷ Рівняння (2.22) ▷ Ройова поведінка
      if $r \leq 0{,}5$ **then**
        $Swarm \leftarrow Swarm + \vec{FS} + \vec{A} \times \vec{PD}$
      else
        $Swarm \leftarrow Swarm + \vec{FS} - \vec{A} \times \vec{PD}$
      end if
    end for
    $\vec{P}_p(x) \leftarrow Swarm \,/\, (2 + c_1)$ ▷ Рівняння (2.24)
    $Swarm \leftarrow 0$
    Оновити параметри $\vec{A}$, $\vec{G}$, $\vec{F}$, $\vec{M}$
    $x \leftarrow x + 1$
  end while
  return $\vec{FS}$
end procedure

2.5.3 Відомі обмеження базового TSA

Незважаючи на перелічені переваги, базовий TSA має ряд суттєвих обмежень, ідентифікованих у літературі [47]:

Передчасна збіжність. Механізм оновлення позицій (2.23)–(2.24) спрямовує всіх агентів до поточного найкращого рішення $\vec{FS}$, що на мультимодальних ландшафтах з великою кількістю локальних мінімумів може призводити до передчасного застрягання популяції в околі субоптимального розв'язку.

Недостатня розвідка. Оновлення позицій (2.24) за фазою ройового інтелекту здійснюється як зважене середнє між поточною та новою позицією, що обмежує крок переміщення агентів і може уповільнювати дослідження віддалених ділянок простору пошуку.

Низька різноманітність популяції. Відсутність механізмів мутації або хаотичної ініціалізації може спричиняти швидку втрату різноманітності рішень, особливо на пізніх ітераціях. Ці обмеження мотивували розробку численних модифікацій TSA, розглянутих далі.

2.5.4 Модифікації алгоритму TSA

З моменту публікації у 2020 році оригінальна стаття [20] отримала понад 1000 цитувань, а кількість публікацій, присвячених TSA та його варіантам, стрімко зростає [47]. Модифікації TSA можна систематизувати за типом вдосконалення: (1) модифіковані версії, що змінюють окремі оператори або параметри алгоритму; (2) гібридизовані версії, що поєднують компоненти TSA з іншими метаевристиками [47]. Розглянемо найбільш значущі стратегії модифікації.

TSA на основі навчання протилежних рішень. Навчання на основі протилежних рішень (англ. Opposition-Based Learning, OBL), запроваджене у [37], є технікою, що підвищує оптимізаційну здатність

алгоритму шляхом дослідження простору пошуку у напрямку, протилежному до поточного рішення. Для кожного рішення $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ у межах $[lb_j, ub_j]$ протилежне рішення обчислюється як [37]:

$$\hat{x}_j = lb_j + ub_j - x_j, \quad j = 1, 2, \ldots, d, \tag{2.25}$$

де lb_j та ub_j – нижня та верхня межі j-го виміру відповідно. Порівнюючи значення цільової функції для $\mathbf{x}$ та $\hat{\mathbf{x}}$, алгоритм обирає кращий з двох варіантів, що збільшує різноманітність популяції та підвищує ймовірність виходу з локальних оптимумів.

Численні дослідження підтвердили ефективність інтеграції OBL з TSA [47]. Зокрема, Хуссейн та ін. [17] запропонували варіант TSA-LEO, посилений оператором локального виходу (англ. Local Escaping Operator, LEO). LEO генерує пробні розв'язки на основі лінійної комбінації випадкових агентів популяції та поточного найкращого рішення, що запобігає стагнації пошуку та прискорює збіжність. Валідація на тестовому наборі CEC'2017 та задачах сегментації зображень засвідчила, що TSA-LEO суттєво покращує якість розв'язків та швидкість збіжності порівняно із іншими метаевристичними алгоритмами [17].

TSA з блуканням Леві. Блукання Леві (Lévy flight) – це випадкове блукання, довжини кроків якого підпорядковуються степеневому розподілу з важким хвостом. Асимптотична апроксимація густини розподілу при $s \to \infty$ має вигляд [44]:

$$L(s, \beta) \sim \frac{\alpha\, \beta\, \Gamma(\beta) \sin(\pi\beta/2)}{\pi\, |s|^{1+\beta}}, \quad 0 < \beta \leq 2, \tag{2.26}$$

де s – довжина кроку; β – індекс стабільності розподілу ($\beta = 2$ відповідає броунівському руху); $\alpha > 0$ – масштабний параметр; $\Gamma(\cdot)$ – гамма-функція Ейлера. На практиці крок Леві обчислюється за алгоритмом Мантеньї [44]:

$$s = \frac{u}{|v|^{1/\beta}}, \quad u \sim \mathcal{N}(0, \sigma_u^2),\, v \sim \mathcal{N}(0, 1), \tag{2.27}$$

де $\sigma_u = \left\{\Gamma(1+\beta)\sin(\pi\beta/2) \,/\, [\Gamma((1+\beta)/2)\,\beta\,2^{(\beta-1)/2}]\right\}^{1/\beta}$. Завдяки рідкісним, але значно довшим крокам, блукання Леві дозволяє алгоритму здійснювати глобальний пошук та ефективно уникати локальних оптимумів. Позиція тунікату з урахуванням блукання Леві оновлюється за модифікованою формулою [10]:

$$\vec{P}_p^{\,\text{new}}(\mathbf{x}) = \vec{P}_p(\mathbf{x}) + \alpha_0 \cdot \frac{u}{|v|^{1/\beta}} \cdot \left(\vec{P}_p(\mathbf{x}) - \vec{FS}\right), \tag{2.28}$$

де $\alpha_0 = 0{,}01$ – масштабний множник кроку; u, v визначені у (2.27). Фетух та Елсайед [10] запропонували покращений TSA (ITSA) з розподілом Леві для оптимізації розподільних мереж, де інтеграція блукання Леві дозволила подолати ризик стагнації пошуку.

Хаотичний TSA. Хаотичні відображення – це математичні моделі, які не використовують справжню випадковість, але здатні генерувати складні, обмежені певним діапазоном і неповторювані траєкторії, що поводяться як хаос. [28]. Застосування хаотичних послідовностей замість рівномірно розподілених випадкових чисел у параметрах TSA (зокрема, у c_1, $\vec{c}_2$, $\vec{c}_3$) дозволяє підвищити ергодичність пошуку та поліпшити контроль різноманітності популяції. Найбільш поширеним є логістичне відображення:

$$x_{n+1} = \mu \cdot x_n \cdot (1 - x_n), \tag{2.29}$$

де $x_n \in (0, 1)$ – значення хаотичної послідовності на n-му кроці; μ – керуючий параметр (при $\mu = 4$ система перебуває у повністю хаотичному режимі). Отримана послідовність $\{x_n\}$ підставляється замість c_1, c_2, c_3 у формулах (2.20)–(2.21), що забезпечує більш рівномірне покриття простору пошуку порівняно з псевдовипадковими числами. Гупта та ін. [14] запропонували хаотичний TSA (CTSA) із адаптивною ваговою моделлю для балансування розвідки та експлуатації при ідентифікації параметрів сонячних фотоелектричних елементів. Цуй та ін. [5] розробили варіант TLTSA, що комбінує дев'ять хаотичних відображень зі стратегією блукання Леві; порівняльний аналіз на бенчмаркових функціях та

інженерних задачах засвідчив посилені здатності TLTSA до розвідки та експлуатації порівняно з базовим TSA та іншими метаевристиками.

Багатостратегічні модифікації. Найбільш ефективні модифікації поєднують кілька стратегій вдосконалення одночасно. Ван та ін. [38] запропонували варіант SLOTSA, що інтегрує OBL, блукання Леві та позитивний косинус-оператор для подолання типових проблем TSA: низької точності, повільної збіжності та схильності до локальних екстремумів; порівняльні експерименти на 10 бенчмаркових функціях та двох задачах промислового проєктування підтвердили широку застосовність підходу. Лі та ін. [24] розробили покращений TSA (ITSA) для задачі динамічного економіко-екологічного розподілу навантаження у енергосистемах, що поєднує тент-відображення для ініціалізації популяції, оптимізатор сірого вовка для генерування вектора глобального пошуку та блукання Леві для розширення діапазону пошуку.

2.5.5 Гібридизовані версії TSA

Гібридизація передбачає поєднання компонентів TSA з операторами або стратегіями інших метаевристичних алгоритмів для посилення пошукових здатностей. Систематичний огляд [47] фіксує понад 40 різних гібридних варіантів TSA, поєднаних з такими алгоритмами, як GWO, PSO, WOA, GA та інші. Чоухан та Джайн [3] запропонували гібридний алгоритм TSGWO, що поєднує TSA з оптимізатором сірого вовка для задачі мультимаршрутної маршрутизації у бездротових сенсорних мережах Інтернету речей; TSGWO оптимізує маршрутизацію з урахуванням затримки, енергії та відстані та перевершує конкурентні методи за залишковою енергією, часом життя каналу та пропускною здатністю. Хаджехзаде та ін. [22] розробили послідовний гібрид CTSA-PS, що комбінує глобальну пошукову здатність TSA із локальним пошуком методу шаблонного пошуку для аналізу сейсмічної стійкості схилів; чисельні дослідження засвідчили, що CTSA-PS забезпечує кращі

оптимальні розв'язки порівняно зі стандартним TSA та іншими методами.

2.6 Алгоритм оптимізації птаха-секретаря

Алгоритм оптимізації птаха-секретаря (Secretary Bird Optimization Algorithm, SBOA) – популяційний метаевристичний алгоритм, запропонований у [11] 2024 року. Алгоритм моделює дві ключові стратегії виживання птаха-секретаря у природному середовищі: полювання на змій та втечу від хижаків. Стратегія полювання формує фазу розвідки, а стратегія ухилення – фазу експлуатації. Завдяки поєднанню диференціальної еволюції, броунівського руху та блукань Леві у фазі розвідки, а також динамічних збурень у фазі експлуатації SBOA забезпечує ефективний баланс між глобальним та локальним пошуком.

2.6.1 Біологічне підґрунтя алгоритму

Птах-секретар – великий хижий птах родини, ендемічний для саван Субсахарської Африки. Він досягає зросту до 1,5 м і відзначається характерними довгими ногами та чубом з пір'я на потилиці. На відміну від більшості хижих птахів, секретар полює переважно наземним способом, пересуваючи за день відстані 20–30 км, за що отримав прізвисько «маршировальний орел».

Рисунок 2.2 – Птах-секретар

В основу алгоритму покладено дві біологічні моделі поведінки, що характеризують стратегію виживання птаха-секретаря.

Мисливська стратегія. Полювання на змій відбувається у три фази. У першій фазі (пошук здобичі) птах використовує гострий зір для

виявлення змій, прихованих у високій траві. Довгі ноги та шия дозволяють підтримувати безпечну дистанцію від потенційно отруйної здобичі. У другій фазі (виснаження здобичі) секретар, на відміну від інших хижих птахів, не атакує одразу, а маневрує навколо змії, провокуючи та виснажуючи її. Товсті кератинові луски на ногах забезпечують захист від укусів. У третій фазі (атака) секретар завдає потужних ударів ногами по голові змії гострими кігтями, а великих змій може піднімати у повітря і скидати на тверду поверхню.

Стратегія ухилення. Природними ворогами птаха-секретаря є великі хижаки – орли, яструби, лисиці та шакали. При зустрічі із загрозою секретар використовує одну з двох стратегій. Перша стратегія – маскування: птах використовує кольори та структуру навколишнього середовища для камуфляжу, щоб ускладнити виявлення хижаком. Друга стратегія – втеча бігом або польотом: завдяки довгим ногам та здатності до швидкого польоту секретар може оперативно покинути небезпечну зону.

Відповідність між біологічною поведінкою та алгоритмічними етапами SBOA є такою: підготовка до полювання відповідає ініціалізації популяції, три фази полювання – трьом етапам розвідки, а дві стратегії ухилення – двом стратегіям експлуатації [11].

2.6.2 Математична модель алгоритму

Розглянемо задачу мінімізації цільової функції $f(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^D$, де D – розмірність простору пошуку. Нехай популяція складається з N агентів (птахів-секретарів). Позиція i-го агента є D-вимірним вектором $\mathbf{X}_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,D})$.

Ініціалізація. Початкові позиції агентів генеруються рівномірно в межах допустимої області [11]:

$$x_{i,j} = lb_j + r \cdot (ub_j - lb_j), \quad i = 1,\ldots,N,\ j = 1,\ldots,D, \tag{2.30}$$

де lb_j та ub_j – нижня й верхня межі j-го виміру відповідно; $r \sim \mathcal{U}(0,1)$ – рівномірно розподілена випадкова величина. Матриця позицій усієї популяції має розмір $N \times D$, а вектор значень цільової функції записується як [11]:

$$\mathbf{F} = \begin{pmatrix} F_1 \\ F_2 \\ \vdots \\ F_N \end{pmatrix} = \begin{pmatrix} f(\mathbf{X}_1) \\ f(\mathbf{X}_2) \\ \vdots \\ f(\mathbf{X}_N) \end{pmatrix}. \tag{2.31}$$

На кожній ітерації визначається найкращий розв'язок $\mathbf{x}_{\text{best}}$ – позиція агента з найменшим (для задач мінімізації) значенням цільової функції. Оновлення позицій відбувається у два послідовні етапи: розвідку (полювання) та експлуатацію (ухилення).

Фаза розвідки: мисливська стратегія. Фаза розвідки моделює три стадії полювання на змію і поділяється на три рівні часові інтервали ітераційного процесу: $t < \frac{1}{3}T$, $\frac{1}{3}T \leq t < \frac{2}{3}T$ та $\frac{2}{3}T \leq t < T$, де t – номер поточної ітерації; T – максимальна кількість ітерацій [11].

Етап 1: пошук здобичі ($t < \frac{1}{3}T$). На початкових ітераціях, коли розвідка є пріоритетною, застосовується стратегія диференціальної еволюції (differential evolution, DE). Диференціальна еволюція – метод оптимізації, що використовує різниці між позиціями випадково обраних агентів для генерування нових розв'язків, чим забезпечується висока різноманітність популяції та широке покриття простору пошуку. Позиції оновлюються за формулою [11]:

$$x_{i,j}^{\text{new}} = x_{i,j} + (x_{\text{rand}_1,j} - x_{\text{rand}_2,j}) \cdot R_1, \tag{2.32}$$

де x_{rand_1} та x_{rand_2} – позиції двох випадково обраних агентів із поточної популяції ($\text{rand}_1 \neq \text{rand}_2 \neq i$); R_1 – випадковий вектор розміру $1 \times D$ з елементами, рівномірно розподіленими на інтервалі $[0, 1]$.

Етап 2: виснаження здобичі ($\frac{1}{3}T \leq t < \frac{2}{3}T$). Моделює маневрування секретаря навколо змії із використанням броунівського

руху. Броунівський рух – стохастичний процес, що описує випадкові переміщення частинки внаслідок хаотичних зіткнень з молекулами середовища; у контексті оптимізації він забезпечує локальне дослідження околу поточної позиції з випадковими збуреннями різної амплітуди. Позиція оновлюється за правилом [11]:

$$x_{i,j}^{\text{new}} = x_{\text{best},j} + \exp\left(\left(\frac{t}{T}\right)^4\right) \cdot (R_B - 0{,}5) \cdot (x_{\text{best},j} - x_{i,j}), \tag{2.33}$$

де x_{best} – позиція найкращого розв'язку поточної ітерації; $R_B = \text{randn}(1,D)$ – вектор броунівського руху, що є випадковим масивом розміру $1 \times D$, згенерованим зі стандартного нормального розподілу. Множник $\exp\big((t/T)^4\big)$ монотонно зростає з ітераціями, поступово збільшуючи амплітуду збурень: на ранніх ітераціях він близький до одиниці, що відповідає обережному маневруванню, тоді як на пізніших ітераціях цього етапу – посилює інтенсивність дослідження околу $\mathbf{x}_{\text{best}}$ [11].

