

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕКОНОМІКИ І МЕНЕДЖМЕНТУ
Кафедра кібернетики та інформатики**

КВАЛІФІКАЦІЙНА РОБОТА

за другим (магістерським) рівнем вищої освіти

**на тему: Інформаційна технологія інтелектуального планування маршруту
безпілотного літального апарату**

Виконав:

Доля Євген Віталійович
здобувач вищої освіти
спеціальності 126 «Інформаційні
системи та технології»

Науковий керівник:

Шелехов Ігор Володимирович
кандидат технічних наук, доцент
кафедри кібернетики та інформатики

Рецензент:

Москаленко Альона Сергіївна
кандидат технічних наук, доцент
кафедри комп'ютерних наук
Сумського державного університету

Суми – 2025

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ**

Факультет економіки і менеджменту
Кафедра кібернетики та інформатики
Ступінь вищої освіти – другий (магістерський)
Спеціальність – 126 «Інформаційні системи і технології»

ЗАТВЕРДЖУЮ:

завідувачка кафедри
кібернетики та інформатики
АГАДЖАНОВА Світлана
25 листопада 2024 р.

ЗАВДАННЯ

на кваліфікаційну роботу здобувачу вищої освіти

Євгену Віталійовичу Долі

1. Тема кваліфікаційної роботи: «Інформаційна технологія інтелектуального планування маршруту безпілотного літального апарату»
2. База практичного дослідження: ФОП «Доля Євген Віталійович», м. Суми
3. Керівник кваліфікаційної роботи: Шелехов Ігор Володимирович, доцент кафедри кібернетики та інформатики, к.т.н., доцент, затверджений наказом по університету № 3873/ос від 25.11.2024 р.
4. Строки подання здобувачем кваліфікаційної роботи: до 03.12.2025 р.
5. Вихідні дані до кваліфікаційної роботи –
6. Зміст розрахунково-пояснювальної записки: 1) Аналіз предметної області та постановка задачі дослідження; 2) Вибір методів дослідження, моделювання та проектування інформаційної технології; 3) Розробка та реалізація інформаційної системи; 4) Аналіз результатів дослідження, оформлення розрахунково-пояснювальної записки до кваліфікаційної роботи

Завдання прийняв до виконання _____

Євген ДОЛЯ

Дата отримання завдання: 25 листопада 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/П	Назва етапів кваліфікаційної роботи	Строки виконання етапів	Примітка
1	Аналіз предметної області та постановка задачі дослідження	травень 2025 року	
2	Вибір методів дослідження, моделювання та проектування інформаційної технології	червень-липень 2025 року	
3	Розробка та тестування інформаційної системи	серпень-вересень 2025 року	
4	Аналіз результатів дослідження	жовтень 2025 року	
5	Оформлення розрахунково-пояснювальної записки до кваліфікаційної роботи, та її подання на попередній захист та рецензування	до 19 листопада 2025 року	
6	Перевірка кваліфікаційної роботи на унікальність	до 03 грудня 2025 року	
7	Подання закінченої кваліфікаційної роботи та документів до неї до деканату	до 08 грудня 2025 року	
8	Захист кваліфікаційної роботи	до 03 грудня 2025 року	

Керівник роботи

Ігор ШЕЛЕХОВ

Здобувач

Євген ДОЛЯ

АНОТАЦІЯ

Долі Євгена Віталійовича, здобувача «Інформаційна технологія інтелектуального планування маршруту безпілотного літального апарату»

Кваліфікаційна робота на здобуття ступеня магістра з інформаційних систем і технологій за освітньою програмою «Інформаційні системи та технології» зі спеціальності 126 (F6) «Інформаційні системи і технології». Сумський національний аграрний університет, Суми, 2025.

У кваліфікаційній роботі Долі Євгена досліджено теоретичні, методологічні та практичні аспекти розроблення інформаційної технології інтелектуального планування маршруту безпілотного літального апарату в умовах множинних обмежень. Основну увагу зосереджено на методах комбінаторної оптимізації та інтелектуальних алгоритмах пошуку, зокрема на використанні генетичних алгоритмів для розв'язання задачі комівояжера, адаптованої до специфіки планування маршрутів БПЛА.

У роботі сформовано математичну модель задачі планування маршруту, розроблено та адаптовано генетичний алгоритм з урахуванням експлуатаційних обмежень безпілотного літального апарату, а також реалізовано інформаційно-аналітичну систему, здатну автоматизувати процес пошуку оптимального або близького до оптимального маршруту. Проведено експериментальне тестування розробленої інформаційної технології з використанням просторових даних, результати якого підтверджують ефективність запропонованого підходу.

Отримані результати демонструють можливість підвищення ефективності планування маршрутів безпілотних літальних апаратів шляхом застосування інтелектуальних методів оптимізації та еволюційних алгоритмів. Розроблена інформаційна технологія може бути використана в системах підтримки прийняття рішень, програмних комплексах планування польотів БПЛА, а також у задачах моніторингу та аналізу територій.

Ключові слова: безпілотний літальний апарат, інформаційна технологія, планування маршруту, задача комівояжера, генетичний алгоритм, інтелектуальна оптимізація, геопросторові дані.

ABSTRACT

Yevhen Dolia Vitaliyovych of the applicant « Information Technology for Intelligent Route Planning of an Unmanned Aerial Vehicle ».

Qualification thesis for obtaining the Master's degree in Information Systems and Technologies under the educational program "Information Systems and Technologies," specialty 126 (F6) "Information Systems and Technologies." Sumy National Agrarian University, Sumy, 2025.

The qualification thesis by Yevhen Dolia investigates the theoretical, methodological, and practical aspects of developing an information technology for intelligent route planning of an unmanned aerial vehicle under multiple operational constraints. The main focus is placed on combinatorial optimization methods and intelligent search algorithms, in particular on the application of genetic algorithms to solve the Traveling Salesman Problem adapted to the specific requirements of UAV route planning.

The thesis formulates a mathematical model of the route planning problem, develops and adapts a genetic algorithm considering the operational limitations of an unmanned aerial vehicle, and implements an information-analytical system capable of automating the search for optimal or near-optimal routes. Experimental testing of the proposed information technology using spatial data has been carried out, and the results confirm the effectiveness of the proposed approach.

The obtained results demonstrate the potential to improve the efficiency of unmanned aerial vehicle route planning through the use of intelligent optimization methods and evolutionary algorithms. The developed information technology can be applied in decision support systems, UAV flight planning software, as well as in territory monitoring and spatial analysis tasks.

Keywords: unmanned aerial vehicle, information technology, route planning, traveling salesman problem, genetic algorithm, intelligent optimization, geospatial data..

ЗМІСТ

ВСТУП.....	7
РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ	11
1.1 Сучасний стан та тенденції розвитку безпілотних літальних апаратів.....	11
1.2. Структура системи керування безпілотними літальними апаратами.....	13
1.3 Сучасні алгоритми керування та навігації безпілотних літальних апаратів .	15
1.4 Огляд сучасних платформ для програмної реалізації систем керування та навігації безпілотних літальних апаратів	18
1.5 Постановка задачі.....	20
РОЗДІЛ 2 ВИБІР МЕТОДІВ ДОСЛІДЖЕННЯ, МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ.....	22
2.1 Основні поняття та визначення генетичних алгоритмів.....	22
2.2 Стандартні операції схрещування в генетичних алгоритмах.....	23
2.3 Стратегії відбору нового покоління у генетичних алгоритмах.....	25
2.4 Мутаційні оператори та їх вплив на різноманітність популяції	27
2.5 Повна схема роботи генетичного алгоритму	30
2.6 Вибір параметрів генетичного алгоритму та їх вплив на ефективність.....	33
2.7 Моделі генетичних алгоритмів.....	34
2.8 Генетичний алгоритм для побудови оптимального маршруту БПЛА у тривимірному просторі.....	37
РОЗДІЛ 3 РОЗРОБКА ТА ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ.....	42
3.1 Вибір середовища для програмної реалізації.....	42
3.2 Короткий опис програмної реалізації	43
3.3 Генерація вхідних даних	48
3.4 Результати тестування	51
ВИСНОВКИ.....	56
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	58
ДОДАТКИ.....	61
ДОДАТОК А.....	62

ДОДАТОК Б	69
ДОДАТОК В	74
ДОДАТОК Г	81

ВСТУП

Актуальність теми.

Сучасні безпілотні літальні апарати (БПЛА) широко застосовуються у сфері моніторингу, картографування, логістики, аграрних технологій, інспекції інфраструктурних об'єктів та в операціях реагування на надзвичайні ситуації. Ефективність виконання таких завдань значною мірою залежить від якості побудови оптимального маршруту польоту, який враховує обмеження по енергії, рельєф місцевості, метеоумови, заборонені зони та формальні вимоги до часу місії. Класичні методи маршрутизації часто виявляються недостатньо продуктивними для складних динамічних середовищ, тоді як використання методів штучного інтелекту, зокрема генетичних алгоритмів, дозволяє ефективно розв'язувати NP-складні задачі типу комівояжера та знаходити рішення, наближені до оптимальних. Тому розробка інформаційної технології інтелектуального планування маршруту БПЛА є актуальним науковим та практичним завданням.

Аналіз стану наукової розробки проблеми. Проблематика інтелектуального планування маршрутів БПЛА посідає значне місце в сучасних дослідженнях у сфері робототехніки, штучного інтелекту та обробки геопросторових даних. Одним із базових підходів до формалізації задач маршрутної оптимізації є зведення її до задачі комівояжера (TSP), яка належить до класу NP-складних і активно досліджується в контексті автономних систем. У науковій літературі описано широкий спектр класичних методів розв'язання TSP: динамічне програмування, методи гілок і меж, евристики «найближчого сусіда», локальний пошук, алгоритм Ліна–Кернігана тощо. Проте для складних умов експлуатації БПЛА ці методи часто виявляються недостатньо гнучкими або потребують значних обчислювальних ресурсів.

Тому особливу увагу дослідників привертають методи еволюційних обчислень, зокрема генетичні алгоритми (ГА). У сучасних робототехнічних дослідженнях ГА застосовуються для оптимізації траєкторій рухомих роботів, мінімізації витрат енергії, обходу перешкод та багатокритеріальної оптимізації. Ряд

праць демонструє їх високу ефективність для задач, де пошуковий простір великий, а вимоги до часу обчислень є гнучкими. Існують модифікації ГА, орієнтовані на маршрутизацію БПЛА, що враховують специфічні фактори: висоту польоту, обмеження акумулятора, полігони заборонених зон, необхідність дотримання пріоритетності точок тощо.

Разом із тим, більшість наявних робіт спрямовані або на класичний TSP, або на окремі аспекти маршрутизації БПЛА (наприклад, уникнення перешкод або оптимізацію енергоспоживання). Комплексні інформаційні технології, що інтегрують модель комівояжера, адаптовані генетичні алгоритми та модуль обробки геопросторових даних, досліджені недостатньо. Залишається відкритим питання підвищення стабільності та швидкодії ГА при збільшенні кількості точок маршруту та обмежень, а також адаптивного налаштування параметрів алгоритму під особливості польотного завдання.

Отже, незважаючи на значну кількість робіт, існує наукова ніша, пов'язана з розробкою цілісної інформаційної технології інтелектуального планування маршруту БПЛА з використанням генетичних алгоритмів, здатної працювати в умовах реальних обмежень та сценаріїв застосування.

Мета і завдання дослідження. Мета роботи полягає у розробці моделей, методів та засобів інформаційної технології інтелектуального планування маршруту БПЛА на основі зведення задачі до задачі комівояжера та застосування генетичних алгоритмів для пошуку оптимального або близького до оптимального маршруту з урахуванням заданих обмежень.

Відповідно до поставленої мети сформульовані такі задачі:

- Проаналізувати існуючі методи маршрутизації БПЛА та підходи до розв'язання задачі комівояжера.
- Розробити математичну модель задачі інтелектуального планування маршруту БПЛА.
- Сформувати модель комівояжера для представлення маршруту відповідно до умов польоту.

- Розробити або адаптувати генетичний алгоритм для пошуку оптимальної траєкторії з урахуванням обмежень.
- Створити програмну реалізацію інформаційної системи здатної до інтелектуального планування маршруту безпілотного літального апарату.
- Провести перевірку працездатності розробленої інформаційної системи.

Об'єктом дослідження є процес планування маршруту безпілотного літального апарату в умовах множинних обмежень та вимог до оптимальності.

Предметом дослідження є моделі, методи та інформаційні технології інтелектуальної оптимізації маршруту БПЛА на основі задачі комівояжера та генетичних алгоритмів

Методи досліджень є методи математичного моделювання та формалізації задач оптимізації, теорія графів і комбінаторна оптимізація, генетичні алгоритми та еволюційні обчислення, алгоритми обробки геопросторових даних, методи порівняльного аналізу ефективності алгоритмів, програмні технології моделювання і симуляції траєкторій БПЛА.

Елементи наукової новизни є розроблено інформаційну технологію інтелектуального планування маршруту БПЛА, що поєднує модель задачі комівояжера з адаптованим генетичним алгоритмом, оптимізованим під специфіку польотного середовища; запропоновано модифікацію генетичного алгоритму з урахуванням реальних експлуатаційних обмежень БПЛА; створено модель оцінювання ефективності побудованих маршрутів, що дозволяє порівнювати різні конфігурації алгоритму.

Практична значимість одержаних результатів Розроблена інформаційна технологія може бути використана для автоматизованого планування маршрутів БПЛА у задачах моніторингу територій, інспекції інфраструктури, картографування, логістичної доставки та агроскаутингу. Програмна реалізація дозволяє скоротити час підготовки польотних місій, підвищити їх енергоефективність та безпеку, а також забезпечує можливість адаптації до різних типів безпілотних платформ і сценаріїв роботи. Отримані результати можуть бути

інтегровані у системи підтримки прийняття рішень оператора БПЛА, а також використані у науково-дослідних і навчальних цілях.

Апробація результатів кваліфікаційної роботи. Результати кваліфікаційної роботи апробовані на міжнародній науковій конференції молодих учених «Інформатика, математика, автоматика : 2025» (21–25 квітня 2025 року), (Суми – Астана) на тему:

- 1) «Інтелектуальне планування маршруту безпілотного літального апарата» та Всеукраїнської наукової конференції студентів і аспірантів, присвяченої Міжнародному дню студента (17-21 листопада 2025 року), (Суми) на тему
- 2) «Еволюційна оптимізація траєкторії польоту безпілотного літального апарата»

Публікації. За матеріалами кваліфікаційної роботи опубліковано 2 тези наукових конференцій.

Структура і обсяг роботи. Кваліфікаційна робота складається із вступу, трьох розділів, висновків, списку використаних джерел з 21 найменувань, додатків. Основний текст викладений на 87 сторінках комп'ютерного тексту, робота містить 6 рисунків, 6 таблиць, 4 додатки.

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Сучасний стан та тенденції розвитку безпілотних літальних апаратів

Безпілотні літальні апарати (БПЛА) за останні десятиліття перетворилися на один із ключових елементів сучасних авіаційних та інформаційно-технологічних систем. Стрімкий розвиток електроніки, сенсорних технологій, бездротового зв'язку та штучного інтелекту сприяв формуванню нового покоління автономних платформ, здатних виконувати широкий спектр завдань у цивільних, промислових, наукових та оборонних застосуваннях. Ринок БПЛА демонструє стабільно високу динаміку зростання, що зумовлено доступністю апаратів, зниженням вартості комплектуючих та розвитком технологій автономного керування.

Сучасні БПЛА класифікуються за конструкцією, призначенням, масою та тривалістю польоту. Найпоширенішими є квадрокоптери, гексакоптери, літальні апарати з фіксованим крилом, гібридні VTOL-платформи та мікро-БПЛА. Квадрокоптери переважають у цивільних застосуваннях завдяки маневровості та простоті експлуатації, тоді як апарати з фіксованим крилом забезпечують більшу дальність та енергоефективність для моніторингових операцій. Окремим напрямом розвитку є ройові системи (swarm UAV), у яких група малих апаратів виконує завдання колективно, що відкриває можливості для більшої стійкості та масштабованості операцій[1-2].

Функціональні можливості сучасних БПЛА суттєво розширюються завдяки використанню комп'ютерного зору, машинного навчання, систем глобального позиціонування, інерціальних навігаційних модулів та лазерних або радарних далекомірів. Інтеграція штучного інтелекту дозволяє реалізовувати автономне ухилення від перешкод, адаптивне управління, обробку відеопотоків у реальному часі та оптимізацію маршруту польоту. Такі можливості є критично важливими для застосувань у галузях картографування, інфраструктурних інспекцій,

агромоніторингу, пошуково-рятувальних операцій, логістики та екологічного нагляду.

Однією з ключових тенденцій розвитку є повна або часткова автономізація польотних місій. Якщо перші покоління БПЛА потребували постійного ручного керування, то сучасні системи здатні самостійно планувати траєкторії, оцінювати стан навколишнього середовища та адаптувати поведінку до непередбачуваних змін. Це зумовлює зростання ролі інформаційних технологій, які забезпечують інтелектуальне планування маршрутів, оптимізацію енерговитрат та безпечне виконання польотних завдань.

Ще однією важливою тенденцією є мініатюризація компонентів і одночасне збільшення функціональних можливостей. Завдяки розвитку елементної бази з'являються надлегкі сенсори, високопродуктивні одноплатні комп'ютери та енергоефективні приводи, що дозволяє створювати компактні апарати із значною обчислювальною спроможністю. Це робить БПЛА більш доступними, придатними для роботи у складних умовах та здатними виконувати локальні операції з мінімальною участю оператора.

Важливою сферою досліджень є також підвищення надійності та безпеки роботи БПЛА, включно з удосконаленням алгоритмів стабілізації польоту, систем уникнення зіткнень і механізмів резервування. Значну увагу приділено розвитку комунікаційних технологій, зокрема мережевого управління у діапазонах LTE/5G, що дозволяє збільшити дальність керування та підвищити якість передачі телеметрії. На глобальному рівні триває стандартизація протоколів та регуляцій, спрямованих на інтеграцію БПЛА у повітряний простір разом із пілотованою авіацією (UTM - Unmanned Traffic Management)[3].

Таким чином, сучасний стан розвитку БПЛА характеризується високим рівнем технологічної зрілості, широким спектром застосувань та швидкими темпами інновацій. Тенденції автономізації, інтелектуалізації, мініатюризації та інтеграції штучного інтелекту формують нові вимоги до систем планування маршруту, які мають бути здатні ефективно функціонувати у складних умовах та забезпечувати високу якість виконання польотних завдань.

1.2. Структура системи керування безпілотними літальними апаратами

Системи керування безпілотними літальними апаратами (БПЛА) є комплексом технічних та програмних засобів, що забезпечують стабільність польоту, навігацію, виконання маневрів, передавання телеметрії та взаємодію з оператором або наземною станцією. Ефективність роботи БПЛА значною мірою залежить від узгодженої взаємодії всіх компонентів системи керування, їх надійності та здатності працювати в умовах динамічно змінюваного середовища. Структура таких систем є багаторівневою та включає низку інтегрованих модулів.

У загальному вигляді система керування БПЛА складається з чотирьох основних підсистем: навігаційної, стабілізаційної, виконавчої (актуаторної) та комунікаційно-інформаційної. Кожна з них виконує специфічні функції, які разом забезпечують повноцінну автономну або напівавтономну роботу апарата [1-3].

Навігаційна підсистема відповідає за визначення просторового положення та параметрів руху БПЛА. До її складу входять приймачі глобальних навігаційних систем (GPS/GNSS), інерціальні вимірювальні модулі (IMU), магнітометри, барометри, лазерні далекоміри та інші датчики. Інтеграція даних здійснюється за допомогою фільтрів Калмана чи інших методів сенсорного злиття, що дозволяє забезпечити точне позиціонування навіть у складних умовах, таких як зашумленість сигналів чи часткова втрата супутникового зв'язку.

Стабілізаційна підсистема забезпечує підтримання заданих параметрів орієнтації, висоти та швидкості. В її основі - автопілот, що реалізує алгоритми керування у замкненому контурі: PID-регулятори, адаптивні та робастні методи керування, а також алгоритми виявлення та компенсації зовнішніх збурень. Стабілізаційна підсистема відіграє ключову роль у забезпеченні безпечного польоту, оскільки саме вона відповідає за реакцію апарата на пориви вітру, зміни навантаження або відхилення від курсу.

Виконавча підсистема включає приводні механізми (електромотори, серводвигуни, гвинтові установки та інші актуатори), які фізично реалізують команди керування. Ефективність роботи цієї підсистеми залежить від швидкодії

приводів, точності реагування та енергоефективності, що безпосередньо впливає на тривалість польоту та маневровість БПЛА.

Комунікаційно-інформаційна підсистема забезпечує передавання даних між БПЛА та наземною станцією, а також обмін інформацією всередині системи. До її складу належать модулі радіозв'язку, телеметрії, відеопередачі та обробки даних. Сучасні БПЛА можуть використовувати різні канали зв'язку - від традиційних радіомодулів до LTE/5G-мереж, що дозволяє реалізовувати операції на великих відстанях та забезпечувати стабільну передачу високобітових потоків даних. У цю підсистему також входять обчислювальні модулі, що реалізують алгоритми маршрутизації, аналізу навколишнього середовища та прийняття рішень.

Окремим компонентом структури системи керування є бортовий програмно-алгоритмічний комплекс [4], що включає модулі планування маршруту, уникнення перешкод, діагностики стану систем та обробки сенсорних даних. Залежно від рівня автономності БПЛА ці модулі можуть функціонувати як частина наземної станції, так і повністю на борту апарата. У перспективних системах важливу роль відіграють алгоритми штучного інтелекту, які дозволяють аналізувати складні ситуації, оптимізувати траєкторії польоту та підвищувати точність навігації.

Важливим елементом є також система забезпечення безпеки [5], що включає режими аварійного зниження, повернення до точки старту (Return to Home), автоматичне припинення польоту у разі втрати зв'язку та інші механізми, спрямовані на мінімізацію ризиків. Застосування цих функцій є обов'язковим для сучасних БПЛА, інтегрованих у повітряний простір з іншими авіаційними засобами.

Таким чином, структура систем керування БПЛА є комплексною і включає взаємодію апаратного та програмного забезпечення на різних рівнях. Її розвиток спрямований на підвищення автономності, надійності, енергоефективності та інтеграції інтелектуальних алгоритмів, що створює основу для ефективного виконання складних польотних місій, зокрема задач маршрутного планування.

1.3 Сучасні алгоритми керування та навігації безпілотних літальних апаратів

Системи керування та навігації безпілотних літальних апаратів є ключовими елементами, що визначають рівень автономності, точність виконання польотних місій, безпеку та ефективність використання повітряної платформи. Сучасні БПЛА функціонують як складні кіберфізичні системи, у яких алгоритмічні рішення відіграють не менш важливу роль, ніж механічні або електронні компоненти. У зв'язку з цим структура алгоритмів керування та навігації включає декілька взаємопов'язаних напрямів - стабілізацію, локалізацію, сенсорну обробку, планування руху, уникнення перешкод, оптимізацію траєкторій та адаптацію до змінних умов середовища.

На базовому рівні керування динамікою БПЛА ґрунтується на математичних моделях, що описують рух апарата як нелінійної багатовимірної системи. Ці моделі враховують аеродинамічні сили, моменти, вплив вітрових навантажень, інерційні властивості платформи, параметри двигунів та особливості конструкції. Найбільш поширеним підходом до стабілізації та регулювання положення в просторі є використання PID-регуляторів, завдяки їхній простоті реалізації, передбачуваності поведінки та здатності забезпечувати високу швидкодію. PID-регулятори застосовуються майже на всіх комерційних мультикоптерах для підтримання висоти, орієнтації, стабілізації по каналах крену, тангажу та курсу. Проте в умовах складної аеродинаміки та нелінійної взаємодії компонентів їх ефективність може знижуватися, що стимулює розвиток нових підходів.