Етап 3: атака здобичі ($\frac{2}{3}T \le t < T$). Моделює фінальний удар секретаря за допомогою блукань Леві (Levy flight). Блукання Леві – різновид випадкового руху, у якому довжина кроку має степеневий (важкохвостий) розподіл; це породжує чергування великої кількості коротких кроків з рідкісними довгими стрибками. Такий механізм дозволяє алгоритму, з одного боку, виконувати точний локальний пошук (короткі кроки), а з іншого – здійснювати «стрибки» у віддалені ділянки простору (довгі кроки), що сприяє уникненню локальних пасток. Оновлення позиції має вигляд [11]:

$$x_{i,j}^{\text{new}} = x_{\text{best},j} + \left(1 - \frac{t}{T}\right)^{\left(2 \cdot \frac{t}{T}\right)} \cdot x_{i,j} \cdot R_L, \tag{2.34}$$

де $R_L = 0{,}5 \cdot \text{Levy}(D)$ – зважений вектор блукань Леві. Функція розподілу

Леві обчислюється як [11]:

$$\text{Levy}(D) = s \cdot \frac{u \cdot \sigma}{|v|^{1/\eta}}, \tag{2.35}$$

де $s = 0{,}01$ – масштабна стала; u та v – незалежні випадкові величини з розподілом $\mathcal{N}(0,1)$; $\eta = 1{,}5$ – індекс стійкості розподілу Леві; параметр σ визначається за формулою [11]:

$$\sigma = \left(\frac{\Gamma(1+\eta) \cdot \sin\left(\frac{\pi\eta}{2}\right)}{\Gamma\left(\frac{1+\eta}{2}\right) \cdot \eta \cdot 2^{(\eta-1)/2}} \right)^{1/\eta}, \tag{2.36}$$

де $\Gamma(\cdot)$ – гамма-функція Ейлера. Нелінійний множник $(1 - t/T)^{2t/T}$ у формулі (2.34) виконує роль адаптивного збурення: на початку третього інтервалу він є значним, що підтримує глобальну розвідку, а наприкінці ітераційного процесу наближається до нуля, забезпечуючи точне уточнення оптимуму [11].

Для кожного з трьох етапів нова позиція приймається лише за умови покращення значення цільової функції:

$$\mathbf{X}_i \leftarrow \begin{cases} \mathbf{X}_i^{\text{new}}, & \text{якщо } f(\mathbf{X}_i^{\text{new}}) < F_i, \\ \mathbf{X}_i, & \text{інакше.} \end{cases} \tag{2.37}$$

Фаза експлуатації: стратегія ухилення. Після завершення фази розвідки (для кожної ітерації) кожен агент оновлює свою позицію за однією з двох стратегій ухилення від хижаків, обраною з рівною ймовірністю. У базовому варіанті SBOA вважається, що з імовірністю $r = 0{,}5$ агент обирає стратегію маскування, а інакше – стратегію втечі [11].

Стратегія C_1: маскування (якщо rand $< r_i$). Агент рухається в напрямку найкращого розв'язку з динамічним збуренням:

$$x_{i,j}^{\text{new}} = x_{\text{best},j} + (2 \cdot R_B - 1) \cdot \left(1 - \frac{t}{T}\right)^2 \cdot x_{i,j}, \tag{2.38}$$

де $R_B = \text{randn}(1,D)$ – вектор броунівського руху; множник $(1 - t/T)^2$

зменшується з ітераціями, поступово звужуючи амплітуду збурень, що забезпечує перехід від широкого обстеження околу $\mathbf{x}_{\text{best}}$ до його точного уточнення [11].

Стратегія C_2: втеча (якщо rand $\geq r_i$). Агент переміщується у напрямку різниці між випадковим агентом та поточною позицією:

$$x_{i,j}^{\text{new}} = x_{i,j} + R_2 \cdot (x_{\text{rand},j} - K \cdot x_{i,j}), \tag{2.39}$$

де $R_2 = \text{randn}(1,D)$ – випадковий вектор з нормального розподілу; x_{rand} – позиція випадково обраного агента поточної популяції; K – цілочисельний коефіцієнт, що випадково набуває значення 1 або 2:

$$K = \text{round}(1 + \text{rand}(0,1)). \tag{2.40}$$

Фізичний зміст стратегії C_2 полягає в імітації швидкої зміни положення птаха при втечі. Параметр K контролює ступінь «відриву» від поточної позиції: при $K = 1$ агент переміщується у напрямку від $x_{i,j}$ до $x_{\text{rand},j}$, а при $K = 2$ – у напрямку, що є дзеркальним відображенням його позиції відносно випадкового агента. Це забезпечує стохастичну диверсифікацію навіть на пізніх ітераціях, коли фаза розвідки вже обмежена.

Аналогічно до фази розвідки, нова позиція приймається лише за умови покращення значення цільової функції:

$$\mathbf{X}_i \leftarrow \begin{cases} \mathbf{X}_i^{\text{new}}, & \text{якщо } f(\mathbf{X}_i^{\text{new}}) < F_i, \\ \mathbf{X}_i, & \text{інакше.} \end{cases} \tag{2.41}$$

Ітераційний процес повторюється до досягнення максимальної кількості ітерацій T.

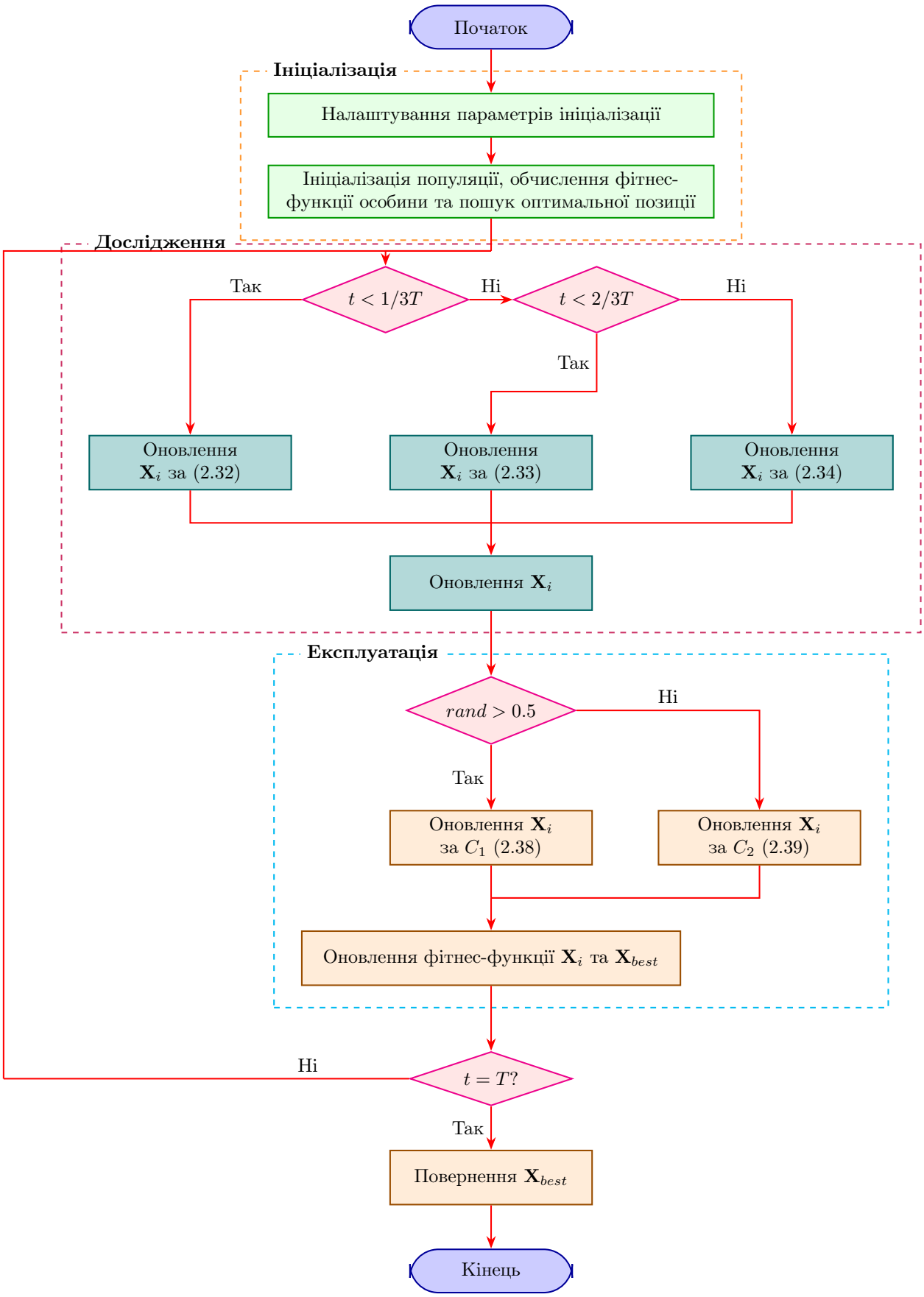

Рисунок 2.3 – Блок-схема алгоритму оптимізації

2.6.3 Результати тестування алгоритму

Ефективність SBOA верифіковано авторами алгоритму на тестових наборах CEC-2017 (29 функцій: унімодальні, мультимодальні, гібридні та композитні) та CEC-2022 (12 функцій) у порівнянні з 15 алгоритмами: LSHADE_cnEpSin, LSHADE_SPACMA, MadDE, DE, GWO, WOA, CPSOGSA, AVOA, SO, GTO, COA, RIME, GJO, DBO та NOA. Параметри експерименту: розмір популяції $N = 30$, максимальна кількість ітерацій $T = 500$, кількість незалежних запусків – 30. Тестування проводилося у розмірностях $D = 30$, 50 та 100 для CEC-2017 та $D = 10$, 20 для CEC-2022 [11].

Узагальнені результати ранжування за непараметричним тестом Фрідмана наведено у табл. 2.6.

Таблиця 2.6 – Середні ранги алгоритмів за тестом Фрідмана на наборах CEC-2017 та CEC-2022 [11]

	CEC-2017			**CEC-2022**	
Алгоритм	$D = 30$	$D = 50$	$D = 100$	$D = 10$	$D = 20$
SBOA	**1,52**	**1,65**	**1,77**	**1,73**	**1,00**
LSHADE_SPACMA	2,52	3,13	4,00	3,73	2,31
LSHADE_cnEpSin	3,03	3,32	4,00	2,31	5,92
RIME	6,35	5,68	5,26	6,69	5,38
GTO	6,71	6,29	5,39	7,15	7,00
DBO	12,55	12,45	12,55	10,31	11,62
WOA	15,10	14,97	14,71	13,92	14,46

SBOA стабільно посідає перше місце за середнім рангом Фрідмана у всіх розмірностях і на обох тестових наборах, суттєво випереджаючи навіть алгоритми-переможці змагань CEC (LSHADE_SPACMA та LSHADE_cnEpSin). Для тестів CEC-2017 із розмірністю $D = 30$ SBOA отримав найкращий середній результат на 22 з 29 функцій, другий найкращий – на 3 функціях, третій – на 4. Зі зростанням розмірності SBOA зберігає лідерство: при $D = 100$ алгоритм отримав найкращі

результати на 20 функціях [11].

Крім бенчмаркінгу на тестових функціях, авторами проведено верифікацію на 12 задачах інженерного проєктування з обмеженнями (оптимізація зварного з'єднання, пружини розтягу/стиску, редуктора швидкості, балки тощо) та задачі тривимірного планування траєкторії безпілотного літального апарата. У всіх 12 інженерних задачах SBOA знайшов розв'язки, що дорівнюють або перевершують результати порівнюваних алгоритмів. У задачі планування траєкторії БПЛА середнє значення функції придатності SBOA виявилося меншим за найкращі значення решти 15 алгоритмів, а стандартне відхилення – найменшим, що засвідчує вищу обчислювальну точність та стабільність [11].

2.6.4 Аналіз балансу розвідки та експлуатації

Баланс між розвідкою та експлуатацією в SBOA забезпечується трьома механізмами [11].

Розвідка простору пошуку реалізується послідовно через три стратегії зростаючої інтенсивності. На ранніх ітераціях ($t < T/3$) диференціальна мутація (2.32) генерує нові розв'язки на основі різниць між випадковими агентами, що забезпечує максимальну різноманітність популяції та широке покриття простору. У середній третині ітерацій ($T/3 \leq t < 2T/3$) броунівський рух (2.33) здійснює локальне дослідження околу поточного найкращого розв'язку $\mathbf{x}_{\text{best}}$, поступово посилюючи інтенсивність збурень через множник $\exp\big((t/T)^4\big)$. На завершальній третині ($t \geq 2T/3$) блукання Леві (2.34) поєднує короткі та довгі стрибки, що дозволяє одночасно уточнювати околиці оптимуму та «вистрибувати» з локальних пасток.

Експлуатація знайдених перспективних ділянок забезпечується фазою ухилення. Стратегія маскування C_1 (2.38) концентрує пошук в околі $\mathbf{x}_{\text{best}}$ із затухаючою амплітудою збурень $(1 - t/T)^2$, що на пізніх ітераціях забезпечує точне уточнення оптимуму. Стратегія втечі C_2 (2.39)

підтримує стохастичну диверсифікацію, переміщуючи агентів у напрямку випадкових членів популяції, що запобігає передчасній збіжності.

Висновки до розділу 2

У другому розділі основну увагу було приділено інструментам пошуку розв'язків – метаевристичним алгоритмам ройового інтелекту. Вони стають незамінними у випадках, коли інженерні задачі мають складний нелінійний характер, а класичні методи не дають результату через неможливість їх застосувати чи ризик застрягання у локальних мінімумах. Детальний розгляд алгоритмів SSA, DBO, TSA та SBOA дозволив зробити кілька важливих висновків:

1) успіх пошуку залежить від того, наскільки вдало алгоритм поєднує глобальну розвідку нових ділянок простору та локальне уточнення знайдених перспективних зон;

2) використання гібридних стратегій та динамічної адаптації параметрів дозволяє значно покращити збіжність і стабільність базових методів пошуку;

3) оскільки універсального алгоритму для всіх типів задач не існує, вибір найкращого інструменту завжди вимагає порівняльного аналізу в межах конкретної предметної області.

Цей теоретичний аналіз створює необхідний фундамент для подальшого практичного застосування та порівняння обраних алгоритмів у межах нашого дослідження.

3 РЕАЛІЗАЦІЯ ТА ПОРІВНЯЛЬНИЙ АНАЛІЗ АЛГОРИТМІВ НА ЗАДАЧАХ ІНЖЕНЕРНОГО ПРОЄКТУВАННЯ

У третьому розділі здійснено практичне порівняння алгоритмів оптимізації, досліджених у другому розділі, на наборі інженерних задач з обмеженнями. Спочатку обґрунтовано методику чисельного експерименту: введено два еталонних алгоритми (генетичний алгоритм та оптимізацію роєм частинок), описано стратегію вибору обчислювального бюджету, метод штрафних функцій для врахування обмежень, а також статистичний апарат порівняння. Далі наведено математичні постановки чотирьох тестових задач зростаючої розмірності (від 4 до 11 змінних). Для кожної задачі подано результати 30 незалежних запусків семи алгоритмів, криві збіжності та статистичні тести. Розділ завершується агрегованим порівняльним аналізом.

3.1 Методика чисельного експерименту

Для коректного оцінювання ефективності сучасних ройових алгоритмів (SSA, DBO, DPDBO, TSA, SBOA), розглянутих у другому розділі, необхідно порівняти їх із добре відомими класичними метаевристиками. У ролі еталонних обрано *генетичний алгоритм* (GA) та *оптимізацію роєм частинок* (PSO) – два найпоширеніші популяційні стохастичні оптимізатори, що є стандартними базовими методами у більшості порівняльних досліджень [44].