Одним із таких напрямів стали адаптивні системи керування [5-6], здатні в реальному часі підлаштовувати параметри регуляторів відповідно до зміни маси БПЛА, швидкості вітру, зносу пропелерів або коливань напруги на двигунах. Адаптивні методи на базі моделей Ляпунова [6], робастні схеми керування [7] та нечітка логіка [8] дають змогу досягати стабільності системи за умов неповної або неточної інформації про модель апарата. Нечіткі регулятори, на відміну від традиційних PID, здатні працювати з нечіткими множинами та лінгвістичними

правилами, що дозволяє швидко реагувати на нестандартні ситуації, де класичні методи втрачають ефективність. Широко досліджуються також такі методи, як LQR (лінійно-квадратичний регулятор), MPC (Model Predictive Control - керування на основі передбачення), а також нелінійне керування з використанням зворотного зв'язку. MPC особливо актуальне для автономних операцій, оскільки дає змогу враховувати обмеження по потужності, динаміці, безпеці та оптимізувати рух на основі прогнозованої поведінки системи на горизонті часу.

Навігаційні алгоритми сучасних БПЛА базуються на інтеграції даних від численних сенсорів - GPS/GNSS, IMU, барометрів, магнітометрів, візуальних камер, лідарів та радарів [6-8]. Важливим аспектом є необхідність компенсувати шум, дрейф та похибки вимірювання. У цьому контексті центральним є застосування фільтрів Калмана та їхніх модифікацій, таких як розширений фільтр Калмана (EKF) або фільтр Калмана з використанням сигма-точок (UKF). Вони дозволяють об'єднати інформацію з IMU та супутникових систем навігації, що забезпечує плавне й високоточне визначення положення та швидкості апарата. У середовищах з відсутністю стабільного GPS-сигналу - наприклад, у міських каньйонах, приміщеннях або під завісою радіоперешкод - застосовуються методи фільтра частинок, які дозволяють ефективно оцінювати стан системи за умов суттєвих нелінійностей та складного профілю шуму сенсорів.

Дедалі більшого значення набувають алгоритми візуальної та візуально-інерціальної навігації [9]. Сучасні підходи, такі як монокулярний, стереоскопічний або лідарний SLAM, забезпечують можливість одночасної локалізації та побудови карти раніше невідомого середовища. Це особливо важливо для автономних місій, під час яких БПЛА повинен орієнтуватися без зовнішньої інфраструктури. Методи SLAM поєднують обробку зображень, аналіз оптичного потоку, відстеження ключових точок, триангуляцію та оптимізацію графів, що дозволяє отримувати точні тривимірні карти [10]. Розвиток глибоких нейронних мереж сприяв підвищенню точності розпізнавання перешкод і навігаційних орієнтирів [11], появи системи семантичної навігації [12], коли БПЛА не лише будує карту, а й розуміє типи об'єктів на ній.

Планування маршруту є одним із найбільш динамічних напрямів розвитку алгоритмів для БПЛА [13-15]. Традиційні алгоритми пошуку, такі як A*, D* Lite, Theta*, Jump Point Search або Rapidly-exploring Random Trees (RRT), забезпечують побудову траєкторій, які враховують перешкоди та обмеження польотної зони. Проте для багатоточкових сценаріїв їх ефективність зменшується. У багатьох реальних застосуваннях маршрутизація БПЛА зводиться до задачі комівояжера або її варіацій, що потребують знаходження оптимального порядку обходу великої кількості точок. У цьому випадку високу ефективність демонструють еволюційні методи та алгоритми рою частинок, які здатні працювати з великими просторами рішення, що містять численні локальні оптимуми.

Генетичні алгоритми, алгоритми мурашиних колоній, оптимізація рою частинок, а також більш сучасні методи, такі як штучні нейронні еволюційні мережі та диференціальна еволюція, забезпечують можливість паралельного пошуку, адаптації до змінних умов, багатоцільової оптимізації та обмеженої чутливості до шумів у даних. Їхнє застосування дає змогу побудувати траєкторії, оптимізовані за довжиною, енерговитратами, безпекою, часом виконання місії або комплексними критеріями.

Уникнення перешкод є ще одним критичним компонентом навігаційної інтелектуальної системи. Тут застосовуються як реактивні, так і глобальні методи. Реактивні підходи, зокрема векторні поля, потенційні поля або методи на основі градієнтного спуску, дозволяють швидко реагувати на раптові зміни середовища. Глобальні підходи, засновані на прогнозуванні руху об'єктів, машинному навчанні або гібридних моделях, забезпечують більш високий рівень безпеки в динамічних умовах, наприклад під час польоту серед рухомих дронів, транспортних засобів або піхотних підрозділів.

Загальна тенденція розвитку сучасних алгоритмів керування та навігації безпілотних літальних апаратів полягає в інтеграції класичних методів із технологіями штучного інтелекту. Системи стають здатними до самонавчання, прогнозування поведінки середовища, адаптації до ворожих умов, компенсації втрат зв'язку та автономного прийняття рішень. Поява глибоких нейронних мереж,

методів підкріплювального навчання та еволюційних стратегій відкриває можливість створення повністю автономних комплексів, що здатні виконувати складні місії без участі оператора. Такі системи формують наступний етап розвитку БПЛА та визначають майбутнє безпілотної авіації.

1.4 Огляд сучасних платформ для програмної реалізації систем керування та навігації безпілотних літальних апаратів

Розглянемо п'ять сучасних платформ для програмної реалізації систем керування та навігації безпілотних літальних апаратів: PX4, ArduPilot (Cube/Pixhawk-екосистема), DJI (комерційна екосистема та SDK), Skydio (автономність на базі AI) та Auterion (комерційний софт-платформний стек). Для кожної платформи коротко опишемо архітектуру ПО, ключові «інтелектуальні» модулі (планування, уникнення перешкод, місія/фліт-менеджмент, інтерфейси) і приклади застосування на конкретних БПЛА.

PX4 – це відкритий flight-stack (прошивка автопілота) і супутня інфраструктура (mission planner, middleware), яка поширюється в екосистемі Dronocode. Архітектура PX4 розділяє реальний-часовий контролер (ниже-рівневі контролери стабілізації й моторні виходи), модуль автопілота та окремий рівень місії/планування, що може працювати на бортовому комп'ютері (companion computer). PX4 пропонує стандартизовані інтерфейси (uORB для внутрішньої шини повідомлень), підтримку різних апаратних платформ (Pixhawk/Herelink, Skynode та ін.) і багату модель flight modes (від manual до fully autonomous). Інтелектуальні функції у PX4 реалізуються як окремі модулі: планувальники шляху (global/local planners), підтримка мультисенсорного злиття (EKF2/EKF3), інтеграція з зовнішніми SLAM-пакетами та можливість запуску нейромереж на companion computer. PX4 також має відкриті інструменти для симуляції (Gazebo) та SDK/ROS-інтеграцію, що робить його популярним для досліджень і виробництва. Приклад: комерційні апарати та навчальні/дослідні платформи на базі Pixhawk / Holybro, а також OEM-виробники, котрі використовують PX4 як базовий flight-stack.

ArduPilot / Cube (Pixhawk-екосистема) – це гнучкий open-source стек з фокусом на надійність та багатоплатформеність. ArduPilot – ще один потужний відкритий flight-stack, що має довгу історію застосування у різних типах повітряних платформ (мультикоптери, літаючі крила, VTOL). ArduPilot орієнтований на можливість тонкої наладки, підтримує конфігурації з підвищеною надійністю (дублювання IMU, резервні GPS) і має багатий набір функцій для місійного планування, управління корисним навантаженням і телеметрії. Cube (The Cube, Cube Orange/Black) - апаратна реалізація автопілота в екосистемі ArduPilot/Pixhawk, яка широко застосовується в промислових і комерційних системах. Інтелектуальні можливості ArduPilot включають розширені режими місій (geofence, RTL з урахуванням запасу енергії), підтримку зовнішніх планувальників й алгоритмів уникнення на базі точкових хмар (OctoMap) і зв'язок з ground control software (Mission Planner, QGroundControl). ArduPilot часто застосовують у проєктах, де важлива глибока конфігурованість і надійність - наприклад, інспекційні платформи та наукові апарати.

DJI – вертикально інтегрована комерційна екосистема з сильними інтелектуальними режимами. DJI розробляє власне закритое бортове ПО, багаті SDK (Mobile SDK, Onboard SDK) та хмарні/ground-control сервіси. У комерційних серіях (Matrice, Phantom, Mavic) DJI інтегрує алгоритми автоматичного слідування, розпізнавання об'єктів, інтелектуального планування польоту (включно з автоматичними route planning/waypoint missions) та просунуте уникнення перешкод з використанням камер і сенсорів. Компанія надає розробникам інтерфейси для управління місіями та отримання телеметрії, але самі внутрішні алгоритми уникнення/локалізації - закриті. Для багатьох промислових застосувань (інспекції ліній, картографія, рятувальні операції) DJI залишається стандартом завдяки поєднанню стабільного апаратного забезпечення, надійного ПО та корисних інтелектуальних функцій (наприклад, SmartTrack, ActiveTrack, автоматичне планування маршрутів у Matrice-серії).

Skydio як приклад «AI-first» підходу. Skydio позиціонує себе як лідер у автономності на базі комп'ютерного зору та глибокого навчання. Компанія

розробляє власний стік автономності (Skydio Autonomy), який опирається на багатокамерні сенсорні конфігурації (наприклад, кілька 4K-камер) і потужні бортові нейромережі, що в режимі реального часу будують 3D-модель оточення, виявляють дрібні перешкоди (дроти, гілки), прогнозують траєкторії та забезпечують «expert pilot» поведінку без людської корекції. Skydio-дрони (S2, X2, ін.) широко використовують автономні інспекційні режими, слідкування за об'єктом у складних умовах та автоматичне планування траєкторій для огляду інфраструктури. Підхід Skydio демонструє, що значуща частина «інтелекту» може бути реалізована локально на борту через оптимізовані нейромережі й спеціалізовану обробку зображення.

Auterion – це «enterprise OS» та fleet-management із акцентом на AI/edge-compute. Auterion розвиває комерційну платформу (AuterionOS, Skynode тощо), яка базується на відкритих стандартах (часто спирається на PX4) і доповнює їх модулями для enterprise-керування флотом, телеметрії, безпечного оновлення та розгортання AI-моделей на борту. Auterion позиціонує свій продукт як «операційну систему для роботів», що дозволяє запускати місійні застосунки, аналіз відео на краю, об'єднувати дані у хмарі та централізовано керувати флотом. Recent news показують, що Auterion також активно просуває рішення для масштабного розгортання AI-керованих комплектів (Skynode) у спеціалізованих сценаріях, включно з оборонними застосуваннями, де важливі відмовостійкість і робота в умовах заглушення GPS/радіо. Auterion дає можливість OEM-виробникам і операторам отримати стандартизований стек з можливістю впровадження кастомних AI-модулів.

1.5 Постановка задачі

За результатами проведеного аналітичного огляду можна зробити висновок про актуальність і перспективність розробки та впровадження інтелектуальних складових в інформаційне, алгоритмічне і програмне забезпечення систем керування БПЛА. Тому метою кваліфікаційної роботи є розробка моделей, методів

та засобів інформаційної технології інтелектуального планування маршруту БПЛА на основі зведення задачі до задачі комівояжера та застосування генетичних алгоритмів для пошуку оптимального або близького до оптимального маршруту з урахуванням заданих обмежень.

Відповідно до поставленої мети сформульовані такі задачі:

- 1) Проаналізувати існуючі методи маршрутизації БПЛА та підходи до розв'язання задачі комівояжера.
- 2) Розробити математичну модель задачі інтелектуального планування маршруту БПЛА.
- 3) Сформувати модель комівояжера для представлення маршруту відповідно до умов польоту.
- 4) Розробити або адаптувати генетичний алгоритм для пошуку оптимальної траєкторії з урахуванням обмежень.
- 5) Створити програмну реалізацію інформаційної системи здатної до інтелектуального планування маршруту безпілотного літального апарату.
- 6) Провести перевірку працездатності розробленої інформаційної системи.

РОЗДІЛ 2

ВИБІР МЕТОДІВ ДОСЛІДЖЕННЯ, МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ

2.1 Основні поняття та визначення генетичних алгоритмів

Генетичні алгоритми (ГА) належать до класу еволюційних обчислювальних методів і є потужним інструментом розв’язання складних оптимізаційних задач, у яких традиційні аналітичні або чисельні методи не дають ефективних результатів. Основна ідея генетичних алгоритмів полягає у моделюванні процесів природного відбору та еволюції, які спостерігаються у біологічних системах. Вони застосовуються для пошуку глобальних або близьких до глобальних оптимумів у багатовимірних, дискретних чи комбінаційних просторах рішень.

Генетичний алгоритм – це стохастичний метод оптимізації, який працює із популяцією потенційних розв’язків задачі [16]. Кожен елемент популяції називається хромосомою або генотипом, і він кодує конкретне рішення у заданому представленні, наприклад, у вигляді бінарного вектора, послідовності чисел або більш складної структури. Значення або компоненти хромосоми називаються генами, які визначають певні параметри чи характеристики розв’язку.

Процес еволюції у генетичних алгоритмах реалізується через послідовне застосування трьох базових операторів: відбір, скрещування (crossover) і мутація. Відбір визначає, які особини з популяції будуть брати участь у створенні нового покоління, зазвичай на основі показника пристосованості, що оцінює якість рішення. Скрещування дозволяє комбінувати гени батьківських хромосом і формувати нові комбінації, потенційно кращі за попередні. Мутація вводить випадкові зміни в гени хромосоми, що забезпечує підтримку різноманітності популяції та запобігає передчасній конвергенції до локальних оптимумів.

Ключовим компонентом генетичного алгоритму є функція пристосованості (fitness function), яка кількісно оцінює якість кожного розв’язку відповідно до поставленої задачі. У задачах маршрутизації БПЛА, наприклад, функція

пристосованості може враховувати сумарну довжину маршруту, обмеження по висоті, безпеку траєкторії та витрати енергії. Якісно сформована функція пристосованості є визначальною для успішності роботи алгоритму.

Генетичні алгоритми відзначаються високою адаптивністю та універсальністю, що дозволяє застосовувати їх до задач різної природи: від комбінаційних задач (задача комівояжера, планування маршрутів) до числових оптимізацій і навчання нейромереж [17]. Основні характеристики, які роблять ГА ефективними, включають можливість паралельного пошуку в багатовимірних просторах, здатність долати локальні оптимуми завдяки стохастичній природі операторів та простоту інтеграції з іншими методами оптимізації.

Залежно від специфіки задачі, генетичні алгоритми можуть бути модифіковані або розширені. Існують мультиоб'єктні ГА, що дозволяють одночасно оптимізувати декілька критеріїв; елітні ГА, які зберігають найкращі рішення з попередніх поколінь; а також гібридні алгоритми, які комбінують ГА з локальними методами оптимізації для підвищення швидкості збіжності.

Таким чином, генетичні алгоритми є фундаментальним методом для розробки інтелектуальних систем оптимізації та прийняття рішень [16-17], особливо у випадках складних задач з великою кількістю змінних і обмежень, що робить їх природним вибором для інтелектуального планування маршрутів безпілотних літальних апаратів. Їхня гнучка структура дозволяє адаптувати алгоритм під конкретні потреби місії, визначати пріоритети різних критеріїв і інтегрувати з сучасними системами автономного керування та навігації.

2.2 Стандартні операції схрещування в генетичних алгоритмах

Схрещування (crossover) є ключовим оператором у генетичних алгоритмах, який забезпечує комбінування генетичної інформації батьківських хромосом для формування нових індивідів у наступному поколінні. Основна мета схрещування полягає у створенні варіацій, що потенційно можуть мати вищу пристосованість, ніж батьківські хромосоми, завдяки поєднанню кращих ознак обох батьків.

Схрещування відповідає біологічному процесу рекомбінації та є основним механізмом пошуку нового простору рішень у генетичному алгоритмі.

У класичному підході до ГА існують кілька стандартних моделей схрещування [18], що визначають спосіб обміну генами між хромосомами.

Одноточкове схрещування (single-point crossover). У цій моделі випадковим чином обирається точка розрізу в довжині хромосоми, після чого частини двох батьків обмінюються своїми сегментами. Наприклад, якщо хромосоми кодують послідовність відвідуваних точок у задачі маршрутизації, одноточкове схрещування дозволяє частину маршруту одного батька поєднати з частиною маршруту іншого, зберігаючи при цьому структуру маршрутних вузлів.

Дво- або багатоточкове схрещування (two- or multi-point crossover). Ця модель передбачає вибір двох або більше точок розрізу, між якими обмінюються сегменти хромосом. Таке схрещування забезпечує більш різноманітні комбінації генів і зменшує ймовірність втрати значущих послідовностей, що може бути критично для задач із жорсткими обмеженнями або порядковою структурою даних.

Унітарне або рівномірне схрещування (uniform crossover). При рівномірному схрещуванні кожен ген батьків вибирається незалежно з певною ймовірністю і передається потомку. Цей метод забезпечує максимальне перемішування генів і дозволяє швидко генерувати нові комбінації, що особливо корисно на ранніх етапах еволюції для підвищення різноманітності популяції.

Схрещування часткового відновлення порядку (Partially Mapped Crossover, PMX). Ця модель широко застосовується для комбінаційних задач, де важлива порядкова структура генів, наприклад у задачі комівояжера або маршрутизації БПЛА. PMX дозволяє обмінювати підпослідовності між батьками та зберігати унікальність елементів у потомках, що критично для забезпечення валідності маршрутів.

Схрещування упорядковане (Order Crossover, OX) та Cycle Crossover (CX). Ці методи також застосовуються для задач з порядковою структурою, забезпечуючи збереження відносного порядку елементів. OX копіює певний сегмент хромосоми одного батька і заповнює решту генів у порядку другого батька. CX формує

потомка таким чином, щоб кожен ген займав позицію одного з батьків, зберігаючи циклічні відповідності.

Вибір конкретної моделі схрещування залежить від характеру задачі, структури хромосом та вимог до збереження порядку або комбінаційних властивостей. Для задач планування маршрутів БПЛА та оптимізації траєкторій важливими є методи РМХ, ОХ та СХ, оскільки вони дозволяють підтримувати валідність маршрутів і уникати дублювання вузлів. Одно- та двоточкові схрещування застосовуються переважно для бінарного або числового кодування хромосом, де порядок генів менш критичний.

Таким чином, оператори схрещування забезпечують основний механізм комбінування генетичної інформації і відіграють ключову роль у пошуку оптимальних або близьких до оптимальних рішень. Грамотне поєднання методів схрещування з відбором і мутацією дозволяє генетичним алгоритмам ефективно досліджувати простір рішень, підтримувати різноманітність популяції і досягати високої швидкості збіжності, що є критично важливим для задач інтелектуального планування маршрутів безпілотних літальних апаратів.

2.3 Стратегії відбору нового покоління у генетичних алгоритмах

Відбір (selection) є критичною стадією генетичного алгоритму, оскільки визначає, які хромосоми з поточної популяції будуть обрані для формування нового покоління через операції схрещування та мутації [19]. Основна мета відбору полягає у наданні більш пристосованим особинам більшої ймовірності передати свої гени нащадкам, одночасно зберігаючи різноманітність популяції для уникнення передчасної збіжності до локальних оптимумів.

Стратегії відбору можна умовно розділити на ймовірнісні та конкурентні методи [19].

1. Рулетковий (Fitness-Proportionate) відбір базується на ймовірності вибору хромосоми, пропорційній її значенню функції пристосованості. Якщо f_i – значення функції пристосованості хромосоми i , а P_i – ймовірність її вибору, то:

$$P_i = \frac{f_i}{\sum_{j=1}^N f_j}, \quad (2.1)$$

де N – розмір популяції.

Рулетковий відбір дозволяє більш пристосованим хромосомам мати вищу ймовірність участі у формуванні нового покоління, проте слабкі хромосоми теж мають невеликий шанс вижити, що підтримує генетичне різноманіття. Недоліком методу є схильність до «premature convergence», коли одна хромосома сильно переважає інші за значенням пристосованості.

2. Турнірний відбір (Tournament Selection) ґрунтується на проведенні серії «турнірів» серед випадково обраних k особин популяції. З кожного турніру обирається хромосома з найвищою пристосованістю для включення у нове покоління. Ймовірність обрання найсильнішої хромосоми в турнірі часто можна задавати як p , а решта $1-p$ для менш пристосованих:

$$P_{winner} = p + \frac{1-p}{k-1}. \quad (2.2)$$

Турнірний відбір добре масштабується з розміром популяції та легко реалізується. Він контролює тиск відбору через параметр k , коли більший турнір підвищує ймовірність обрання сильних хромосом, зменшуючи випадковість.

3. У Rank-based відборі (відборі за рангом) хромосоми сортуються за зростанням або спаданням значень функції пристосованості, і ймовірність відбору визначається рангом, а не абсолютним значенням:

$$P_i = \frac{r_i}{\sum_{j=1}^N r_j}, \quad (2.3)$$

де r_i – ранг i -тої хромосоми.

Цей метод зменшує ризик домінування однієї сильної хромосоми і забезпечує більш рівномірний тиск відбору. Він особливо ефективний у задачах з широким діапазоном значень функції пристосованості.

4. Стохастичне універсальне відтворення (Stochastic Universal Sampling, SUS) є варіацією рулеткового відбору, яка забезпечує рівномірне та менш випадкове відтворення популяції. У цій стратегії використовується єдина «рулетка» з N рівномірно розташованими вказівниками, які визначають, які хромосоми будуть вибрані. Це зменшує дисперсію результатів відбору і гарантує, що хромосоми з високою пристосованістю будуть представлені у пропорційній кількості.

5. Елітний відбір (Elitism) не є самостійною стратегією, а доповнює інші методи. Він передбачає автоматичне перенесення E найкращих хромосом поточного покоління у наступне без змін. Це гарантує збереження найкращих рішень і прискорює збіжність алгоритму. При цьому параметр E зазвичай становить від 1 до 5% від розміру популяції:

$$\text{Elite} = \arg \max_{i=1 \dots N} f_i. \quad (2.4)$$

Елітність у поєднанні з рулеткою або турніром дозволяє балансувати між дослідженням простору рішень і швидким використанням найкращих знайдених рішень.

2.4 Мутаційні оператори та їх вплив на різноманітність популяції

Мутація є одним із базових операторів генетичного алгоритму, що забезпечує введення випадкових змін у хромосоми і, таким чином, підтримує генетичне різноманіття популяції [20]. Вона виконує функцію «дослідження» простору рішень, протидіючи надмірній збіжності популяції і дозволяє алгоритму уникати локальних оптимумів.

Мутація застосовується після операцій відбору та схрещування і може виконуватися з певною ймовірністю P_m , яка називається ймовірністю мутації.

Нехай x_i – ген i -тої хромосоми, тоді операція мутації формально може бути описана як:

$$x'_i = \begin{cases} f_{mut}(x_i), & \text{з ймовірністю } P_m, \\ x_i, & \text{з ймовірністю } 1 - P_m, \end{cases}$$

де $f_{mut}(x_i)$ – функція, яка визначає тип зміни гену (наприклад, бінарне інвертування, випадкова заміна або перестановка).