Генетичний алгоритм (англ. Genetic Algorithm, GA) – клас еволюційних метаевристик, запропонований Голландом [16] та популяризований Ґолдберґом [12]. GA моделює механізми природного добору: популяція розв'язків-«хромосом» еволюціонує шляхом

ітеративного застосування операторів *селекції*, *схрещування* та *мутації*.

У даній роботі застосовано дійсночислову реалізацію GA з імітованим бінарним схрещуванням (англ. Simulated Binary Crossover, SBX) [7] та поліноміальною мутацією [8], які є стандартними операторами для задач неперервної оптимізації.

Селекція. Батьки обираються турнірним відбором з розміром турніру 2: два випадкові індивіди порівнюються за значенням цільової функції, і кращий проходить до наступного етапу.

Схрещування SBX. Для пари батьківських хромосом $\mathbf{p}_1$ та $\mathbf{p}_2$ нащадки генеруються покомпонентно. Для j-ої координати з імовірністю p_c обчислюється коефіцієнт розподілу [7]:

$$\beta_q = \begin{cases} (2u)^{1/(\eta_c+1)}, & \text{якщо } u \le 0{,}5, \\ \left(\dfrac{1}{2(1-u)}\right)^{1/(\eta_c+1)}, & \text{якщо } u > 0{,}5, \end{cases} \tag{3.1}$$

де $u \sim U(0,1)$ – рівномірно розподілена випадкова величина; η_c – індекс розподілу, що контролює інтенсивність схрещування (більші значення генерують нащадків ближче до батьків). Координати нащадків обчислюються як:

$$c_{1,j} = \frac{1}{2}\big[(1+\beta_q)\,p_{1,j}+(1-\beta_q)\,p_{2,j}\big], \quad c_{2,j} = \frac{1}{2}\big[(1-\beta_q)\,p_{1,j}+(1+\beta_q)\,p_{2,j}\big]. \tag{3.2}$$

Поліноміальна мутація. Для j-ої координати нащадка з імовірністю $p_m = 1/D$ (де D – розмірність задачі) обчислюється збурення [8]:

$$\delta_q = \begin{cases} (2u)^{1/(\eta_m+1)} - 1, & \text{якщо } u < 0{,}5, \\ 1 - \big(2(1-u)\big)^{1/(\eta_m+1)}, & \text{якщо } u \ge 0{,}5, \end{cases} \tag{3.3}$$

де η_m – індекс розподілу мутації. Мутований ген отримує значення $c'_j = c_j + \delta_q \cdot (x_j^{\max} - x_j^{\min})$.

Заміна. Застосовується елістична стратегія $(\mu + \lambda)$: об'єднана

популяція батьків і нащадків сортується за значенням цільової функції, і N найкращих індивідів переходять у наступне покоління.

У всіх експериментах прийнято: імовірність схрещування $p_c = 0{,}9$; індекси розподілу $\eta_c = \eta_m = 20$; імовірність мутації $p_m = 1/D$.

Оптимізація роєм частинок (англ. Particle Swarm Optimization, PSO) запропонована Кеннеді та Еберхартом [21] за аналогією з колективною поведінкою зграй птахів та косяків риб. Кожна частинка i має позицію $\mathbf{x}_i$ та швидкість $\mathbf{v}_i$, що оновлюються на кожній ітерації t за формулами [35]:

$$\mathbf{v}_i^{t+1} = w\,\mathbf{v}_i^t + c_1\,\mathbf{r}_1 \circ (\mathbf{p}_i - \mathbf{x}_i^t) + c_2\,\mathbf{r}_2 \circ (\mathbf{g} - \mathbf{x}_i^t), \tag{3.4}$$

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^{t+1}, \tag{3.5}$$

де w – коефіцієнт інерції; c_1, c_2 – когнітивний та соціальний коефіцієнти прискорення; $\mathbf{r}_1, \mathbf{r}_2 \sim U(0,1)^D$ – вектори випадкових чисел; $\circ$ – покомпонентний добуток Адамара; $\mathbf{p}_i$ – найкраща позиція, знайдена i-ою частинкою за весь час пошуку; $\mathbf{g}$ – глобально найкраща позиція серед усіх частинок рою.

Коефіцієнт інерції w лінійно зменшується протягом ітерацій [35]:

$$w(t) = w_{\max} - \frac{(w_{\max} - w_{\min})\,t}{T_{\max}}, \tag{3.6}$$

де $w_{\max} = 0{,}9$, $w_{\min} = 0{,}4$ – початкове та кінцеве значення інерції; $T_{\max}$ – максимальна кількість ітерацій. Такий механізм забезпечує перехід від розвідки (великий w на ранніх ітераціях) до експлуатації (малий w на пізніх ітераціях). Швидкість обмежується покоординатно: $|v_{ij}| \le 0{,}2\,(x_j^{\max} - x_j^{\min})$.

У всіх експериментах прийнято: $c_1 = c_2 = 2{,}0$; $w_{\max} = 0{,}9$; $w_{\min} = 0{,}4$.

3.1.1 Обчислювальний бюджет

Ключовим параметром будь-якого порівняльного дослідження метаевристик є обчислювальний бюджет – максимальна кількість обчислень цільової функції (англ. Function Evaluations, FEs), дозволена кожному алгоритму. Фіксація бюджету в термінах FEs, а не ітерацій чи часу, забезпечує справедливе порівняння алгоритмів із різною обчислювальною вартістю однієї ітерації.

У даній роботі прийнято *адаптивну* стратегію визначення бюджету, стандартизовану в рамках змагань CEC з оптимізації на дійсних параметрах [26]:

$$\text{MaxFEs} = 10\,000 \times D, \tag{3.7}$$

де D – розмірність задачі (кількість змінних проєктування). Обґрунтування цієї стратегії полягає у тому, що складність задачі оптимізації зростає зі збільшенням розмірності простору пошуку: більший D потребує більшого обчислювального ресурсу для адекватного дослідження допустимої області. Множник $10\,000$ є загальновизнаним компромісом між обчислювальною економністю та достатністю бюджету для збіжності популяційних алгоритмів [26].

Для розглянутих задач це дає бюджети від $40\,000$ FEs (Welded Beam, $D = 4$) до $110\,000$ FEs (Car Side Impact, $D = 11$). Розмір популяції для всіх алгоритмів зафіксовано $N = 30$ агентів.

3.1.2 Метод штрафних функцій для задач з обмеженнями

Усі сім тестових задач містять обмеження-нерівності $g_k(\mathbf{x}) \leq 0$, $k = 1, \ldots, K$. Для їх урахування в безобмеженому оптимізаційному циклі метаевристик застосовано метод *квадратичних штрафних функцій* [4].

Штрафна цільова функція визначається як:

$$F(\mathbf{x}) = f(\mathbf{x}) + \mu \sum_{k=1}^{K} \big[\max(0,\, g_k(\mathbf{x}))\big]^2, \tag{3.8}$$

де $f(\mathbf{x})$ – початкова («сира») цільова функція; $\mu > 0$ – коефіцієнт штрафу, що визначає інтенсивність покарання за порушення обмежень. Квадрат $[\max(0, g_k)]^2$ забезпечує прогресивне зростання штрафу. Значення μ підбирається індивідуально для кожної задачі залежно від масштабу цільової функції: від $\mu = 10^4$ для задач із великими значеннями f до $\mu = 10^{12}$ для задач, де порушення обмежень має бути практично неприпустимим.

3.1.3 Статистичний апарат порівняння

Стохастична природа метаевристичних алгоритмів зумовлює необхідність статистичного аналізу результатів [9]. Порівняння алгоритмів лише за середніми значеннями цільової функції $\bar{f}$ є некоректним, оскільки розподіли результатів можуть бути несиметричними та мати важкі «хвости». Тому застосовано непараметричні статистичні тести, що не вимагають припущення про нормальність розподілу [9]. Для кожної комбінації «алгоритм × задача» проведено $R = 30$ незалежних запусків із різними початковими значеннями генератора випадкових чисел.

Тест Фрідмана [9] є непараметричним ранговим тестом для порівняння кількох пов'язаних вибірок. Він відповідає на питання: *«Чи є хоча б один алгоритм, який статистично відрізняється від решти?»*. Процедура полягає в наступному. Для кожного запуску $r = 1, \ldots, R$ значення f_r^*, отримані J алгоритмами, ранжуються від 1 (найкращий) до J (найгірший). Якщо два або більше алгоритмів мають однакові значення f_r^*, їм присвоюються усереднені ранги. Після обробки всіх R запусків для

кожного алгоритму j обчислюється середній ранг:

$$\bar{R}_j = \frac{1}{R}\sum_{r=1}^{R} R_j^{(r)}, \tag{3.9}$$

де $R_j^{(r)}$ – ранг j-го алгоритму у r-му запуску. Якщо всі алгоритми рівноцінні, їхні середні ранги мають бути близькими до $(J+1)/2$. Тестова статистика Фрідмана вимірює відхилення від цього рівномірного розподілу рангів:

$$\chi_F^2 = \frac{12R}{J(J+1)}\sum_{j=1}^{J}\left(\bar{R}_j - \frac{J+1}{2}\right)^2, \tag{3.10}$$

де $J = 7$ – кількість порівнюваних алгоритмів; $R = 30$ – кількість запусків. За великих R статистика χ_F^2 має розподіл χ^2 з $J-1$ ступенями свободи. Якщо отримане p-значення менше за рівень значущості $\alpha = 0{,}05$, нульова гіпотеза H_0 («усі алгоритми рівноцінні») відхиляється на користь альтернативної H_1 («існує щонайменше один алгоритм, який відрізняється»).

Тест Фрідмана є глобальним: він фіксує наявність відмінностей, але не вказує, між якими саме парами алгоритмів ця відмінність існує. Для відповіді на це питання застосовується попарний тест.

Парний тест Вілкоксона зі знаковими рангами [9] порівнює два алгоритми A_i та A_j на одних і тих самих R запусках. Для кожного запуску r обчислюється різниця $d_r = f_{i,r}^* - f_{j,r}^*$. Нульові різниці ($d_r = 0$) відкидаються, а решта ранжується за абсолютною величиною $|d_r|$. Визначаються суми рангів додатних різниць (W^+, коли A_j краще) та від'ємних (W^-, коли A_i краще). Тестова статистика $W = \min(W^+, W^-)$ порівнюється з табличним розподілом; якщо $p < 0{,}05$, різниця між алгоритмами визнається статистично значущою.

Перевагою тесту Вілкоксона є те, що він враховує не лише *напрямок* різниці (який алгоритм краще), а й її *величину* через ранжування абсолютних різниць. Це робить його чутливішим за тест знаків. У даній

роботі тест Вілкоксона застосовується для всіх $\binom{7}{2} = 21$ пар алгоритмів на кожній із чотирьох задач.

3.2 Тестові задачі інженерного проєктування

Для верифікації алгоритмів обрано чотири задачі оптимального конструктивного проєктування з обмеженнями, що охоплюють спектр розмірностей від 4 до 11 змінних (таблиця 3.1). Ці задачі є стандартними бенчмарками, що широко використовуються у літературі з метаевристичної оптимізації [45, 11]. Зростаюча розмірність та різна кількість обмежень дозволяють виявити сильні та слабкі сторони кожного алгоритму залежно від складності задачі.

Таблиця 3.1 – Зведена характеристика тестових задач інженерного проєктування.

№	Задача	D	Обмеж.	MaxFEs	f^* (літ.)
1	Welded Beam	4	7	40 000	1,7249
2	Pressure Vessel	4	4	40 000	5885,3
3	Speed Reducer	7	11	70 000	2996,3
4	Car Side Impact	11	8	110 000	22,84

3.2.1 Задача проєктування зварної балки

Задача проєктування зварної балки (англ. Welded Beam Design) полягає у мінімізації вартості виготовлення конструкції, що складається з горизонтальної балки, приваленої до жорсткої опори зварним швом (рис. 3.1). До вільного кінця прикладена зосереджена сила $P = 6000$ фунтів [45].

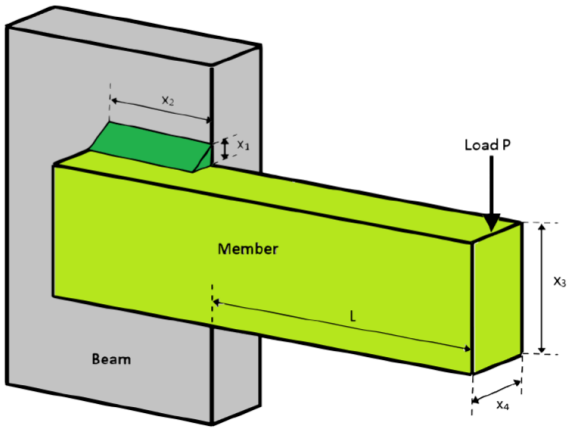

Рисунок 3.1 – Конструкція зварної балки [45]

Змінні проєктування: висота зварного шва h (x_1), довжина зварного шва l (x_2), ширина балки t (x_3) та товщина балки b (x_4). Цільова функція [45]:

$$\min_{\mathbf{x}} \; f(\mathbf{x}) = 1{,}10471\, x_1^2\, x_2 + 0{,}04811\, x_3\, x_4\, (14 + x_2). \tag{3.11}$$

Сім обмежень включають обмеження на дотичне напруження у зварному шві ($\tau \leq 13\,600$ psi), нормальне напруження згину ($\sigma \leq 30\,000$ psi), прогин вільного кінця ($\delta \leq 0{,}25$ дюйма), критичне навантаження втрати стійкості ($P \leq P_c$), мінімальну товщину шва ($x_1 \geq 0{,}125$), конструктивну вимогу ($x_1 \leq x_4$) та обмеження на вартість:

$$\begin{aligned}
&g_1 = \tau(\mathbf{x}) - 13\,600 \leq 0, && g_2 = \sigma(\mathbf{x}) - 30\,000 \leq 0,\\
&g_3 = \delta(\mathbf{x}) - 0{,}25 \leq 0, && g_4 = x_1 - x_4 \leq 0,\\
&g_5 = P - P_c(\mathbf{x}) \leq 0, && g_6 = 0{,}125 - x_1 \leq 0,\\
&g_7 = 1{,}10471\, x_1^2 + 0{,}04811\, x_3\, x_4\, (14 + x_2) - 5 \leq 0
\end{aligned} \tag{3.12}$$

де дотичне напруження у зварному шві обчислюється як:

$$\tau = \sqrt{\tau'^2 + 2\,\tau'\,\tau''\,\frac{x_2}{2R} + \tau''^2}, \quad \tau' = \frac{P}{\sqrt{2}\,x_1\,x_2}, \quad \tau'' = \frac{M\,R}{J}, \tag{3.13}$$

де $M = P\,(L + x_2/2)$ – згинальний момент; $R = \sqrt{x_2^2/4 + ((x_1 + x_3)/2)^2}$ – відстань до критичної точки; $J = 2\sqrt{2}\,x_1\,x_2\left[x_2^2/12 + ((x_1 + x_3)/2)^2\right]$ – полярний момент інерції зварного шва; $L = 14$ дюймів, $\sigma = 504\,000/(x_4\,x_3^2)$, $\delta = 2{,}1952/(x_3^3\,x_4)$.

Межі змінних: $x_1, x_4 \in [0{,}1;\ 2{,}0]$, $x_2, x_3 \in [0{,}1;\ 10{,}0]$. Відоме оптимальне значення $f^* \approx 1{,}7249$ [45].

3.2.2 Задача проєктування посудини високого тиску

Задача полягає у мінімізації загальної вартості матеріалів, зварювання та формування циліндричної ємності з напівсферичними днищами (рис. 3.2) [11].