Існують різні типи мутацій залежно від кодування хромосом:

1) Бінарна мутація (bit-flip mutation). Для бінарного кодування генів зміна полягає в інверсії біта:

$$x'_i = 1 - x_i.$$

2) Арифметична мутація (arithmetic mutation). Для числового кодування генів зміна здійснюється шляхом додавання випадкового числа δ із заданим розподілом:

$$\begin{aligned} x'_i &= x_i + \delta, \\ \delta &\sim U(-\Delta, +\Delta) \end{aligned}$$

де $U(-\Delta, +\Delta)$ – рівномірний розподіл у межах допустимого діапазону.

3) Перестановка (swap mutation). Використовується для порядкового кодування хромосом, коли два гени міняються місцями:

$$(x'_i, x'_j) = (x_j, x_i)$$

де $i \neq j$.

4) Інверсія підрядка (inversion mutation). Обирається випадкова підпослідовність генів і змінюється порядок її елементів:

$$(x'_i, \dots, x'_j) = (x_j, x_{j-1}, \dots, x_{i+1}, x_i)$$

де $i < j$.

Мутаційний оператор впливає на різноманітність популяції, яку можна формалізувати як середнє або максимальне відстань між хромосомами. Наприклад, для популяції з N хромосом різноманітність D можна оцінити через середню відстань Хеммінга для бінарного кодування:

$$D = \frac{2}{N(N-1)} \sum_{i=1}^{N-1} \sum_{j=i+1}^N H(x_i, x_j),$$

де $H(x_i, x_j)$ – відстань Хеммінга між хромосомами i та j . Введення мутації збільшує D , запобігаючи передчасній збіжності та дозволяючи алгоритму досліджувати нові області простору рішень.

Вибір ймовірності мутації P_m є критичним параметром. Надмірно велика P_m призводить до випадкового пошуку і втрати накопиченої еволюційної інформації, тоді як надмала P_m зменшує ефективність обходу локальних оптимумів. Зазвичай значення P_m обирають у межах 0,001–0,1 для числових або бінарних задач, залежно від розміру хромосом та складності простору рішень.

Існують також адаптивні стратегії мутації, коли ймовірність P_m змінюється під час еволюції залежно від поточної різноманітності популяції або швидкості покращення функції пристосованості. Наприклад:

$$P_m(t) = P_{m0} \cdot e^{-\alpha \cdot t}$$

де P_{m0} – початкове значення ймовірності мутації, t – номер покоління, α – параметр зменшення. Такий підхід дозволяє на ранніх етапах підтримувати високу різноманітність, а пізніше зосередити пошук на локальних оптимумах.

Таким чином, мутаційні оператори є важливим елементом генетичного алгоритму, який забезпечує баланс між експлуатацією та дослідженням, підтримує генетичне різноманіття та підвищує ймовірність знаходження глобального оптимуму.

2.5 Повна схема роботи генетичного алгоритму

Генетичний алгоритм (ГА) є ітераційним стохастичним методом оптимізації, що поєднує оператори відбору, схрещування та мутації для пошуку глобальних або близьких до глобальних оптимумів у складних просторах рішень. Повна схема роботи ГА [21] складається з наступних основних етапів:

1. Ініціалізація популяції

На початку алгоритму формується початкова популяція з N хромосом:

$$P(0) = \{x_1^0, x_2^0, \dots, x_N^0\}$$

де кожна хромосома x_1^0 представляє допустимий розв'язок задачі.

Ініціалізація може виконуватися випадковим чином або з використанням евристичних методів для кращого старту.

2. Оцінка функції пристосованості

Для кожної хромосоми обчислюється значення функції пристосованості $f(x_i^{(t)})$, що кількісно оцінює її якість:

$$f_i^{(t)} = f(x_i^{(t)})$$

де $i=1,2,\dots,N$.

Це значення використовується для відбору та визначає ймовірність участі хромосоми у формуванні наступного покоління.

3. Відбір батьків

На основі значень функції пристосованості обираються хромосоми для схрещування. Якщо використовується рулетковий відбір, ймовірність вибору хромосоми $x_i^{(t)}$ визначається як (2.1). Інші методи відбору, такі як турнірний або rank-based, формалізуються відповідними правилами (2.2 – 2.3), описаними у підрозділі 2.4.

4. Схрещування (Crossover)

Для кожної пари обраних батьків $x_p^{(t)}$, $x_q^{(t)}$ застосовується оператор схрещування з ймовірністю P_c . Наприклад, при одноточковому схрещуванні потомки формуються так:

$$\begin{aligned} x_p'^{(t+1)} &= (x_p^{(t)}[1:k], x_q^{(t)}[k+1:L]), \\ x_q'^{(t+1)} &= (x_q^{(t)}[1:k], x_p^{(t)}[k+1:L]), \end{aligned}$$

де L – довжина хромосоми, k – випадкова точка розрізу.

Для порядкового кодування застосовуються спеціальні методи (PMX, ОХ, СХ), що зберігають валідність комбінацій.

5. Мутація

Кожна хромосома $x_i'^{(t+1)}$ підлягає мутації з ймовірністю P_m . Якщо x_i' містить гени $x'_{i,j}$, тоді:

$$x_{i,j}''^{(t+1)} = \begin{cases} f_{mut}(x'_{i,j}), & \text{з ймовірністю } P_m, \\ x'_{i,j}, & \text{з ймовірністю } 1 - P_m, \end{cases}$$

де f_{mut} – функція мутації, яка може бути бінарною, арифметичною або перестановочною.

6. Формування нового покоління

Нове покоління $P(t+1)$ формується шляхом включення потомків, отриманих після схрещування та мутації. Якщо використовується елітність, найкращі E хромосом попереднього покоління автоматично переносяться у $P(t+1)$:

$$P(t+1) = \text{Elite}(P(t)) \cup \text{Offspring}.$$

7. Умови завершення алгоритму

Процес еволюції повторюється до виконання одного з критеріїв зупинки:

- Досягнення максимального числа поколінь t_{max}

$$t \geq t_{max}.$$

- Досягнення задовільного значення функції пристосованості f_{target}

$$\max_{i=1 \dots N} f_i^{(t)} \geq f_{target}.$$

- Відсутність покращення кращого рішення протягом T_{stall} поколінь:

$$\max_{i=1 \dots N} f_i^{(t)} - \max_{i=1 \dots N} f_i^{(t-T_{stall})} < \varepsilon.$$

Повна ітераційна схема генетичного алгоритму може бути записана так:

$$P(t+1) = \text{Mutate}(\text{Crossover}(\text{Select}(P(t)))),$$

де *Select*, *Crossover*, *Mutate* – відповідні оператори відбору, схрещування та мутації. Процес повторюється $t = 0, 1, \dots, t_{max}$ або до виконання критеріїв зупинки.

Таким чином, генетичний алгоритм формує циклічний процес еволюції, у якому поєднання відбору, схрещування та мутації забезпечує баланс між

експлуатацією наявних гарних рішень та дослідженням нового простору рішень, що є основою його ефективності для широкого класу оптимізаційних задач.

2.6 Вибір параметрів генетичного алгоритму та їх вплив на ефективність

Ефективність генетичного алгоритму (ГА) значною мірою залежить від правильного вибору його параметрів, таких як розмір популяції, ймовірність схрещування P_c , ймовірність мутації P_m та критерії відбору. Неправильне налаштування цих параметрів може призвести до передчасної збіжності або повільного пошуку оптимального рішення.

Розмір популяції визначає кількість хромосом у кожному поколінні. Він впливає на здатність алгоритму досліджувати простір рішень та підтримувати різноманітність. Занадто мала популяція $N \ll 50$ може призвести до швидкої збіжності до локальних оптимумів, тоді як надмірно велика $N \gg 500$. Рекомендована оцінка мінімального розміру популяції може бути виконана за формулою емпіричного характеру:

$$N_{min} \approx 10 \cdot L,$$

де L – довжина хромосоми.

Ймовірність застосування оператора схрещування визначає, який відсоток обраних батьків бере участь у комбінуванні генів. Якщо P_c занадто мала, нові комбінації генів будуть обмежені, що уповільнює пошук глобального оптимуму. Якщо P_c занадто велика, знижується вплив елітності та накопиченої інформації. Типові значення $0.6 \leq P_c \leq 0.9$.

Для числових задач можна використовувати адаптивне схрещування:

$$P_c(t) = P_{c\ min} + (P_{c\ max} - P_{c\ min}) \frac{f_{max}(t) - \bar{f}(t)}{f_{max}(t) - f_{min}(t)}$$

де $f_{max}(t)$, $f_{min}(t)$ – максимальна, мінімальна та середня пристосованість у поколінні t , $P_{c\ max}$, $P_{c\ min}$ – задані межі ймовірності. Така адаптація дозволяє більш активно комбінувати генетичний матеріал на ранніх етапах і зменшувати хаотичність у пізніх поколіннях.

Мутація вводить випадкові зміни в хромосоми та підтримує різноманітність популяції. Занадто велика ймовірність мутації перетворює алгоритм на випадковий пошук, тоді як занадто мала – зменшує здатність долати локальні оптимуми.

Зазвичай P_m задають у межах $0.001 \leq P_m \leq 0.1$.

Для адаптивних стратегій:

$$P_m(t) = P_{m\ min} + (P_{m\ max} - P_{m\ min}) \frac{D_{max}(t) - D(t)}{D_{max}(t)},$$

де $D(t)$ – поточна різноманітність популяції, $D_{max}(t)$ – максимальна можлива різноманітність. Такий підхід збільшує ймовірність мутації при низькому різноманітті та зменшує її, коли популяція достатньо різноманітна.

Елітні хромосоми, які автоматично переносяться в наступне покоління, дозволяють зберегти найкращі рішення. Типові значення $E \approx 1\% - 5\%$ від N .

Надмірна елітність може зменшити різноманітність та призвести до передчасної збіжності, тоді як недостатня — зменшити ефективність пошуку.

Таким чином, ефективність генетичного алгоритму залежить не тільки від структури операторів, а й від правильного налаштування їх параметрів, які можна адаптувати під конкретну задачу для досягнення оптимального компромісу між швидкістю збіжності та якістю рішень.

2.7 Моделі генетичних алгоритмів

Класичний генетичний алгоритм (рис. 2.1 а) зазвичай працює з однією популяцією в серійному режимі, де формування поколінь відбувається послідовно. Однак для складних оптимізаційних задач, великих розмірів популяції або значних

обчислювальних витрат застосовуються паралельні моделі генетичних алгоритмів. Паралелізація дозволяє прискорити збіжність, покращити різноманітність популяції та зменшити ймовірність передчасної збіжності до локальних оптимумів.