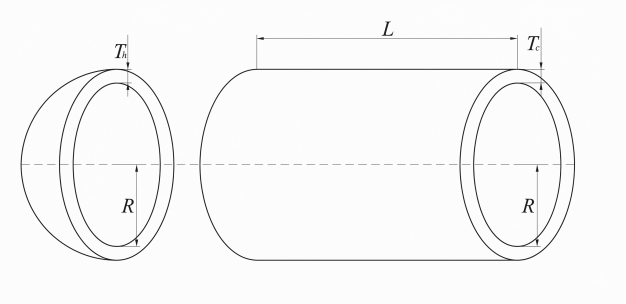

Рисунок 3.2 – Конструкція посудини високого тиску [11]

Змінні проєктування: товщина оболонки T_s (x_1), товщина днища T_h (x_2), внутрішній радіус R (x_3), довжина циліндричної частини L (x_4):

$$\min_{\mathbf{x}}\ f(\mathbf{x}) = 0{,}6224\,x_1x_3x_4 + 1{,}7781\,x_2x_3^2 + 3{,}1661\,x_1^2x_4 + 19{,}84\,x_1^2x_3, \quad (3.14)$$

$$\begin{aligned} g_1 &= -x_1 + 0{,}0193\,x_3 \le 0, & g_2 &= -x_2 + 0{,}00954\,x_3 \le 0, \\ g_3 &= -\pi x_3^2 x_4 - \tfrac{4}{3}\pi x_3^3 + 1\,296\,000 \le 0, & g_4 &= x_4 - 240 \le 0, \end{aligned} \quad (3.15)$$

де $x_1, x_2 \in [0;\ 99]$, $x_3, x_4 \in [10;\ 200]$. Відоме оптимальне значення $f^* \approx 5885{,}3$ [11].

3.2.3 Задача проєктування редуктора швидкості

Задача проєктування редуктора швидкості (англ. Speed Reducer Design) є класичним прикладом багатовимірної задачі з великою кількістю обмежень [45]. Конструкція (рис. 3.3) включає зубчасту пару та два вали.

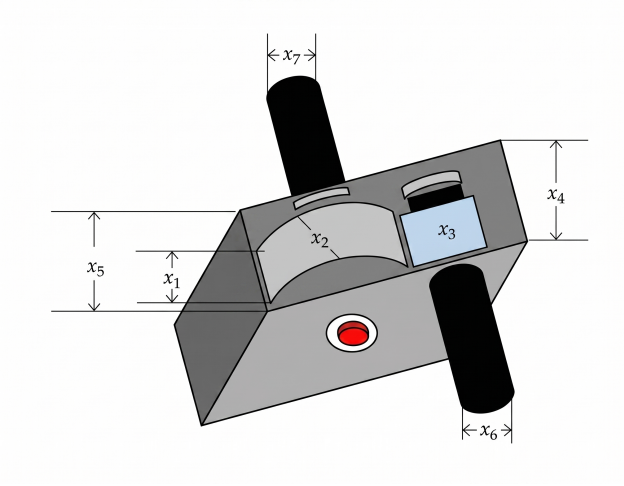

Рисунок 3.3 – Конструкція редуктора швидкості [45]

Сім змінних: ширина шестерні b (x_1), модуль зачеплення m (x_2), кількість зубців z (x_3), довжини валів l_1, l_2 (x_4, x_5), діаметри валів d_1, d_2 (x_6, x_7). Цільова функція – мінімізація маси [45]:

$$\begin{aligned} \min_{\mathbf{x}} \; f(\mathbf{x}) = {} & 0{,}7854\, x_1 x_2^2\, (3{,}3333\, x_3^2 + 14{,}9334\, x_3 - 43{,}0934) - \\ & - 1{,}508\, x_1\, (x_6^2 + x_7^2) + \\ & + 7{,}4777\, (x_6^3 + x_7^3) + 0{,}7854\, (x_4 x_6^2 + x_5 x_7^2), \end{aligned} \tag{3.16}$$

за одинадцяти обмежень (g_1–$g_{11} \leq 0$) на згинальне та контактне

напруження зубців, прогини та розміри валів [45]:

$$
\begin{aligned}
g_1 &= \frac{27}{x_1 x_2^2 x_3} - 1 \le 0, & g_2 &= \frac{397{,}5}{x_1 x_2^2 x_3^2} - 1 \le 0,\\
g_3 &= \frac{1{,}93\, x_4^3}{x_2\, x_3\, x_6^4} - 1 \le 0, & g_4 &= \frac{1{,}93\, x_5^3}{x_2\, x_3\, x_7^4} - 1 \le 0,\\
g_5 &= \frac{\sqrt{M_1^2 + 16{,}9 \cdot 10^6}}{110\, x_6^3} - 1 \le 0, & g_6 &= \frac{\sqrt{M_2^2 + 157{,}5 \cdot 10^6}}{85\, x_7^3} - 1 \le 0,\\
g_7 &= \frac{x_2 x_3}{40} - 1 \le 0, & g_8 &= \frac{5x_2}{x_1} - 1 \le 0,\\
g_9 &= \frac{x_1}{12 x_2} - 1 \le 0, & g_{10} &= \frac{1{,}5x_6 + 1{,}9}{x_4} - 1 \le 0,\\
g_{11} &= \frac{1{,}1x_7 + 1{,}9}{x_5} - 1 \le 0,
\end{aligned}
\tag{3.17}
$$

де $M_1 = 745\, x_4/(x_2 x_3)$, $M_2 = 745\, x_5/(x_2 x_3)$. Межі: $x_1 \in [2{,}6;\ 3{,}6]$, $x_2 \in [0{,}7;\ 0{,}8]$, $x_3 \in [17;\ 28]$, $x_4 \in [7{,}3;\ 8{,}3]$, $x_5 \in [7{,}8;\ 8{,}3]$, $x_6 \in [2{,}9;\ 3{,}9]$, $x_7 \in [5{,}0;\ 5{,}5]$. Відоме $f^* \approx 2996{,}3$ [45].

3.2.4 Задача оптимізації конструкції автомобіля для захисту від бокового удару

Модель базується на сурогатних поліноміальних апроксимаціях результатів скінченно-елементного моделювання краш-тесту бокового удару [13].

Одинадцять змінних: $x_1, \ldots, x_7 \in [0{,}5;\ 1{,}5]$ – товщини конструктивних елементів; $x_8, x_9 \in \{0;\ 1\}$ – вибір матеріалу; $x_{10}, x_{11} \in [-30;\ 30]$ – параметри шуму бар'єра. Цільова функція – мінімізація маси [11]:

$$
\min_{\mathbf{x}}\ f(\mathbf{x}) = 2 + 4{,}9\, x_1 + 6{,}7\, x_2 + 7\, x_3 + 4\, x_4 + 1{,}8\, x_5 + 2{,}7\, x_7. \tag{3.18}
$$

Вісім обмежень формулюються через сурогатні поліноміальні моделі

відгуків. Нижче наведено повні вирази, що відповідають рівнянням (2)–(9) оригінальної роботи:

$$
\begin{aligned}
V_B &= 10{,}6 - 0{,}67\,x_1x_2 - 2{,}0\,x_2x_8 + 0{,}02\,x_3x_{10} - \\
&\quad - 0{,}02\,x_4x_{10} + 0{,}28\,x_6x_{10}, \\
V_{\text{door}} &= 16{,}4 - 0{,}49\,x_3x_4 - 0{,}84\,x_5x_6 + 0{,}43\,x_9x_{10} - \\
&\quad - 0{,}56\,x_9x_{11} + 0{,}0008\,x_{11}^2, \\
F_{\text{abd}} &= 1{,}163 - 0{,}3717\,x_{10}x_2 - 0{,}484\,x_3x_9 + 0{,}01343\,x_6x_{10}, \\
VC_{\text{up}} &= 0{,}28 - 0{,}016\,x_1x_2 - 0{,}19\,x_2x_7 + 0{,}014\,x_3x_5 + \\
&\quad + 0{,}01\,x_5x_{10} + 0{,}08\,x_6x_9 + 0{,}001\,x_8x_{11}, \\
VC_{\text{low}} &= 0{,}74 - 0{,}61\,x_2 - 0{,}13\,x_3x_8 + 0{,}0012\,x_3x_{10} - \\
&\quad - 0{,}17\,x_7x_9 + 0{,}023\,x_2^2, \\
D_{\text{up}} &= 29{,}0 - 4{,}2\,x_1x_2 - 7{,}8\,x_7x_8 + 0{,}021\,x_5x_{10} + \\
&\quad + 6{,}6\,x_6x_9 + 0{,}32\,x_9x_{10}, \\
D_{\text{low}} &= 46{,}4 - 9{,}9\,x_2 - 12{,}89\,x_1x_8 + 0{,}11\,x_3x_{10}, \\
F_{\text{pub}} &= 4{,}7 - 0{,}5\,x_4 - 0{,}19\,x_2x_3 - 0{,}012\,x_4x_{10} + \\
&\quad + 0{,}009\,x_6x_{10} + 0{,}0002\,x_{11}^2,
\end{aligned}
\tag{3.19}
$$

де V_B – швидкість B-стійки (мм/мс); V_{door} – швидкість передньої дверної панелі (мм/мс); F_{abd} – сила на живіт (кН); $VC_{\text{up}}, VC_{\text{low}}$ – швидкісні критерії ушкоджень верхніх та нижніх ребер (м/с); $D_{\text{up}}, D_{\text{low}}$ – прогини верхніх та нижніх ребер (мм); F_{pub} – сила на лобковий симфіз (кН). Обмеження:

$$
\begin{aligned}
&g_1 = F_{\text{abd}} - 1{,}0 \le 0, && g_2 = VC_{\text{up}} - 0{,}32 \le 0, \\
&g_3 = VC_{\text{low}} - 0{,}32 \le 0, && g_4 = D_{\text{up}} - 32 \le 0, \\
&g_5 = D_{\text{low}} - 32 \le 0, && g_6 = F_{\text{pub}} - 4{,}0 \le 0, \\
&g_7 = V_B - 9{,}9 \le 0, && g_8 = V_{\text{door}} - 15{,}7 \le 0.
\end{aligned}
\tag{3.20}
$$

Відоме оптимальне значення детерміністичної задачі $f^* \approx 23{,}59$ кг [13]. Слід зазначити, що оригінальна постановка [13] містить додаткові обмеження на критерій травми голови (HIC $\le$ 650) та середній прогин

ребер ($D_{\text{mid}} \leq 32$ мм), для яких явні поліноміальні моделі відгуку у рівняннях (1)–(9) оригінальної роботи не наведені. У даному дослідженні використано лише 8 обмежень, для яких доступні сурогатні моделі, що робить допустиму область ширшою і може приводити до нижчих значень f порівняно з літературним оптимумом.

3.3 Результати обчислювальних експериментів

Для кожної з чотирьох задач усі сім алгоритмів (SSA, DBO, DPDBO, TSA, SBOA, GA, PSO) запущено 30 разів із різними параметрами ініціалізації генератора псевдовипадкових чисел. Результати подано у вигляді таблиць описової статистики (Best, Mean, Median, Worst, Std, Feas%), кривих збіжності та діаграм розмаху. Жирним шрифтом у таблицях виділено найкращі значення по кожному стовпцю. Стовпець Feas% відображає частку запусків, у яких найкращий знайдений розв'язок задовольняє усі обмеження задачі.

3.3.1 Результати для задачі Welded Beam

Задача Welded Beam має 4 змінних проєктування та 7 обмежень, що робить її однією з найскладніших за структурою допустимої області серед низькорозмірних задач. Результати 30 незалежних запусків наведено у таблиці 3.2.

На задачі Welded Beam з 7 обмеженнями PSO демонструє найкращу медіанну продуктивність: Median = 1,72485, що збігається з відомим оптимумом. Високі результати PSO на цій 4-вимірній задачі пояснюються тим, що його базова математична модель (3.4) дозволяє швидко та надійно сходитися до оптимуму в просторах невеликої розмірності. SBOA посідає друге місце з ще меншою варіацією (Std $= 7{,}45 \times 10^{-4}$ проти 0,016 у PSO), що свідчить про його вищу стабільність. GA має найбільший

Таблиця 3.2 – Описова статистика $f(\mathbf{x}^\star)$: **Welded Beam** ($R = 30$, MaxFEs= 40 020).

Алгоритм	Best	Mean	Median	Worst	Std	Feas%
SSA	1,72669	1,76146	1,74379	1,93160	0,04646	100
DBO	**1,72485**	1,75571	1,72643	1,81465	0,04222	100
DPDBO	1,72485	1,74824	1,72964	1,81429	0,03270	100
TSA	1,73695	1,74509	1,74508	1,75421	0,00428	100
SBOA	1,72485	**1,72510**	1,72486	**1,72866**	**7,45e-04**	100
GA	1,72692	2,06511	2,01564	2,68934	0,30258	100
PSO	**1,72485**	1,72789	**1,72485**	1,81429	0,01632	100

розкид (Std = 0,303, Worst = 2,689), що свідчить про систематичне застрягання у локальних мінімумах.

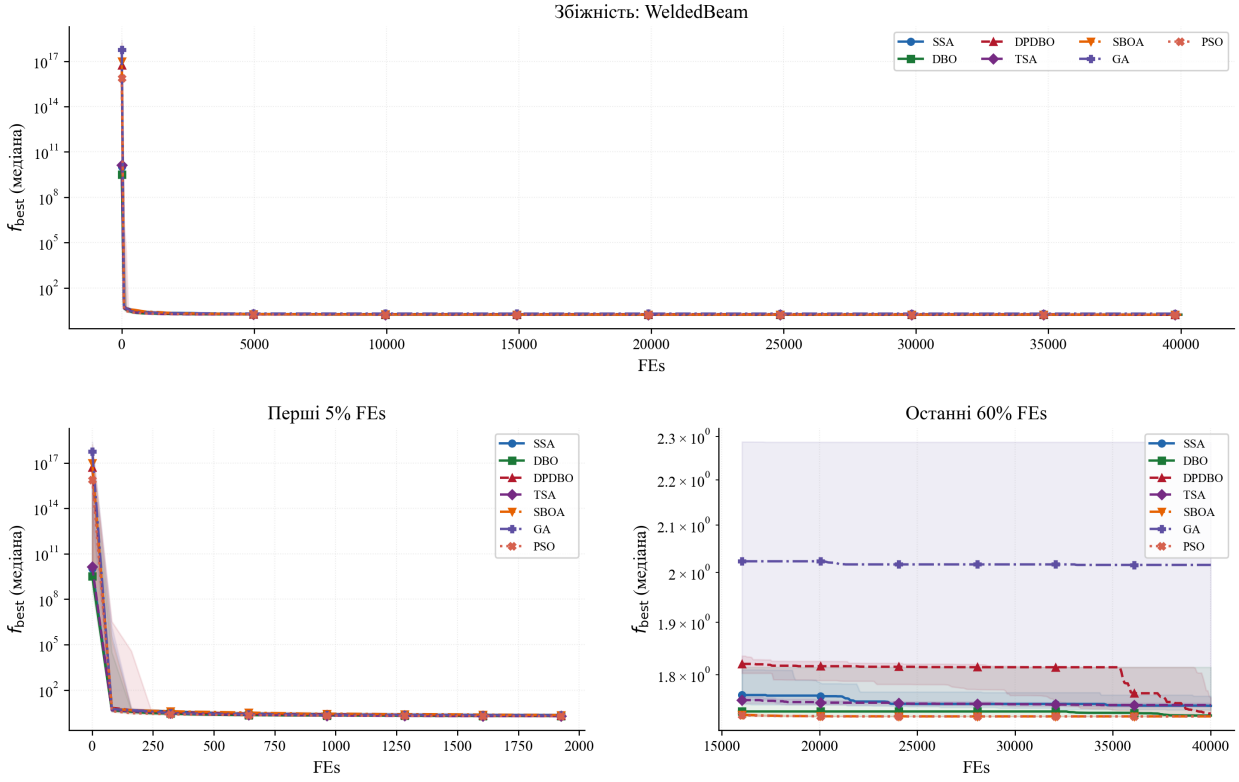

Рисунок 3.4 – Криві збіжності для задачі Welded Beam

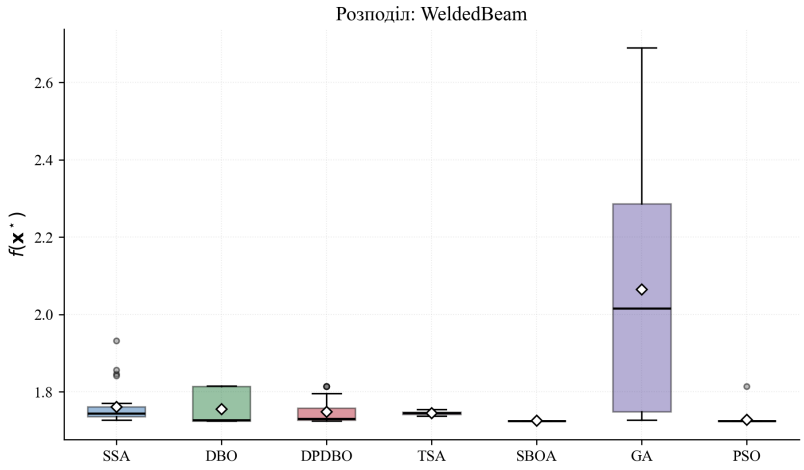

Рисунок 3.5 – Діаграма розмаху для задачі Welded Beam

Діаграма розмаху (рис. 3.5) наочно підтверджує домінування PSO та SBOA за медіаною, а також аномально великий розкид GA.