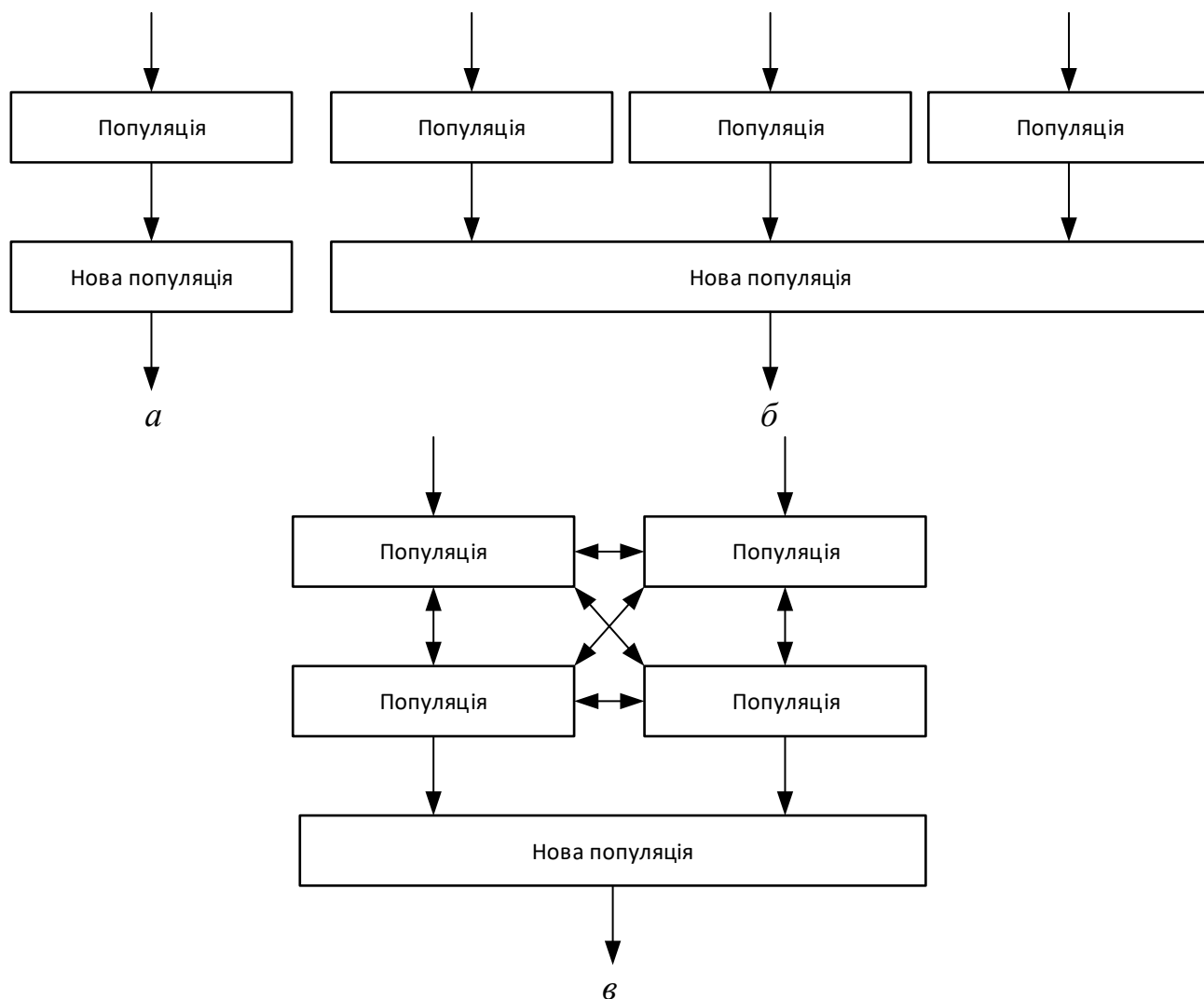


Рисунок 2.1 – Схематичне відображення основних моделей генетичних алгоритмів: *а* – класична модель, *б* – модель паралельного генетичного алгоритму, *в* – клітинна модель генетичних алгоритмів

Джерело: запропоновано автором

Паралельні генетичні алгоритми [22] (рис. 2.1 б) передбачають одночасну обробку частин популяції або різних популяцій на окремих процесорах чи обчислювальних вузлах. Це може бути реалізовано шляхом одночасної оцінки функції пристосованості для всіх хромосом або розподілення популяції на

підпопуляції, які еволюціонують майже незалежно. Такий підхід забезпечує ефективне використання обчислювальних ресурсів та дозволяє працювати з великими популяціями без значного збільшення часу виконання.

Однією з найбільш поширених паралельних моделей є островна модель. Вона передбачає розбиття популяції на декілька підпопуляцій, які еволюціонують незалежно одна від одної, підтримуючи різноманітність і зменшуючи ризик передчасної збіжності. Час від часу між підпопуляціями здійснюється обмін найкращими хромосомами, що дозволяє успішним рішенням поширюватися по всій популяції. Такий механізм міграції забезпечує баланс між локальною оптимізацією на кожному «острові» та глобальним пошуком в просторі рішень.

Клітинна модель генетичних алгоритмів [22] (рис. 2.1 в) організовує популяцію у вигляді регулярної решітки, де кожна хромосома займає певну клітину. Операції відбору, схрещування та мутації виконуються лише з урахуванням локального оточення кожної хромосоми. Це дозволяє еволюції відбуватися поступово, без різких змін у всій популяції, та підтримує високу різноманітність. Клітинна модель природно піддається паралельній обробці і добре підходить для багатопікових задач та задач з локальною оптимізацією, де необхідно одночасно досліджувати велику кількість потенційних рішень.

В цілому, паралельні моделі генетичних алгоритмів дозволяють підвищити ефективність пошуку, підтримувати генетичне різноманіття та зменшити ймовірність застрягання в локальних оптимумах. Островна модель ефективна для багатопікових задач завдяки незалежній еволюції підпопуляцій та обмеженій міграції, тоді як клітинна модель забезпечує локальну взаємодію хромосом та плавний розвиток популяції, що особливо важливо для стабільної та адаптивної еволюції. Комбінування цих підходів з класичними генетичними алгоритмами дозволяє значно підвищити стабільність та продуктивність алгоритму.

2.8 Генетичний алгоритм для побудови оптимального маршруту БПЛА у тривимірному просторі

Оптимізація маршруту безпілотних літальних апаратів (БПЛА) у 3D-просторі природним чином формулюється як варіант задачі комівояжера (TSP), у якій необхідно знайти найкоротший замкнений маршрут через заданий набір опорних точок. Для такої задачі доцільним є використання генетичного алгоритму, здатного ефективно працювати з великими комбінаційними просторами.

2.8.1 Подання вихідних даних

Задано:

- множину опорних точок $P = \{p_0, p_1, \dots, p_m\}$, $p_m = (x_i, y_i, z_i)$,
- p_0 – точка старту/фінішу БПЛА;
- параметри ГА:
 - N – розмір популяції,
 - α – частка нових особин у кожному поколінні,
 - P_m – ймовірність мутації.

Для швидкого обчислення довжини маршруту будується матриця:

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$$

2.8.2 Хромосомне представлення маршруту

Оскільки маршрут є перестановкою опорних точок, хромосома подається як упорядкована послідовність індексів:

$$[0, i_1, i_2, \dots, i_k, 0]$$

де 0 — стартова точка.

Нехай маємо точки:

0 – база

- 1 – оглядова точка
- 2 – точка збору даних
- 3 – точка розвороту
- 4 – зона контролю

Тоді хромосома може бути:

[0, 3, 1, 4, 2, 0]

2.8.3 Ініціалізація першої популяції

Початкова популяція — це N випадкових перестановок:

[0, 1, 2, 3, 4, 0]

[0, 3, 2, 4, 1, 0]

[0, 4, 1, 3, 2, 0]

...

Для кожної особини обчислюється її шлях:

$$L = \sum_{i=0}^m d_{route[i], route[i+1]} =$$

$$= \sum_{i=0}^m \sqrt{(x_{route[i]} - x_{route[i+1]})^2 + (y_{route[i]} - y_{route[i+1]})^2 + (z_{route[i]} - z_{route[i+1]})^2}$$

Після чого популяція впорядковується за зростанням довжини маршруту.

2.8.4. Оператор схрещування

Розглянемо роботу оператора схрещування на прикладі.

Нехай є дві хромосоми:

A = [0, 2, 5, 4, 1, 3, 0]

B = [0, 4, 1, 3, 5, 2, 0]

Позиції 0 і 6 фіксовані та не беруть участі в операціях.

Крок 1. Вибір сегмента для обміну

Нехай точки розриву: $p1 = 2$, $p2 = 4$

Виділяємо фрагменти:

A: [2 | 5, 4, 1 | 3]

B: [4 | 1, 3, 5 | 2]

Крок 2. Створення заготовок для нащадків

C1 = [0, x, x, x, x, x, 0]

C2 = [0, x, x, x, x, x, 0]

Крок 3. Обмін середніх сегментів

C1 = [0, x, 1, 3, 5, x, 0]

C2 = [0, x, 5, 4, 1, x, 0]

Крок 4. Заповнення решти позицій з уникненням дублювання

Для **C1**

Хромосома **B** поза сегментом:

[4, 2]

Перевіряємо кожний елемент:

4 → не присутній у **C1** → вставляємо у першу порожню позицію

2 → теж не присутній → вставляємо далі

Отримуємо:

C1 = [0, 4, 1, 3, 5, 2, 0]

Для c_2

Хромосома **A** поза сегментом:

[2, 3]

2 → не дублюється

3 → дублюється (3 є всередині сегмента ⇒ конфлікт)

Тому замінюємо на відповідний елемент із **B** у цій позиції: замість 3 беремо

5 → але 5 теж дублюється беремо наступний відповідний елемент → 2 → вже є

У результаті застосовується стандартна РМХ-відповідність і вибирається перший елемент, що не дублюється ⇒ 4.

Отже:

$c_2 = [0, 2, 5, 4, 1, 3, 0]$

Потомство:

$c_1 = [0, 4, 1, 3, 5, 2, 0]$

$c_2 = [0, 2, 5, 4, 1, 3, 0]$

2.8.5 Оператор мутації

Застосуємо swap-мутацію.

До хромосоми:

[0, 4, 1, 3, 5, 2, 0]

Вибираємо випадково позиції 2 і 5 → міняємо їх місцями:

[0, 4, 2, 3, 5, 1, 0]

Така мутація гарантує збереження коректності перестановки.

2.8.6. Формування нового покоління

Цикл, що виконується у кожному поколінні:

1. копіювання найкращої частини популяції (елітність);
2. генерація нових особин через PMX;
3. застосування мутації з імовірністю P_m ;
4. повторна оцінка довжин маршрутів;
5. сортування;
6. перехід до наступного покоління.

Цикл триває, доки не буде досягнуто умови завершення (кількість поколінь, стабілізація фітнес-функції, зупинка користувачем).

РОЗДІЛ 3

РОЗРОБКА ТА ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Вибір середовища для програмної реалізації

Для реалізації програмної системи оптимізації маршруту БПЛА було обрано середовище MATLAB, що зумовлено низкою технічних, функціональних та дослідницьких переваг, які роблять його оптимальним інструментом саме для задачі еволюційної оптимізації у тривимірному середовищі з рельєфом.

Перш за все, MATLAB забезпечує високий рівень інтеграції математичних і візуалізаційних засобів, що є критично важливим для задач маршрутизації у просторі [23]. Для розроблення генетичного алгоритму були використані переваги MATLAB у роботі з матрицями, зокрема швидкі операції над великими масивами, зручна реалізація циклів та векторизація обчислень. Це сприяло побудові ефективних модулів, таких як генерація популяцій, формування матриць відстаней та реалізація операторів кросоверу і мутації.

Також MATLAB має розвинені засоби тривимірної графіки [24], які дозволили забезпечити якісну візуалізацію рельєфу та поточних маршрутів. Для задачі оптимізації траєкторії БПЛА в умовах складного рельєфу можливість інтерактивного перегляду тривимірних поверхонь, трасування маршруту та побудови динамічних графіків збіжності є суттєвою перевагою. Це спрощує аналіз поведінки алгоритму, виявлення локальних мінімумів і порівняння різних конфігурацій параметрів.

Важливо також зазначити, що MATLAB забезпечує просту організацію модульної структури програми [25]. Кожна функція системи реалізована як окремий підмодуль, який легко відлагоджувати, тестувати й розширювати. Такий підхід особливо корисний для еволюційних алгоритмів, що складаються з великої кількості незалежних компонентів (селекція, кросовер, мутація, локальний пошук,

оцінка фітнесу). Вибір MATLAB уможливив розроблення зрозумілої, структурованої та масштабованої архітектури.

З точки зору проведення експериментальних досліджень MATLAB також є вигідним рішенням. Він містить необхідні засоби для збору статистики, побудови графіків, аналізу збіжності та автоматизації серій експериментів [26]. Це дозволяє легко оцінити ефективність різних параметрів генетичного алгоритму, а також порівняти гібридні та класичні підходи.