3.3.2 Результати для задачі Pressure Vessel

Задача Pressure Vessel має 4 змінних проєктування та 4 обмеження. Результати 30 запусків наведено у таблиці 3.3.

Таблиця 3.3 – Описова статистика $f(\mathbf{x}^\star)$: **Pressure Vessel** ($R = 30$, MaxFEs= 40 020).

Алгоритм	**Best**	**Mean**	**Median**	**Worst**	**Std**	**Feas%**
SSA	6017,02	6833,52	6777,00	7416,05	455,41	93
DBO	**5885,33**	6087,28	5894,22	7319,00	428,75	100
DPDBO	**5885,33**	5972,31	**5885,33**	7319,00	267,92	100
TSA	5915,00	6009,89	5952,03	7249,74	242,17	93
SBOA	**5885,33**	**5905,78**	5891,42	**6078,42**	**37,20**	100
GA	5917,24	6490,06	6498,77	7367,09	380,00	100
PSO	**5885,33**	6337,28	6160,27	7319,00	468,03	100

SBOA стабільно знаходить оптимум $f^* = 5885{,}33$ з мінімальним розкидом (Std = 37,2), що на порядок менше за інші алгоритми. Примітним є результат DPDBO: медіана збігається з відомим оптимумом, а Std = 267,9 суттєво менша за Std базового DBO (428,8), що підтверджує

ефективність стратегій динамічної адаптації. SSA та TSA мають Feas% = 93, тобто у 7% запусків найкращий знайдений розв'язок порушує обмеження, що свідчить про недостатню здатність цих алгоритмів до навігації у допустимій області.

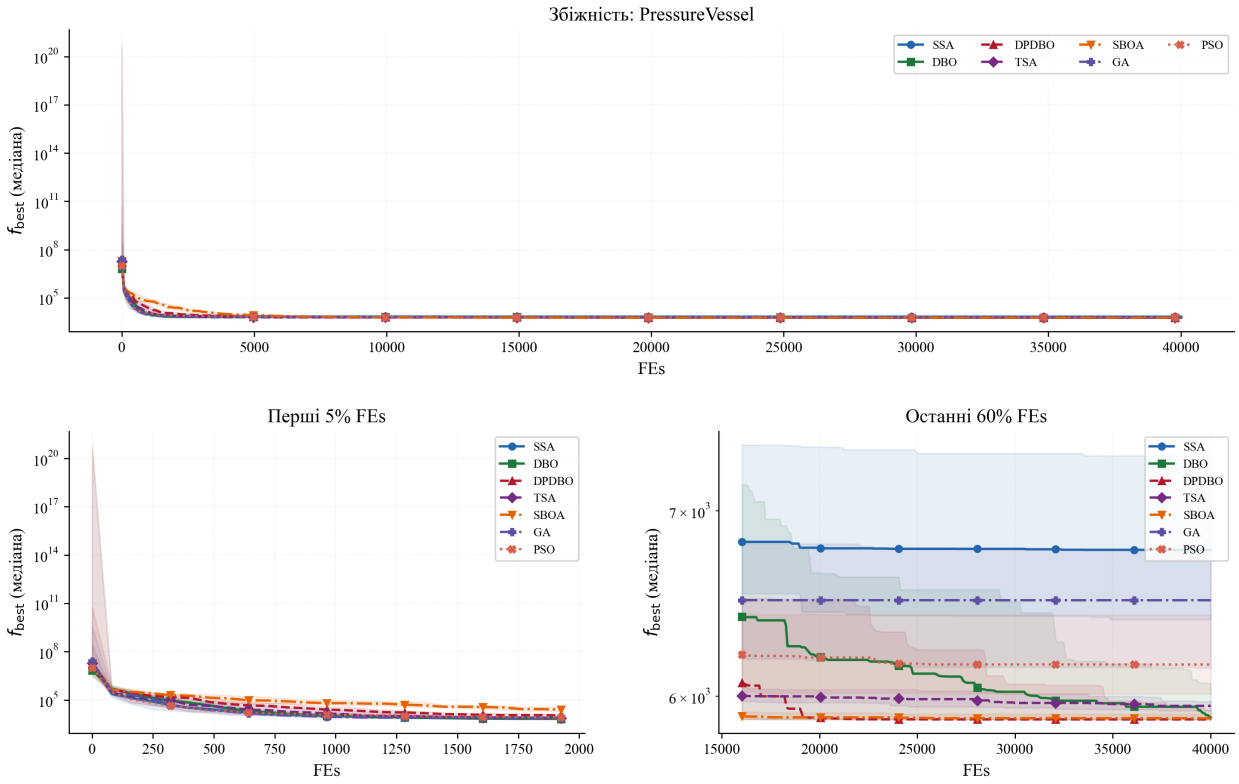

Рисунок 3.6 – Криві збіжності для задачі Pressure Vessel

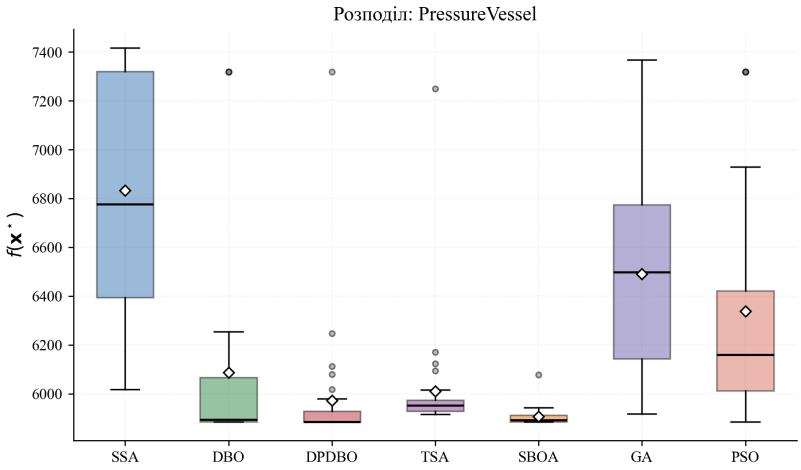

Рисунок 3.7 – Діаграма розмаху для задачі Pressure Vessel

Діаграма розмаху (рис. 3.7) ілюструє компактний розподіл SBOA та DPDBO порівняно зі значним розкидом SSA та GA.

3.3.3 Результати для задачі Speed Reducer

Задача Speed Reducer є найскладнішою серед обраних за кількістю обмежень: 7 змінних та 11 нерівностей. Результати наведено у таблиці 3.4.

Таблиця 3.4 – Описова статистика $f(\mathbf{x}^\star)$: **Speed Reducer** ($R = 30$, MaxFEs= 70 020).

Алгоритм	**Best**	**Mean**	**Median**	**Worst**	**Std**	**Feas%**
SSA	2998,45	3003,97	3004,36	3011,19	3,39	90
DBO	**2996,35**	2999,97	2996,35	3035,63	7,73	100
DPDBO	**2996,35**	2996,43	**2996,35**	2998,10	0,33	97
TSA	3019,99	3043,70	3045,08	3062,00	9,89	100
SBOA	**2996,35**	**2996,35**	**2996,35**	**2996,35**	**8,27e-13**	100
GA	2996,35	2996,36	2996,36	2996,38	0,0076	97
PSO	**2996,35**	3128,85	3035,63	5547,54	458,69	100

Speed Reducer є задачею із найбільшою кількістю обмежень ($K = 11$), що створює значні труднощі для пошуку допустимих розв'язків. SBOA демонструє стовідсоткову надійність, збігаючись до $f^* = 2996{,}35$ у кожному запуску з практично нульовою варіацією (Std $\approx 10^{-13}$). DPDBO стабільно посідає друге місце (Std = 0,33), що на порядок менше за Std базового DBO (7,73). GA також досягає оптимуму з мінімальним розкидом (Std = 0,008), що свідчить про ефективність елістичної стратегії $(\mu + \lambda)$ на цій задачі. Натомість PSO демонструє катастрофічну деградацію: $\bar{f} = 3129$ – на 4,4% гірше за оптимум, а Worst = 5548 – відхилення на 85%. Std = 459 – на 5 порядків більше за SBOA, що свідчить про повну нездатність когнітивно-соціальної моделі до навігації у 7-вимірному просторі з 11 обмеженнями.

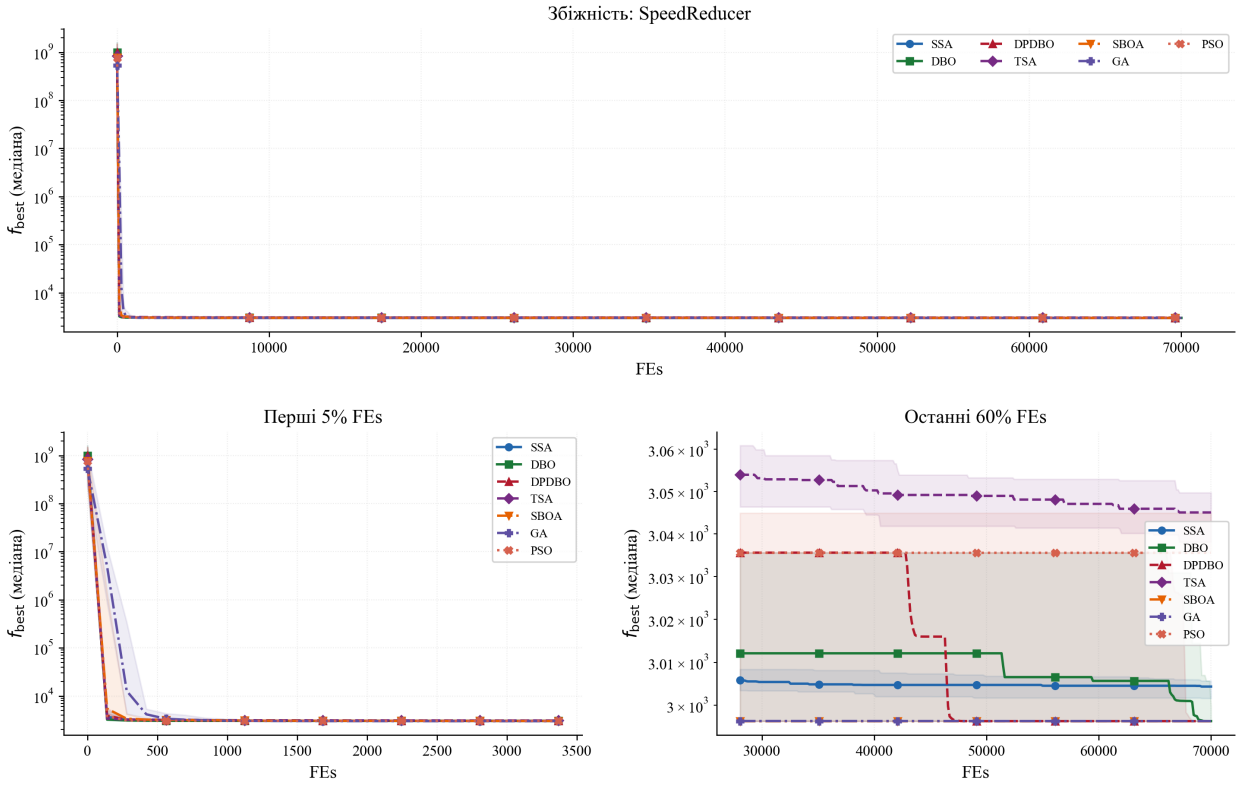

Рисунок 3.8 – Криві збіжності для задачі Speed Reducer

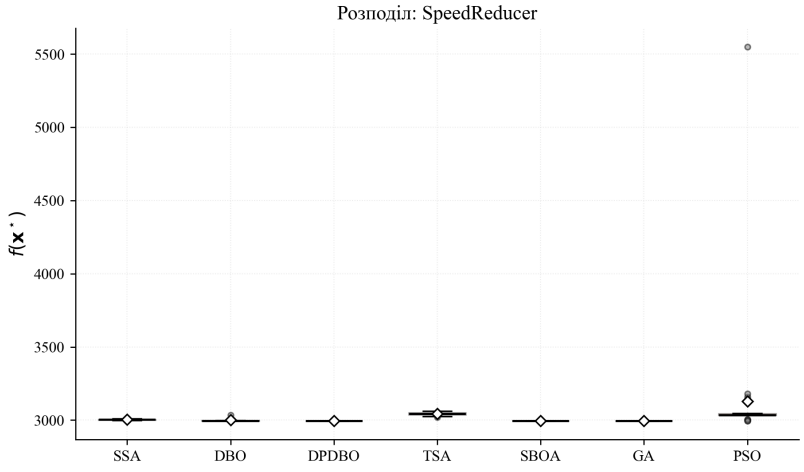

Рисунок 3.9 – Діаграма розмаху для задачі Speed Reducer

Діаграма розмаху (рис. 3.9) наочно демонструє катастрофічний розкид PSO та контраст із практично детерміністичною збіжністю SBOA.

3.3.4 Результати для задачі Car Side Impact

Car Side Impact є найвищерозмірною задачею дослідження: 11 змінних та 8 обмежень. Результати наведено у таблиці 3.5.

Таблиця 3.5 – Описова статистика $f(\mathbf{x}^\star)$: **Car Side Impact** ($R = 30$, MaxFEs= 110 010).

Алгоритм	**Best**	**Mean**	**Median**	**Worst**	**Std**	**Feas%**
SSA	19,774	19,958	19,909	20,505	0,181	100
DBO	19,672	20,061	19,913	21,574	0,442	100
DPDBO	19,670	20,007	19,882	20,593	0,334	100
TSA	19,880	20,188	20,163	20,870	0,230	100
SBOA	**19,669**	**19,670**	**19,670**	**19,673**	**9,23e-04**	100
GA	19,895	20,634	20,518	22,926	0,629	100
PSO	**19,669**	19,970	20,031	20,672	0,296	100

На найвищерозмірній задачі ($D = 11$) SBOA демонструє радикальну перевагу: Std $= 9{,}23 \times 10^{-4}$, що на 2–3 порядки менше за будь-який інший алгоритм. Це означає, що SBOA знаходить оптимум $f^* = 19{,}669$ кг у кожному із 30 запусків з майже детерміністичною точністю. PSO посідає друге місце за тестом Фрідмана (ранг 3,20) із середнім $\bar{f} = 19{,}970$, що є найкращим результатом серед нелідерських алгоритмів. GA демонструє найгірше середнє ($\bar{f} = 20{,}634$) та найбільший розкид (Std = 0,629, Worst = 22,926), що свідчить про неефективність еволюційних операторів у 11-вимірному просторі з 8 обмеженнями.

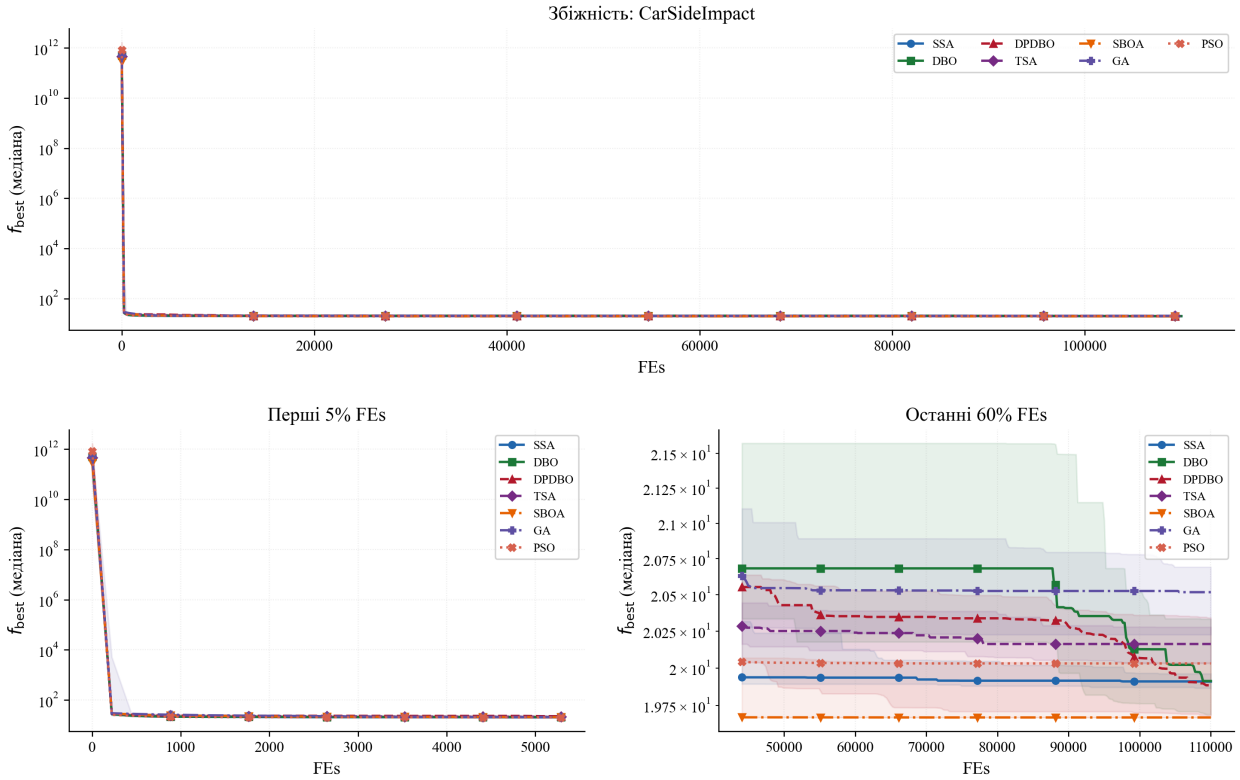

Рисунок 3.10 – Криві збіжності для задачі Car Side Impact

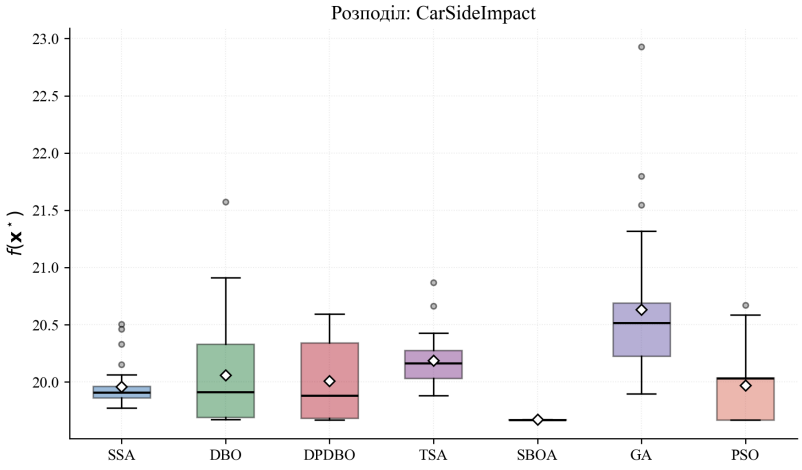

Рисунок 3.11 – Діаграма розмаху для задачі Car Side Impact

Діаграма розмаху (рис. 3.11) підтверджує радикальну перевагу SBOA у стабільності на найвищерозмірній задачі дослідження.

3.4 Порівняльний аналіз збіжності та ефективності алгоритмів

3.4.1 Обчислювальна складність

Перед аналізом якості розв'язків доцільно порівняти обчислювальну складність алгоритмів. Нехай N – розмір популяції (кількість агентів), D – розмірність задачі (кількість змінних проєктування), а MaxFEs – обчислювальний бюджет. Усі алгоритми мають однакову просторову складність $O(N \cdot D)$, що визначається зберіганням матриці позицій популяції розміром $N \times D$. Часова складність одного повного запуску також однакова – $O(\text{MaxFEs} \cdot D)$ – для всіх алгоритмів, окрім SBOA. У SBOA коефіцієнт подвоюється ($O(2 \cdot \text{MaxFEs} \cdot D)$), бо на кожній ітерації виконуються дві фази: полювання (розвідка) та втеча (експлуатація). Оскільки порівняння у даній роботі базується на фіксованому бюджеті FEs (3.7), а не на кількості ітерацій, ця різниця не порушує чесності порівняння.

3.4.2 Статистичне порівняння алгоритмів

Для кожної з чотирьох задач проведено тест Фрідмана. На всіх задачах отримано $p \ll 0{,}05$, що підтверджує статистично значущу різницю між алгоритмами. Детальні таблиці рангів Фрідмана для кожної задачі наведено у Додатку Б.1.

Для попарного порівняння алгоритмів застосовано тест Вілкоксона зі знаковими рангами. У таблиці 3.6 наведено результати порівняння SBOA з кожним із шести конкурентів на чотирьох задачах. Позначення: «+» – SBOA статистично значущо кращий ($p < 0{,}05$); «−» – SBOA значущо гірший; «≈» – різниця незначуща.

Таблиця 3.6 – Результати парного тесту Вілкоксона: SBOA проти кожного алгоритму ($\alpha = 0{,}05$).

Пара	Задача				Підсумок		
	Welded	Vessel	Reducer	CarSide	+	−	≈
SBOA vs DBO	+	≈	+	+	3	0	1
SBOA vs DPDBO	+	≈	+	+	3	0	1
SBOA vs SSA	+	+	+	+	4	0	0
SBOA vs TSA	+	+	+	+	4	0	0
SBOA vs GA	+	+	+	+	4	0	0
SBOA vs PSO	−	+	+	≈	2	1	1

Результати тесту Вілкоксона підтверджують домінування SBOA: він статистично значущо кращий за GA, SSA та TSA на *всіх* чотирьох задачах (4/4). Перевага SBOA над DBO та DPDBO є значущою на 3 із 4 задач; на Pressure Vessel різниця незначуща, оскільки DPDBO посідає перше місце на цій задачі. Єдиний програш SBOA – на Welded Beam ($D = 4$) проти PSO, де когнітивно-соціальна модель PSO ефективніша у низькорозмірному просторі.

Аналогічно, у таблиці 3.7 наведено порівняння DPDBO із базовим DBO та класичними методами для підтвердження ефективності запропонованих модифікацій.

Таблиця 3.7 – Результати парного тесту Вілкоксона: DPDBO проти DBO, GA та PSO ($\alpha = 0{,}05$).

Пара	Задача				Підсумок		
	Welded	Vessel	Reducer	CarSide	+	−	≈
DPDBO vs DBO	≈	+	+	≈	2	0	2
DPDBO vs GA	+	+	+	+	4	0	0
DPDBO vs PSO	−	+	+	≈	2	1	1

DPDBO статистично значущо кращий за GA на всіх 4 задачах.

Порівняно з базовим DBO, DPDBO значущо кращий на 2 задачах (Pressure Vessel та Speed Reducer) і не поступається на решті, що підтверджує ефективність стратегій динамічної адаптації параметрів та перезапуску.

3.4.3 Агреговане ранжування

Для узагальненої оцінки обчислено середню позицію кожного алгоритму за тестом Фрідмана по чотирьох задачах (таблиця 3.8).

Таблиця 3.8 – Агреговане ранжування алгоритмів за 4 задачами (середня позиція за тестом Фрідмана).

Алгоритм	Середня позиція	Перше місце	Топ-3	Загальний ранг
SBOA	**1,50**	2/4	4/4	**1**
DPDBO	2,75	1/4	2/4	2
DBO	3,50	0/4	3/4	3
PSO	3,50	1/4	2/4	4
SSA	5,25	0/4	1/4	5
TSA	5,50	0/4	0/4	6
GA	6,00	0/4	0/4	7

Результати агрегованого ранжування дозволяють сформулювати кілька ключових висновків.

SBOA – безумовний лідер. Алгоритм птаха-секретаря входить до трійки лідерів на *всіх* чотирьох задачах без винятку ($\bar{P} = 1{,}50$), посідаючи перше місце на Speed Reducer та Car Side Impact. Перевага SBOA є особливо вираженою у стабільності: на трьох із чотирьох задач стандартне відхилення менше 10^{-3}, тобто алгоритм знаходить практично один і той же оптимум у кожному запуску. Тест Вілкоксона підтверджує:

SBOA статистично значущо кращий за GA, SSA та TSA на всіх 4 задачах [11].

DPDBO – надійний другий. Покращена версія DBO посідає друге місце у загальному ранжуванні ($\bar{P} = 2{,}75$). Особливо значущим є результат на Pressure Vessel, де DPDBO посідає *перше* місце, випереджаючи SBOA. Тест Вілкоксона показує, що DPDBO значущо кращий за базовий DBO на 2 із 4 задач і не поступається на решті, що підтверджує ефективність стратегій динамічної адаптації параметрів та перезапуску. Порівняно з DBO ($\bar{P} = 3{,}50$), DPDBO покращує середню позицію на 21%.

PSO – ефективний лише у низькорозмірних просторах. PSO перемагає на Welded Beam ($D = 4$), де його когнітивно-соціальна модель (3.4) ефективно працює у компактному просторі з чітко вираженим глобальним мінімумом. Проте зі зростанням розмірності PSO катастрофічно деградує: на Speed Reducer ($D = 7$) він посідає 6-те місце із $\bar{f} = 3129$ (на 4,4% гірше за оптимум) та Worst = 5548. Тест Вілкоксона підтверджує: PSO значущо кращий за SBOA лише на Welded Beam; на решті задач SBOA або кращий, або різниця незначуща.

DBO – стабільний, але обмежений. Базовий оптимізатор гнойового жука стабільно входить до трійки лідерів на 3 із 4 задач, проте на Car Side Impact ($D = 11$) опускається на 5-те місце. Це пряме підтвердження необхідності модифікацій, реалізованих у DPDBO.

GA – систематичний аутсайдер. Генетичний алгоритм посідає останнє або передостаннє місце на трьох із чотирьох задач (Welded Beam – 7-ме, Car Side Impact – 7-ме, Pressure Vessel – 6-те). Тест Вілкоксона підтверджує: і SBOA, і DPDBO статистично значущо кращі за GA на всіх 4 задачах. Елістична стратегія $(\mu + \lambda)$ забезпечує надійну експлуатацію, але недостатню розвідку складних допустимих областей.

SSA та TSA – аутсайдери. Обидва алгоритми жодного разу не посідають першого місця ($\bar{P} > 5$). TSA не входить до трійки лідерів на жодній задачі.

Висновки до розділу 3

У 3-ому розділі виконано комплексне чисельне дослідження ефективності семи метаевристичних алгоритмів на 4 задачах інженерного проєктування з обмеженнями (Welded Beam, Pressure Vessel, Speed Reducer, Car Side Impact) розмірністю 4–11 змінних.

Обґрунтовано методику порівняння: адаптивний обчислювальний бюджет $\text{MaxFEs} = 10\,000 \times D$ забезпечує справедливу оцінку; метод квадратичних штрафних функцій інтегрує обм-ня; непараметричні тести Фрідмана та Вілкоксона надають статистично обґрунтовані висновки.

Результати 30 незалежних запусків на кожній із 28 комбінацій «алгоритм × задача» дозволяють зробити наступні висновки:

1) алгоритм птаха-секретаря (SBOA) є безумовним лідером: середня позиція за тестом Фрідмана $\bar{P} = 1{,}50$, входження до трійки лідерів на всіх 4 задачах. Тест Вілкоксона підтверджує статистично значущу перевагу SBOA над GA, SSA та TSA на всіх задачах;

2) покращений оптимізатор гнойового жука (DPDBO) стабільно посідає друге місце ($\bar{P} = 2{,}75$) та перемагає SBOA на задачі Pressure Vessel. Порівняно з базовим DBO ($\bar{P} = 3{,}50$), DPDBO покращує ранг на 21%, що підтверджено тестом Вілкоксона;

3) PSO конкурентоспроможний лише на низькорозмірних задачах ($D = 4$): перше місце на Welded Beam, але 6-те на Speed Reducer ($D = 7$) із катастрофічним Worst = 5548 при оптимумі 2996;

4) GA систематично поступається ройовим алгоритмам: останнє місце на Welded Beam та Car Side Impact, тест Вілкоксона підтверджує значущу перевагу SBOA та DPDBO на всіх задачах;

5) Жоден алгоритм не є найкращим на всіх задачах одночасно. Тим не менш, SBOA демонструє найбільш стійку та збалансовану продуктивність, а DPDBO – найвищу консистентність серед нелідерських алгоритмів.

ВИСНОВКИ

В ході даної бакалаврської роботи було розглянуто та реалізовано підходи до ідентифікації оптимальних параметрів математичних моделей у задачах інженерного проєктування з використанням сучасних метаевристичних методів ройового інтелекту. З метою визначення ефективності алгоритмів було виконано усі завдання досліджень та отримано відповіді на поставлені дослідницькі питання.

Спочатку було проведено огляд сучасних практичних задач, пов'язаних із проєктуванням інтегрованих енергетичних систем, теплообмінників та елементів автомобілебудування. Під час виконання цього етапу було встановлено, що строга математична формалізація у вигляді задач умовної оптимізації є необхідною передумовою успішного проєктування, проте класичні детерміновані підходи часто не здатні впоратися зі складною нелінійною природою цих задач та їхніми жорсткими обмеженнями.

На наступному етапі було досліджено сучасні біоінспіровані методи оптимізації, зокрема алгоритми пошуку горобців (SSA), гнойового жука (DBO) та його вдосконалену версію (DPDBO), рою тунікатів (TSA), птаха-секретаря (SBOA), а також класичні GA і PSO. Аналіз показав, що ефективність пошуку глобального екстремуму критично залежить від механізмів балансування між розвідкою та експлуатацією, а модифіковані стратегії (як у DPDBO чи SBOA) дозволяють значно знизити ризик застрягання алгоритму в локальних мінімумах.