Окрему увагу заслуговує можливість швидкого прототипування [27-28]. Реалізація гібридного ГА з 2-opt у MATLAB значно скорочує час розробки порівняно з низькорівневими мовами. Завдяки цьому можна зосередитися на експериментальній частині дослідження та аналізі отриманих результатів, а не на технічних аспектах управління пам'яттю чи реалізації спеціалізованих структур даних.

Отже, вибір MATLAB як середовища програмної реалізації є обґрунтованим через його високий рівень інтеграції засобів чисельного моделювання, можливості тривимірної візуалізації, розвинуту екосистему для наукових обчислень та зручність побудови модульних алгоритмічних структур. У сукупності ці чинники роблять MATLAB оптимальною платформою для реалізації та дослідження генетичних алгоритмів оптимізації маршруту БПЛА у складних просторових умовах.

3.2 Короткий опис програмної реалізації

Програмна реалізація складається з набору MATLAB-функцій, об'єднаних у єдину систему для розв'язання задачі оптимізації маршруту БПЛА у тривимірному просторі з урахуванням рельєфу. Основна функція `main` виконує ініціалізацію параметрів, генерацію точок та рельєфу, формування початкової популяції, обчислення довжин маршрутів та ітеративну еволюцію популяції відповідно до процедур селекції, кросоверу, мутації та локального пошуку. Візуалізація здійснюється в реальному часі, що дозволяє оцінити збіжність алгоритму. Архітектура коду є модульною: кожен етап (генерація, оцінювання, рекомбінація,

локальний пошук, відображення результатів) реалізовано окремими функціями, які взаємодіють через стандартизовану структуру даних. (Таблиця 3.1)

Подамо опис кожної функції з позицій її ролі в архітектурі, алгоритмічних особливостей і взаємодії з іншими компонентами.

Таблиця 3.1 – Опис функцій програмної реалізації

Назва функції	Призначення	Основні вхідні дані	Основний вихід
main	Головна функція; виконує повний цикл ГА: генерацію даних, оптимізацію, візуалізацію	Параметри задачі та ГА (всередині функції)	Найкращий знайдений маршрут, графіки збіжності
dist_matrix	Обчислення матриці евклідових відстаней між усіма точками	points ($N \times 3$)	D — матриця відстаней $N \times N$
popul_gen	Генерує початкову популяцію перестановок точок (2..n)	n — кількість точок; pop_size	Матриця популяції pop (pop_size $\times$ (n-1))
popul_fitness	Обчислює довжину маршруту (фітнес) для кожного індивіда популяції	pop, D	Вектор fit — довжини маршрутів
tournament_select	Виконує турнірну селекцію для вибору батьків	fit, k — кількість учасників	Індекс обраного індивіда
order_crossover	Order Crossover (OX) для перестановок; формує 2 нащадків	p1, p2 — батьківські маршрути	c1, c2 — нащадки
mutation_inversion	Мутація інверсією: переверот випадкового сегмента перестановки	ch — маршрут	Модифікований маршрут
two_opt_local	Локальний пошук 2-opt для покращення маршруту	route, D	Оптимізований маршрут
get_idx	Повертає номер реального вузла у перестановці (службова функція)	route, pos	idx — індекс вузла
points_gen	Генерує контрольні точки маршруту та рельєф, виконує нормування у глобальний простір	n, coords_limits	points, ground_points, ground
points_show	Візуалізація рельєфу та точок у 3D	points, ground_points, ground	Графічне зображення
draw_progress	Відображає найкращий поточний маршрут під час роботи ГА	points, ground, pop, D, best_history, gen	Оновлена візуалізація

Джерело: запропоновано автором

1. Функція `main` як центральний координатор алгоритму

Функція `main` виконує завдання комплексного керування життєвим циклом оптимізаційного процесу. Вона ініціалізує параметри генетичного алгоритму, визначає розмір популяції, кількість поколінь, параметри селекції та мутації. На етапі підготовки вводяться координати просторових точок та генерується модель рельєфу через функцію `points_gen`.

Після цього `main` створює початкову популяцію маршрутів у вигляді перестановок точок, формує матрицю відстаней, а далі запускає еволюційний цикл. На кожній генерації виконується добір батьківських хромосом, застосовується кросовер і мутація, проводиться локальне покращення `2-opt`. У разі потреби алгоритм періодично візуалізує стан пошуку через функцію `draw_progress`.

Таким чином, `main` відповідає за керування усіма етапами — від підготовки даних до отримання фінального оптимального маршруту.

2. Функція `dist_matrix(points)`

Ця функція створює матрицю попарних відстаней між усіма точками в просторі. На рівні програмної реалізації вона повертає квадратну таблицю, де кожен елемент відповідає відстані між парою точок із множини. Попереднє обчислення значно прискорює виконання основного алгоритму, оскільки під час еволюційної оптимізації маршрути оцінюються багаторазово, а повторний прямий перерахунок координат був би надмірно витратним.

3. Функція `popul_gen(n, pop_size)`

Це генератор початкової популяції. Кожен індивід у популяції — це перестановка точок, що задає порядок їх обходу БПЛА. Початкова позиція приймається фіксованою (наприклад, точка 1), а всі інші точки перемішуються випадковим чином. Таким чином формується різноманіття маршрутів, необхідне для старту еволюції.

4. Функція `popul_fitness(pop, D)`

Виконує ключове завдання — обчислення довжини маршруту для кожного індивіда. Використовуючи вже сформовану матрицю відстаней, ця функція проходить послідовно всі точки маршруту і сумує довжини переходів. Саме фітнес

визначає якість маршруту, тому функція активно використовується у всіх механізмах відбору та покращення.

5. Функція `tournament_select(fit, k)`

Реалізує турнірну селекцію, що належить до найстабільніших і найпоширеніших методів еволюційного добору. Принцип роботи полягає у випадковому виборі кількох індивідів з популяції, порівнянні їхніх фітнесів та виборі найкращого. Цей підхід забезпечує контрольований тиск селекції й уникнення передчасної втрати різноманіття.

6. Функція `order_crossover(p1, p2)`

Функція реалізує Order Crossover (OX) — спеціальний оператор кросоверу для перестановок. На рівні коду оператор:

- обирає випадковий сегмент у першого батька;
- переносить цей сегмент до нащадка в незмінному вигляді;
- доповнює маршрут елементами другого батька в тому порядку, у якому вони присутні в його хромосомі;
- пропускає ті елементи, що вже потрапили в маршрут.

Таким чином, OX уникає дублювання та зберігає послідовні структури маршруту, що робить його придатним для задачі комівояжера.

7. Функція `mutation_inversion(ch)`

Це мутаційний оператор інверсії. Алгоритм вибирає випадкові позиції у перестановці і перевертає елементи на цьому відрізку у зворотному напрямку. Така операція ефективно руйнує локальні мінімальні структури та створює нові комбінації точок, не порушуючи вимог до маршрутів (перестановка залишається коректною).

8. Функція `two_opt_local(route, D)`

Функція впроваджує локальне покращення маршруту шляхом застосування методу 2-opt. У програмному контексті процедура перебирає всі пари позицій у маршруті, пропонує замінити сегмент маршруту його перевернутою версією та перевіряє, чи покращує це загальний результат. Якщо покращення відбувається,

зміни застосовуються, і пошук триває, доки не буде досягнуто стабільності маршруту.

Цей модуль перетворює стандартний генетичний алгоритм на гібридний, оскільки додає елемент інтенсивного локального пошуку всередині глобального еволюційного механізму.

9. Функція `get_idx(route, pos)`

Допоміжна функція, що повертає фактичний індекс елемента у маршруті для заданої позиції. Використовується переважно в алгоритмі 2-opt для коректної обробки маршрутів зі зафіксованим початком і кінцем.

10. Функція `points_gen(n, coords_limits)`

Функція генерує набір точок у 3D-просторі, а також створює модель рельєфу. Для побудови рельєфу використовується функція типу `peaks`, що дозволяє змодельовати складні поверхні з різкими висотними перепадами. Після генерації дані масштабуються відповідно до заданих меж координат. У вихідний набір входять:

- точки, що мають бути відвідані БПЛА;
- матриця рельєфу;
- проєкції точок на поверхню рельєфу.

11. Функція `points_show(points, ground_points, ground)`

Ця функція відповідає за візуалізацію рельєфу, точок і їхнього взаємного розташування. Вона будує тривимірну поверхню, наносить на неї точки маршруту та додає вертикальні зв'язки між точками та рельєфом. Це забезпечує наочне уявлення про складність геометрії середовища.

12. Функція `draw_progress(points, ground, pop, D, best_history, gen)`

Функція створює проміжні зображення, що демонструють перебіг еволюції. Вона показує:

- поточний найкращий маршрут,
- розташування точок на рельєфі,
- динаміку зміни найкращого фітнесу у вигляді графіка.

Таким чином, користувач отримує змогу відстежувати хід оптимізації, виявляти стадії стагнації та оцінювати ефективність параметрів ГА.

Подані функції утворюють цілісну програмну архітектуру, орієнтовану на ефективне моделювання задачі побудови оптимального маршруту БПЛА у тривимірному просторі з урахуванням рельєфу. Реалізована система поєднує глобальний стохастичний пошук із локальною детермінованою оптимізацією, що забезпечує високу якість маршрутів і стійкість до потрапляння в локальні мінімуми. Детальний опис компонентів дозволяє легко розширювати або модифікувати функціонал залежно від вимог до практичних задач.

3.3 Генерація вхідних даних

Для проведення моделювання маршруту безпілотного літального апарата було розроблено процедуру автоматичної генерації вхідних даних, яка забезпечує формування тривимірної сцени з контрольними точками та модельованим рельєфом місцевості. Такий підхід дає змогу відтворити умови просторої навігації з урахуванням геометричних обмежень середовища та забезпечує коректність розрахунку просторових відстаней при подальшій оптимізації маршруту.

3.3.1 Просторові межі моделювання

Усі дані нормуються до єдиного контейнера тривимірного простору, визначеного діапазонами:

$$x_i \in [0,1000], y_i \in [0,1000], z_i \in [0,100], i = 1..m.$$

Таке масштабування спрощує візуалізацію та дозволяє контролювати співвідношення між горизонтальними координатами та висотою рельєфу, що є важливим для забезпечення реалістичності побудованих маршрутів.

3.3.2. Генерація набору контрольних точок

Контрольні точки маршруту генеруються випадковим чином у внутрішньому локальному просторі. Стартова точка жорстко фіксується в координаті $((0,0))$, що моделює початкову позицію БПЛА. Інші точки рівномірно розподіляються у межах:

$$x_i \in [-2,2], y_i \in [-2,2], z_i \in [5,10], i = 1..m.$$

Такий підхід дозволяє зберегти геометричну різноманітність траєкторій до проведення перетворення координат у глобальний простір.

3.3.3 Побудова рельєфу місцевості

Як штучна модель рельєфу використовується гладка поверхня, згенерована за допомогою стандартної MATLAB-функції **peaks**. Ця поверхня забезпечує наявність чергування локальних підвищень і западин, що дозволяє імітувати реалістичний ландшафт. Рельєф формується на сітці локальних координат і відповідно масштабується до глобальних меж простору.

3.3.4. Встановлення висоти точок над рельєфом

Щоб уникнути розміщення контрольних точок під поверхнею рельєфу, їхні висотні координати коригуються відповідно до значень рельєфної функції в проєкціях на площину (XY) . Для кожної точки (i) обчислюється висота поверхні $(z_{ground,i})$, після чого її координата (z_i) піднімається над рельєфом:

$$z'_i = z_i + z_{ground,i}, \quad i = 2..m,$$

Для стартової точки приймається $(z_1 = z_{ground,1})$, що відповідає моделі зльоту з поверхні.