Для практичної перевірки досліджених методів було розроблено програмне забезпечення та розв'язано чотири класичні задачі інженерного проєктування. У програмний комплекс було успішно інтегровано метод квадратичних штрафних функцій, що дозволило алгоритмам коректно обробляти жорсткі експлуатаційні вимоги до конструкцій зварної балки, посудини високого тиску, редуктора

швидкості та елементів пасивної безпеки автомобіля.

Останнім етапом стало проведення порівняльного аналізу отриманих результатів на основі 30 незалежних запусків із використанням статистичних тестів Фрідмана та Вілкоксона. Обчислювальні експерименти підтвердили високу ефективність алгоритму SBOA, який продемонстрував найвищу точність у більшості тестів, зокрема з мінімальним середнім квадратичним відхиленням ($9{,}23 \times 10^{-4}$) у складній задачі Car Side Impact. Модифікований алгоритм DPDBO завдяки стратегії динамічної популяції суттєво перевершив за точністю базову версію DBO, тоді як класичний метод PSO виявився конкурентоспроможним лише у просторах малої розмірності, продемонструвавши значну деградацію на складніших інженерних задачах.

Напрямки подальших досліджень у даній тематиці пов'язані з розширенням випробувальної бази алгоритмів на інженерні задачі вищої розмірності та безпосереднім переходом до багатокритеріальних постановок Парето-оптимізації. Також перспективним є дослідження більш гнучких адаптивних методів обробки нелінійних обмежень, що дозволить відійти від жорстких коефіцієнтів штрафних функцій, та розробка нових гібридних схем ройових оптимізаторів із залученням процедур локального пошуку.

ПЕРЕЛІК ПОСИЛАНЬ

[1] Ali Belhocine, Dinesh Shinde та Rahul Patil. «Thermo-mechanical coupled analysis based design of ventilated brake disc using genetic algorithm and particle swarm optimization». Англ. В: *JMST Advances* 3 (2021), с. 163—176. DOI: 10.1007/s42791-021-00040-0.

[2] Zhang Chen та ін. «Optimal design and performance assessment for a solar powered electricity, heating and hydrogen integrated energy system». Англ. В: *Energy* 262 (2023). DOI: 10.1016/j.energy.2022.125453.

[3] Navneet Chouhan та Sunil C. Jain. «Tunicate Swarm Grey Wolf Optimization for Multi-path Routing Protocol in IoT Assisted WSN Networks». В: *Journal of Ambient Intelligence and Humanized Computing* 12 (2021), с. 3285—3301. DOI: 10.1007/s12652-020-02657-w.

[4] Carlos A. Coello Coello. «Theoretical and Numerical Constraint-Handling Techniques Used with Evolutionary Algorithms: A Survey of the State of the Art». В: *Computer Methods in Applied Mechanics and Engineering* 191.11–12 (2002), с. 1245—1287. DOI: 10.1016/S0045-7825(01)00323-1.

[5] Yafei Cui, Rui Shi та Jiacheng Dong. «CLTSA: A Novel Tunicate Swarm Algorithm Based on Chaotic-Lévy Flight Strategy for Solving Optimization Problems». В: *Mathematics* 10.18 (2022), с. 3405. DOI: 10.3390/math10183405.

[6] K. Deb та ін. «A fast and elitist multiobjective genetic algorithm: NSGA-II». В: *IEEE Transactions on Evolutionary Computation* 6.2 (2002), с. 182—197. DOI: 10.1109/4235.996017.

[7] Kalyanmoy Deb та Ram B. Agrawal. «Simulated Binary Crossover for Continuous Search Space». В: *Complex Systems* 9.2 (1995), с. 115—148.

[8] Kalyanmoy Deb та Mayank Goyal. «A Combined Genetic Adaptive Search (GeneAS) for Engineering Design». В: *Computer Science and Informatics* 26.4 (1996), с. 30—45.

[9] Joaquín Derrac та ін. «A Practical Tutorial on the Use of Nonparametric Statistical Tests as a Methodology for Comparing Evolutionary and Swarm Intelligence Algorithms». В: *Swarm and Evolutionary Computation* 1.1 (2011), с. 3—18. DOI: 10.1016/j.swevo.2011.02.002.

[10] Tamer Fetouh та Abdallah M. Elsayed. «Optimal Control and Operation of Fully Automated Distribution Networks Using Improved Tunicate Swarm Intelligent Algorithm». В: *IEEE Access* 8 (2020), с. 129689—129708. DOI: 10.1109/ACCESS.2020.3009113.

[11] Youfa Fu та ін. «Secretary bird optimization algorithm: a new metaheuristic for solving global optimization problems». В: *Artificial Intelligence Review* 57.5 (2024), с. 123. DOI: 10.1007/s10462-024-10729-y.

[12] David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[13] Liya Gu та ін. «Optimisation and robustness for crashworthiness of side impact». В: *International Journal of Vehicle Design - INT J VEH DES* 26 (січ. 2001). DOI: 10.1504/IJVD.2001.005210.

[14] Jitendra Gupta, Parag Nijhawan та Sourabh Ganguli. «Parameter Extraction of Solar PV Cell Models Using Novel Metaheuristic Chaotic Tunicate Swarm Algorithm». В: *International Transactions on Electrical Energy Systems* 31.12 (2021), e13244. DOI: 10.1002/2050-7038.13244.

[15] D. Gürses та ін. «Artificial gorilla troops algorithm for the optimization of a fine plate heat exchanger». В: *Materials Testing* 64.9 (2022), с. 1325—1331. DOI: 10.1515/mt-2022-0049.

[16] John H. Holland. *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: University of Michigan Press, 1975.

[17] Essam H. Houssein та ін. «An Improved Tunicate Swarm Algorithm for Global Optimization and Image Segmentation». В: *IEEE Access* 9 (2021), с. 56066—56092. DOI: 10.1109/ACCESS.2021.3072336.

[18] Xu Ji та ін. «Improved DBO Algorithm Incorporating Disorienting Behavior and Dynamic Population Strategy for Engineering Problem Solving». В: URL: https://api.semanticscholar.org/CorpusID:279053936.

[19] Kanak Kalita, Dinesh Shinde та Shankar Chakraborty. «Grey wolf optimizer-based design of ventilated brake disc». В: *Journal of the Brazilian Society of Mechanical Sciences and Engineering* 43.8 (2021), с. 405. DOI: 10.1007/s40430-021-03125-y.

[20] Satnam Kaur та ін. «Tunicate Swarm Algorithm: A new bio-inspired optimization metaheuristic for engineering optimization problems». В: *Computational Intelligence* 36.4 (2020), с. 1186—1214. DOI: 10.1016/j.engappai.2020.103541.

[21] J. Kennedy та R. Eberhart. «Particle swarm optimization». В: *Proceedings of ICNN'95 - International Conference on Neural Networks*. Т. 4. 1995, 1942—1948 vol.4. DOI: 10.1109/ICNN.1995.488968.

[22] Mohammad Khajehzadeh та ін. «Seismic Analysis of Earth Slope Using a Novel Sequential Hybrid Optimization Algorithm». В: *Periodica Polytechnica Civil Engineering* 66.2 (2022), с. 355—366. DOI: 10.3311/PPci.19356.

[23] Jingyun Li та Hong Zhao. «Multi-Objective Optimization and Performance Assessments of an Integrated Energy System Based on Fuel, Wind and Solar Energies». В: *Entropy* 23.4 (2021), с. 431. DOI: 10.3390/e23040431.

[24] Ling-Ling Li та ін. «Improved Tunicate Swarm Algorithm: Solving the Dynamic Economic Emission Dispatch Problems». В: *Applied Soft Computing* 108 (2021), с. 107504. DOI: 10.1016/j.asoc.2021.107504.

[25] Shengchen Li та ін. «Multi-Objective Optimization of Plate-Fin Heat Exchangers via Non-Dominated Sequencing Genetic Algorithm (NSGA-II)». В: *Applied Sciences* 12.22 (2022). DOI: 10.3390/app122211792.

[26] J. J. Liang, B. Y. Qu та P. N. Suganthan. *Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization*. Technical Report. Singapore: Nanyang Technological University, 2014.

[27] Neelesh Maheshwari та ін. «Finite Element Analysis and Multi-criteria Decision-Making (MCDM)-Based Optimal Design Parameter Selection of Solid Ventilated Brake Disc». В: *Journal of The Institution of Engineers (India): Series C* 102.2 (2021), с. 349—359. DOI: 10.1007/s40032-020-00650-y.

[28] Robert M. May. «Simple Mathematical Models with Very Complicated Dynamics». В: *Nature* 261.5560 (1976), с. 459—467. DOI: 10.1038/261459a0.

[29] Seyedali Mirjalili, Seyed Mohammad Mirjalili та Andrew Lewis. «Grey Wolf Optimizer». В: *Advances in Engineering Software* 69 (2014), с. 46—61. DOI: 10.1016/j.advengsoft.2013.12.007.

[30] H. Najafi, B. Najafi та P. Hoseinpoori. «Energy and cost optimization of a plate and fin heat exchanger using genetic algorithm». В: *Applied Thermal Engineering* 31.10 (2011), с. 1839—1847. DOI: 10.1016/j.applthermaleng.2011.02.031.

[31] Santosh Pachpute та Kiran More. «Design Optimization of a Plate-Fin Heat Exchanger With Metaheuristic Hybrid Algorithm». В: *Heat Transfer* 54 (листоп. 2024), с. 1163—1172. DOI: 10.1002/htj.23213.

[32] R. V. Rao та V. K. Patel. «Thermodynamic optimization of cross flow plate-fin heat exchanger using a particle swarm optimization algorithm». В: *International Journal of Thermal Sciences* 49.9 (2010), с. 1712—1721. DOI: 10.1016/j.ijthermalsci.2010.04.001.

[33] S. Sanaye та H. Hajabdollahi. «Thermal-economic multi-objective optimization of plate fin heat exchanger using genetic algorithm». В: *Applied Energy* 87.6 (2010), с. 1893—1902. DOI: 10.1016/j.apenergy.2009.11.016.

[34] R. K. Shah та D. P. Sekulić. *Fundamentals of Heat Exchanger Design*. John Wiley & Sons, 2003. ISBN: 978-0-471-32171-2.

[35] Yuhui Shi та Eberhart RC. «A Modified Particle Swarm Optimizer». В: т. 6. Черв. 1998, с. 69 —73. ISBN: 0-7803-4869-9. DOI: 10.1109/ICEC.1998.699146.

[36] Alex J. Smola та Bernhard Schölkopf. «A Tutorial on Support Vector Regression». В: *Statistics and Computing* 14.3 (2004), с. 199—222. DOI: 10.1023/B:STCO.0000035301.49549.88.

[37] Hamid R. Tizhoosh. «Opposition-Based Learning: A New Scheme for Machine Intelligence». В: *Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation (CIMCA)*. IEEE, 2005, с. 695—701. DOI: 10.1109/CIMCA.2005.1631345.

[38] Wenning Wang та ін. «SLOTSA: A Multi-Strategy Improved Tunicate Swarm Algorithm for Engineering Constrained Optimization Problems». В: *2023 IEEE International Conference on Software Services Engineering (SSE)*. IEEE, 2023, с. 35—42. DOI: 10.1109/sse60056.2023.00016.

[39] Zongyao Wang та ін. «An improved sparrow search algorithm with multi-strategy integration». В: *Scientific Reports* 15 (2025), с. 3314. DOI: 10.1038/s41598-025-86298-z.

[40] David H. Wolpert та William G. Macready. «No Free Lunch Theorems for Optimization». В: *IEEE Transactions on Evolutionary Computation* 1.1 (1997), с. 67—82. DOI: 10.1109/4235.585893.

[41] Jiankai Xue та Bo Shen. «A Novel Swarm Intelligence Optimization Approach: Sparrow Search Algorithm». В: *Systems Science & Control Engineering* 8.1 (2020), с. 22—34. DOI: 10.1080/21642583.2019.1708830.

[42] Jiankai Xue та Bo Shen. «Dung beetle optimizer: a new meta-heuristic algorithm for global optimization». В: *The Journal of Supercomputing* 79.7 (2023), с. 7305—7336. DOI: 10.1007/s11227-022-04959-6.

[43] Xiaohui Yang та ін. «Optimisation and analysis of an integrated energy system with hydrogen supply using solar spectral beam splitting pre-processing». В: *Energy* 287 (2024). DOI: 10.1016/j.energy.2023.129715.

[44] Xin-She Yang. *Nature-Inspired Metaheuristic Algorithms*. 2nd. Frome, UK: Luniver Press, 2010. ISBN: 978-1-905986-28-6.

[45] Xin-She Yang та Amir Hossein Gandomi. «Bat algorithm: a novel approach for global engineering optimization». В: *Engineering Computations* 29.5 (2012), 464–483. ISSN: 0264-4401. DOI: 10.1108/02644401211235834. URL: http://dx.doi.org/10.1108/02644401211235834.

[46] Xiongwen Zhang та ін. «Components sizing of hybrid energy systems via the optimization of power dispatch simulations». В: *Energy* 52 (2013), с. 165—172. DOI: 10.1016/j.energy.2013.01.013.

[47] Rong Zheng та ін. «A Comprehensive Review of the Tunicate Swarm Algorithm: Variations, Applications, and Results». В: *Archives of Computational Methods in Engineering* (2025). DOI: 10.1007/s11831-025-10228-5.

ДОДАТОК А ТЕКСТИ ПРОГРАМ

А.1 Реалізація алгоритму SSA

```
class SSA(Algorithm):
    def run(self, c: FECounter, n_agents: int, seed: int):
        rng = np.random.default\_rng(seed)
    N, D = n_agents, c.dim
    nP = max(1, int(0.2*N)); nS = max(1, int(0.2*N)); ST = 0.8
    X = init_population_uniform(rng, c.lb, c.ub, N)
    f = c.eval_pop(X)
    while not c.exhausted:
        order = np.argsort(f); X = X[order]; f = f[order]
        Xw = X[-1].copy(); fw = f[-1]
        R2 = rng.random()
        for i in range(nP):
            if R2 < ST:
                a = rng.uniform(0,1)
                X[i] = X[i]*math.exp(-(i+1)/(a*c.max_fes/N+1e-12))
            else:
                X[i] = X[i] + rng.standard_normal()*np.ones(D)
                XP = X[0].copy()
        for i in range(nP, N):
            if i > N//2:
                X[i] = rng.standard_normal()*np.exp((Xw-X[i])/((i+1)**2))
            else:
                A = rng.choice([-1.,1.], D)
                X[i] = XP + np.abs(X[i]-XP)*A
        ids = rng.choice(N, nS, replace=False)
        for i in ids:
            b = rng.standard_normal(); K = rng.uniform(-1,1)
            if f[i] > c.best_f:
                X[i] = c.best_x + b*np.abs(X[i]-c.best_x)
            else:
                X[i] = X[i] + K*np.abs(X[i]-Xw)/(f[i]-fw+1e-50)
        X = clip(X, c.lb, c.ub)
        f = c.eval_pop(X)
```

А.2 Реалізація алгоритму DBO

```
class DBO(Algorithm):
    def run(self, c: FECounter, n_agents: int, seed: int):
        rng = np.random.default_rng(seed)
        N, D = n_agents, c.dim
        nR = max(1,int(.2*N)); nB = max(1,int(.2*N))
        nF = max(1,int(.25*N)); nS = N-nR-nB-nF
        X = init_population_uniform(rng, c.lb, c.ub, N)
        Xp = X.copy()
        f = c.eval_pop(X)
        t = 0; Tmax = c.max_fes // N
        while not c.exhausted:
            t += 1; R = max(1e-6, 1 - t/max(1,Tmax))
            wi = int(np.argmax(f)); Xw = X[wi].copy()
            Xb = c.best_x.copy(); Xs = Xb.copy()
            Xn = X.copy()
            for i in range(nR):
                if rng.random() < 0.9:
                    a = rng.choice([-1.,1.]); k = rng.uniform(0,.2)
                    b = rng.uniform(0,1)
                    Xn[i] = X[i] + a*k*Xp[i] + b*np.abs(X[i]-Xw)
```

```
                else:
                    th = rng.uniform(0,math.pi)
                    Xn[i] = X[i] + math.tan(th)*np.abs(X[i]-Xp[i])
            Lbs = np.maximum(Xs*(1-R), c.lb); Ubs = np.minimum(Xs*(1+R), c.ub)
            for i in range(nR, nR+nB):
                Xn[i] = Xs + rng.random(D)*(X[i]-Lbs) +
    rng.random(D)*(X[i]-Ubs)
            Lbb = np.maximum(Xb*(1-R), c.lb); Ubb = np.minimum(Xb*(1+R), c.ub)
            for i in range(nR+nB, nR+nB+nF):
                Xn[i] = X[i] + rng.standard_normal(D)*(X[i]-Lbb) +
    rng.random(D)*(X[i]-Ubb)
            for i in range(nR+nB+nF, N):
                Xn[i] = Xb +
    0.5*rng.standard_normal(D)*(np.abs(X[i]-Xs)+np.abs(X[i]-Xb))
            Xn = clip(Xn, c.lb, c.ub)
            fn = c.eval_pop(Xn)
            if c.exhausted: break
            imp = fn < f; Xp = X.copy(); X[imp] = Xn[imp]; f[imp] = fn[imp]
```

A.3 Реалізація алгоритму DPDBO

```
class DPDBO(Algorithm):
    @staticmethod
    def _pwlcm_init(rng, lb, ub, N, D, p=0.4):
        Z = np.empty((N, D))
        f0 = rng.random(D)
        for i in range(N):
            for j in range(D):
                v = f0[j]
                if 0 <= v < p:        v = v/p
                elif v < 0.5:         v = (v-p)/(0.5-p)
                elif v < 1-p:         v = (1-p-v)/(0.5-p)
                else:                 v = (1-v)/p
                f0[j] = v
            Z[i] = f0.copy()
        return lb + Z*(ub - lb)
    def run(self, c: FECounter, n_agents: int, seed: int):
        rng = np.random.default_rng(seed)
        N, D = n_agents, c.dim
        Tmax = max(1, c.max_fes // N)
        X = self._pwlcm_init(rng, c.lb, c.ub, N, D)
        Xp = X.copy()
        f = c.eval_pop(X)
        pTS = int(round(N * 0.55))
        nB = max(1, int(.2*N)); nF = max(1, int(.25*N))
        Q1, Q2, adj_c = 0.8, 0.2, 1.25
        w1, w2 = pTS*Q1, pTS*Q2
        t = 0
        while not c.exhausted:
            t += 1; R = max(1e-6, 1 - t/Tmax)
            w = w1 + adj_c - (w1-w2)/(0.75 + math.exp(-2*(2*t -
    Tmax)/max(1,Tmax)))
            nR = max(1, min(pTS-1, int(round(w))))
            nS = pTS - nR
            wi = int(np.argmax(f)); Xw = X[wi].copy()
            Xb = c.best_x.copy(); Xs = Xb.copy()
            Xn = X.copy()
            for i in range(nR):
                alpha = rng.random(); delta = rng.random()
                if alpha < 0.7:
                    if delta < 0.9:
                        a = rng.choice([-1.,1.]); k = rng.uniform(0,.2)
                        b = rng.uniform(0,1)
```

```
                        Xn[i] = X[i] + a*k*Xp[i] + b*np.abs(X[i]-Xw)
                    else:
                        th = rng.uniform(0, math.pi)
                        Xn[i] = X[i] + math.tan(th)*np.abs(X[i]-Xp[i])
                else:
                    b_sp = 0.3; l = rng.uniform(-1, 1)
                    a = rng.choice([-1.,1.]); k = rng.uniform(0,.2)
                    Xn[i] = X[i] +
    b_sp*l*math.cos(2*math.pi*l)*np.abs(X[i]-Xw) + a*k*Xp[i]
            Lbs = np.maximum(Xs*(1-R), c.lb); Ubs = np.minimum(Xs*(1+R), c.ub)
            for i in range(nR, nR+nB):
                Xn[i] = Xs + rng.random(D)*(X[i]-Lbs) +
    rng.random(D)*(X[i]-Ubs)
            Lbb = np.maximum(Xb*(1-R), c.lb); Ubb = np.minimum(Xb*(1+R), c.ub)
            for i in range(nR+nB, nR+nB+nF):
                Xn[i] = X[i] + rng.standard_normal(D)*(X[i]-Lbb) +
    rng.random(D)*(X[i]-Ubb)
            for i in range(nR+nB+nF, nR+nB+nF+nS):
                if i >= N: break
                Xn[i] = Xb +
    0.5*rng.standard_normal(D)*(np.abs(X[i]-Xs)+np.abs(X[i]-Xb))
            for i in range(nR+nB+nF+nS, N):
                Xn[i] = X[i] + rng.standard_normal(D)*(X[i]-Lbb) +
    rng.random(D)*(X[i]-Ubb)
            Xn = clip(Xn, c.lb, c.ub)
            fn = c.eval_pop(Xn)
            if c.exhausted: break
            imp = fn < f; Xp = X.copy(); X[imp] = Xn[imp]; f[imp] = fn[imp]
```

A.4 Реалізація алгоритму TSA

```
class TSA(Algorithm):
    def run(self, c: FECounter, n_agents: int, seed: int):
        rng = np.random.default_rng(seed)
        N, D = n_agents, c.dim
        Pmin, Pmax = 1, 4
        X = init_population_uniform(rng, c.lb, c.ub, N)
        f = c.eval_pop(X)
        while not c.exhausted:
            FS = c.best_x
            M = max(1, int(Pmin + rng.random() * (Pmax - Pmin)))
            C1 = rng.random((N, D))
            C2 = rng.random((N, D))
            C3 = rng.random((N, D))
            A = (C2 + C3 - 2.0 * C1) / M
            PD = np.abs(FS - C2 * X)
            T1 = FS + A * PD
            T2 = FS - A * PD
            Pos = np.where(C3 >= 0.5, T1, T2)
            X_new = np.empty_like(X)
            X_new[0] = Pos[0]
            for i in range(1, N):
                X_new[i] = (Pos[i] + X_new[i - 1]) * 0.5
            X = clip(X_new, c.lb, c.ub)
            f = c.eval_pop(X)
```

A.5 Реалізація алгоритму SBOA

```
class SBOA(Algorithm):
    def run(self, c: FECounter, n_agents: int, seed: int):
        rng = np.random.default_rng(seed)
        N, D = n_agents, c.dim
        Tmax = max(1, c.max_fes // (2*N))  # 2N evals per iter
        X = init_population_uniform(rng, c.lb, c.ub, N)
        f = c.eval_pop(X)
        t = 0
        while not c.exhausted:
            t += 1
            for i in range(N):
                if c.exhausted: return
                if t < Tmax/3:
                    idx = rng.choice(N, 2, replace=False)
                    Xn = X[i] + (X[idx[0]]-X[idx[1]])*rng.random(D)
                elif t < 2*Tmax/3:
                    RB = rng.standard_normal(D)
                    Xn = c.best_x +
    math.exp((t/Tmax)**4)*(RB-.5)*(c.best_x-X[i])
                else:
                    RL = 0.5*levy_flight(rng, D)
                    base = max(0, 1-t/Tmax)
                    Xn = c.best_x + (base**(2*t/Tmax))*X[i]*RL
                Xn = clip(Xn, c.lb, c.ub); fn = c(Xn)
                if fn < f[i]: X[i]=Xn; f[i]=fn
            for i in range(N):
                if c.exhausted: return
                if rng.random() < 0.5:
                    RB = rng.standard_normal(D)
                    Xn = c.best_x + (2*RB-1)*(1-t/Tmax)**2*X[i]
                else:
                    K = rng.choice([1,2]); R2 = rng.standard_normal(D)
                    Xn = X[i] + R2*(X[int(rng.integers(N))]-K*X[i])
                Xn = clip(Xn, c.lb, c.ub); fn = c(Xn)
                if fn < f[i]: X[i]=Xn; f[i]=fn
```

A.6 Реалізація алгоритму GA

```
class GA(Algorithm):
    def run(self, c: FECounter, n_agents: int, seed: int,
            pc: float = 0.9, pm: float = None, eta_c: float = 20, eta_m: float
    = 20):
        rng = np.random.default_rng(seed)
        N, D = n_agents, c.dim
        if pm is None:
            pm = 1.0 / D
        X = init_population_uniform(rng, c.lb, c.ub, N)
        f = c.eval_pop(X)
        while not c.exhausted:
            parents = np.empty_like(X)
            for i in range(N):
                a, b = rng.choice(N, 2, replace=False)
                parents[i] = X[a] if f[a] < f[b] else X[b]
            offspring = parents.copy()
            for i in range(0, N - 1, 2):
                if rng.random() < pc:
                    p1, p2 = parents[i], parents[i + 1]
```

```
                    ch1, ch2 = p1.copy(), p2.copy()
                    for j in range(D):
                        if rng.random() < 0.5 and abs(p1[j] - p2[j]) > 1e-14:
                            u = rng.random()
                            if u <= 0.5:
                                beta_q = (2.0 * u) ** (1.0 / (eta_c + 1.0))
                            else:
                                beta_q = (1.0 / (2.0 * (1.0 - u))) ** (1.0 /
    (eta_c + 1.0))
                            ch1[j] = 0.5 * ((1 + beta_q) * p1[j] + (1 -
    beta_q) * p2[j])
                            ch2[j] = 0.5 * ((1 - beta_q) * p1[j] + (1 +
    beta_q) * p2[j])
                    offspring[i], offspring[i + 1] = ch1, ch2
            for i in range(N):
                for j in range(D):
                    if rng.random() < pm:
                        u = rng.random()
                        if u < 0.5:
                            delta_q = (2.0 * u) ** (1.0 / (eta_m + 1.0)) - 1.0
                        else:
                            delta_q = 1.0 - (2.0 * (1.0 - u)) ** (1.0 / (eta_m
    + 1.0))
                        offspring[i, j] += delta_q * (c.ub[j] - c.lb[j])
            offspring = clip(offspring, c.lb, c.ub)
            f_off = c.eval_pop(offspring)
            if c.exhausted:
                break
            all_X = np.vstack([X, offspring])
            all_f = np.hstack([f, f_off])
            idx = np.argsort(all_f)[:N]
            X = all_X[idx]
            f = all_f[idx]
```

A.7 Реалізація алгоритму PSO

```
class PSO(Algorithm):
    def run(self, c: FECounter, n_agents: int, seed: int,
            w_max: float = 0.9, w_min: float = 0.4, c1: float = 2.0, c2: float
    = 2.0):
        rng = np.random.default_rng(seed)
        N, D = n_agents, c.dim
        Tmax = max(1, c.max_fes // N)
        X = init_population_uniform(rng, c.lb, c.ub, N)
        V = rng.uniform(-1,1,(N,D)) * (c.ub - c.lb) * 0.1
        f = c.eval_pop(X)
        pBest = X.copy(); pBest_f = f.copy()
        t = 0
        while not c.exhausted:
            t += 1; w = w_max - (w_max - w_min)*t/Tmax
            r1 = rng.random((N,D)); r2 = rng.random((N,D))
            V = w*V + c1*r1*(pBest - X) + c2*r2*(c.best_x - X)
            Vmax = 0.2*(c.ub - c.lb)
            V = np.clip(V, -Vmax, Vmax)
            X = clip(X + V, c.lb, c.ub)
            f = c.eval_pop(X)
            if c.exhausted: break
            imp = f < pBest_f; pBest[imp] = X[imp]; pBest_f[imp] = f[imp]
```

ДОДАТОК Б РЕЗУЛЬТАТИ СТАТИСТИЧНОГО ТЕСТУВАННЯ

Таблиця Б.1 – Детальні результати тесту Фрідмана для всіх інженерних задач

Алгоритм	CarSideImpact		PressureVessel		SpeedReducer		WeldedBeam	
	Ранг	Поз.	Ранг	Поз.	Ранг	Поз.	Ранг	Поз.
SBOA	1.367	1	2.367	2	1.500	1	2.200	2
DPDBO	4.067	4	2.150	1	2.283	2	4.300	4
DBO	4.100	5	3.067	3	3.167	3	3.733	3
PSO	3.200	2	4.650	5	5.950	6	1.333	1
SSA	3.867	3	6.433	7	4.933	5	5.167	6
TSA	5.033	6	3.633	4	6.733	7	5.100	5
GA	6.367	7	5.700	6	3.433	4	6.167	7
Статистика тесту:								
CarSideImpact: $\chi^2 = 91.771$, $p = 1.30e-17$			PressureVessel: $\chi^2 = 105.857$, $p = 1.50e-20$					
SpeedReducer: $\chi^2 = 146.516$, $p = 4.22e-29$			WeldedBeam: $\chi^2 = 114.286$, $p = 2.58e-22$					

Таблиця Б.2 – Попарне порівняння алгоритмів (тест Вілкоксона, p-значення) для всіх задач ($\alpha = 0{,}05$)

Пара алгоритмів	CarSideImpact	PressureVessel	SpeedReducer	WeldedBeam
SSA vs DBO	0.7562	**2.49e-08**	**5.10e-05**	**9.27e-03**
SSA vs DPDBO	0.5946	**3.64e-10**	**2.87e-11**	**7.79e-03**
SSA vs TSA	**3.06e-05**	**2.49e-10**	**2.87e-11**	0.8825
SSA vs SBOA	**2.87e-11**	**3.18e-11**	**2.87e-11**	**3.51e-11**
SSA vs GA	**1.15e-08**	**7.13e-03**	**2.87e-11**	**3.88e-04**
SSA vs PSO	0.7675	**5.10e-05**	**3.21e-08**	**3.64e-10**
DBO vs DPDBO	0.8245	**0.0436**	0.0798	0.2871
DBO vs TSA	**0.0321**	0.0528	**5.23e-11**	**0.0266**
DBO vs SBOA	**7.03e-11**	0.4161	**1.63e-04**	**6.82e-03**
DBO vs GA	**1.29e-05**	**2.78e-06**	0.8016	**3.13e-07**
DBO vs PSO	0.1354	**2.82e-03**	**1.85e-08**	**4.13e-08**
DPDBO vs TSA	**0.0385**	**8.93e-05**	**2.87e-11**	**0.0219**
DPDBO vs SBOA	**1.15e-10**	0.0604	**5.12e-04**	**2.29e-08**
DPDBO vs GA	**4.91e-06**	**6.81e-09**	**5.39e-07**	**1.24e-06**
DPDBO vs PSO	0.1071	**2.13e-05**	**3.66e-09**	**1.48e-09**
TSA vs SBOA	**2.87e-11**	**3.80e-08**	**2.87e-11**	**2.87e-11**
TSA vs GA	**2.07e-04**	**3.80e-08**	**2.87e-11**	**9.27e-04**
TSA vs PSO	**2.96e-03**	**9.27e-04**	0.0266	**5.32e-10**
SBOA vs GA	**2.87e-11**	**1.04e-10**	**2.87e-11**	**3.51e-11**
SBOA vs PSO	0.0690	**1.29e-05**	**1.12e-09**	**1.79e-06**
GA vs PSO	**6.28e-07**	0.0690	**8.12e-09**	**7.03e-11**