3.3.5 Масштабування у глобальний простір

Після формування точок та рельєфу всі їхні координати підлягають лінійному перетворенню у глобальні межі ($[0,1000] \times [0,1000] \times [0,100]$). Це гарантує узгодженість даних, стабільність відстаней та коректну інтеграцію з матрицею евклідових відстаней, яка використовується генетичним алгоритмом.

3.3.6 Структура кінцевих даних

У результаті генерації формується три масиви:

points (n×3) — координати контрольних точок маршруту;

ground_points (n×1) — висота рельєфу під кожною точкою;

ground (N×N×3) — тривимірне представлення рельєфної поверхні у вигляді сітки.

Ця процедура забезпечує створення штучного, але структурно коректного тривимірного середовища, що дозволяє досліджувати ефективність алгоритмів оптимізації маршруту БПЛА в умовах, наближених до реальних. На рисунку 3.1 подано графічне відображення згенерованих даних.

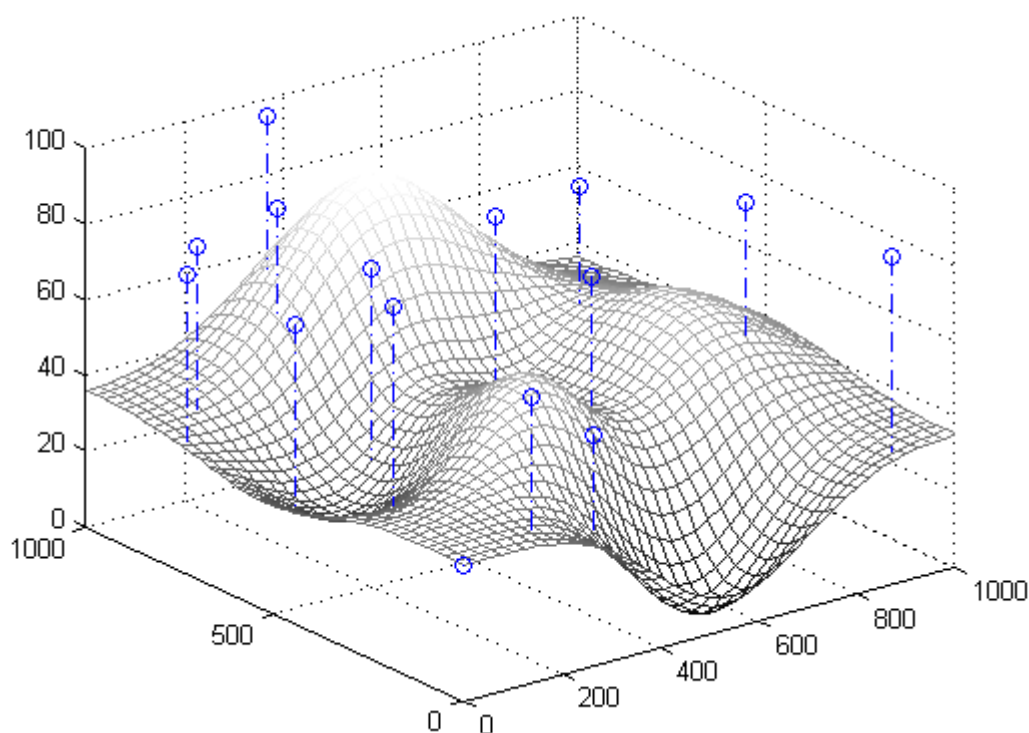


Рисунок 3.1 – Графічне відображення згенерованих даних

Джерело: запропоновано автором

3.4 Результати тестування

Для перевірки працездатності системи було згенеровано 10 точок з координатами (x, y, z) (табл. 3.2).

Таблиця 3.2 – Вхідні дані

№	x	y	z
1	0	0	32
2	881	725	87
3	115	993	64
4	524	608	60
5	869	920	81
6	615	861	100
7	843	722	72
8	878	556	83
9	271	936	86
10	338	237	90

Джерело: запропоновано автором

Координати x та y генерувалися на інтервалі від 0 до 1000, а z – від 0 до 100. Матриця відстаней між точками наведена у табл. 3.3.

Таблиця 3.3 – Матриця відстаней

	1	2	3	4	5	6	7	8	9	10
1	0	1142.7	1000.3	802.8	1266.2	1060.6	1111.2	1039.9	975.8	416.8
2	1142.7	0	812.2	377.1	195	298.9	40.6	169.8	646	730.6
3	1000.3	812.2	0	562.3	757.5	518.8	777.2	879.4	167.3	788.8
4	802.8	377.1	562.3	0	465.6	272.5	339.5	358	415.6	415.8
5	1266.2	195	757.5	465.6	0	260.7	199.3	364.4	598.3	865
6	1060.6	298.9	518.8	272.5	260.7	0	268.3	402.9	353	683.1
7	1111.2	40.6	777.2	339.5	199.3	268.3	0	170.6	611.4	701
8	1039.9	169.8	879.4	358	364.4	402.9	170.6	0	716.2	626.7
9	975.8	646	167.3	415.6	598.3	353	611.4	716.2	0	702.2
10	416.8	730.6	788.8	415.8	865	683.1	701	626.7	702.2	0

Джерело: запропоновано автором

Графічне відображення вхідних даних наведено на рис. 3.1. Як карта висот використовувалися точки поверхні, сформовані стандартною функцією MATLAB `peaks()`.

Популяція, згенерована на першому кроці роботи генетичного алгоритму, представлена у табл. 3.4.

Таблиця 3.4 – Початкова популяція

№	Хромосоми											Фітнес функція
1	1	6	3	8	7	2	5	4	10	9	1	5424.5
2	1	6	4	9	5	10	2	3	8	7	1	6915.9
3	1	5	2	9	8	6	4	3	7	10	1	5956
4	1	2	9	8	5	7	10	3	6	4	1	6152.5
5	1	2	8	9	5	4	3	6	7	10	1	5559.8
6	1	3	10	4	7	8	5	9	6	2	1	5472.4
7	1	10	2	4	9	3	6	5	8	7	1	4533.2
8	1	9	3	7	5	2	6	8	4	10	1	4297
9	1	2	6	10	7	4	9	3	5	8	1	5909.9
10	1	2	8	7	5	9	4	6	10	3	1	5441

Джерело: запропоновано автором

Перша й остання точка в маршруті повинні збігатися, оскільки БПЛА має повернутися в початкову точку після обльоту інших точок. Як функцію придатності використовувалася довжина відповідного маршруту, але найбільш пристосованими вважалися особини з найменшим значенням функції придатності. На рис. 3.2 у графічній формі наведені найбільш пристосовані особини популяцій, отриманих на 10, 100 епосі оптимізації.

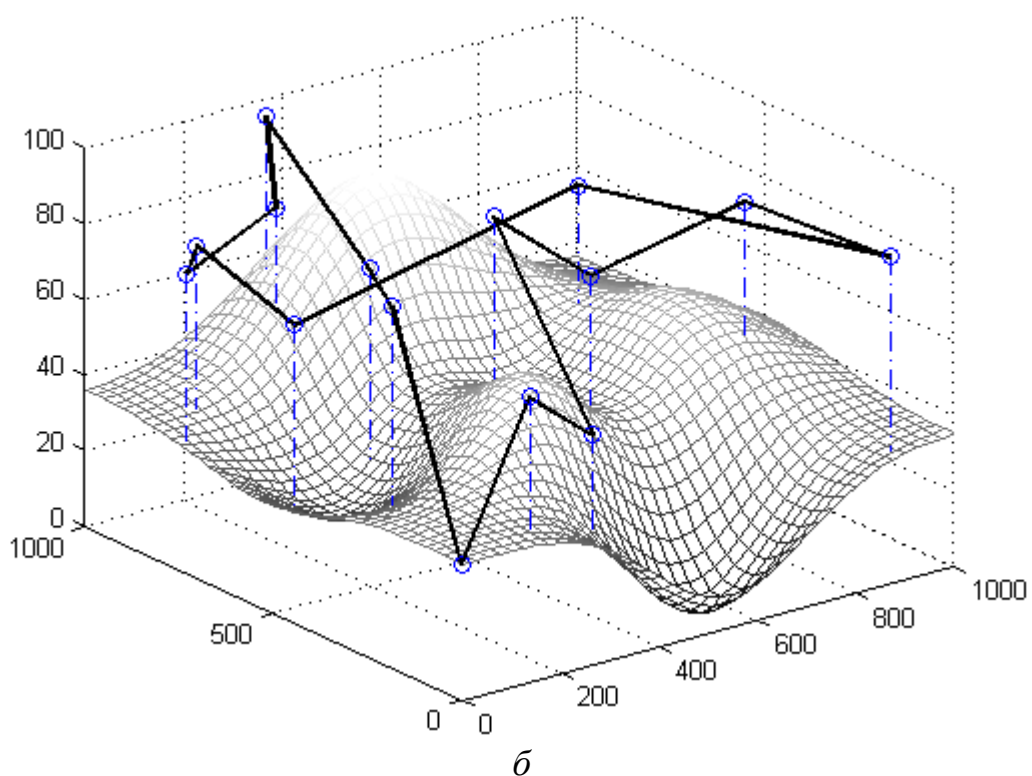
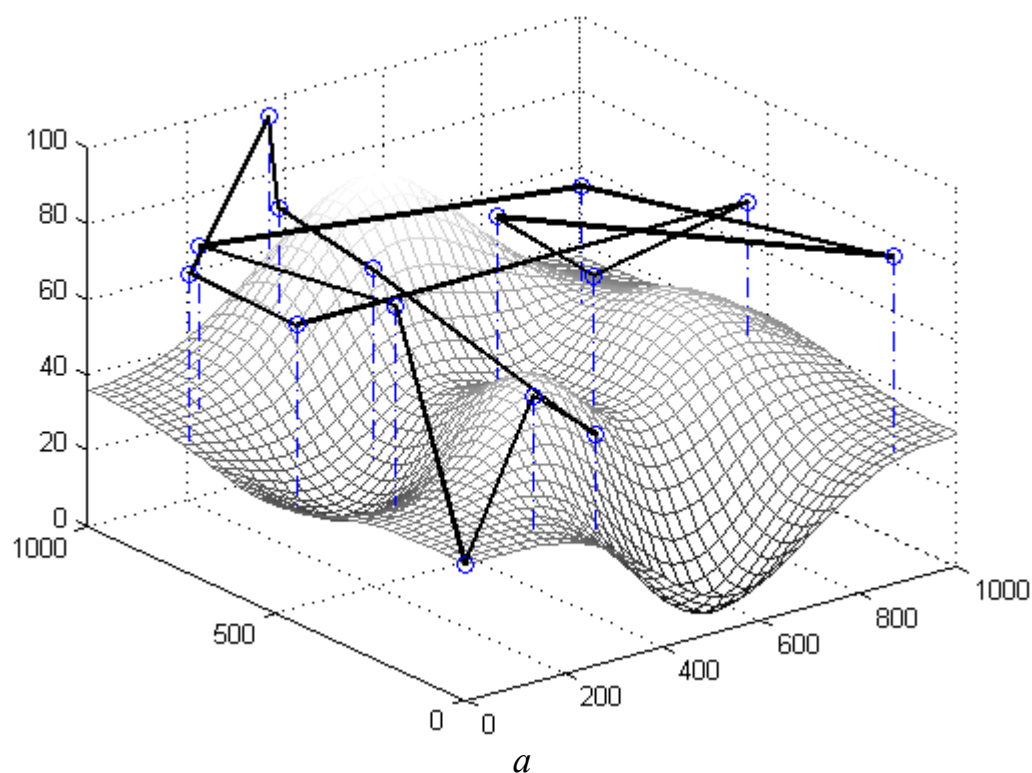


Рисунок 3.2 – Графічне відображення найбільш пристосованих особин популяцій, отриманих на а) 10 епосі, б) 100 епосі генетичного алгоритму

Джерело: запропоновано автором

Графік залежності значення функції придатності найбільш пристосованої особини від номера епохи наведено на рис. 3.3.

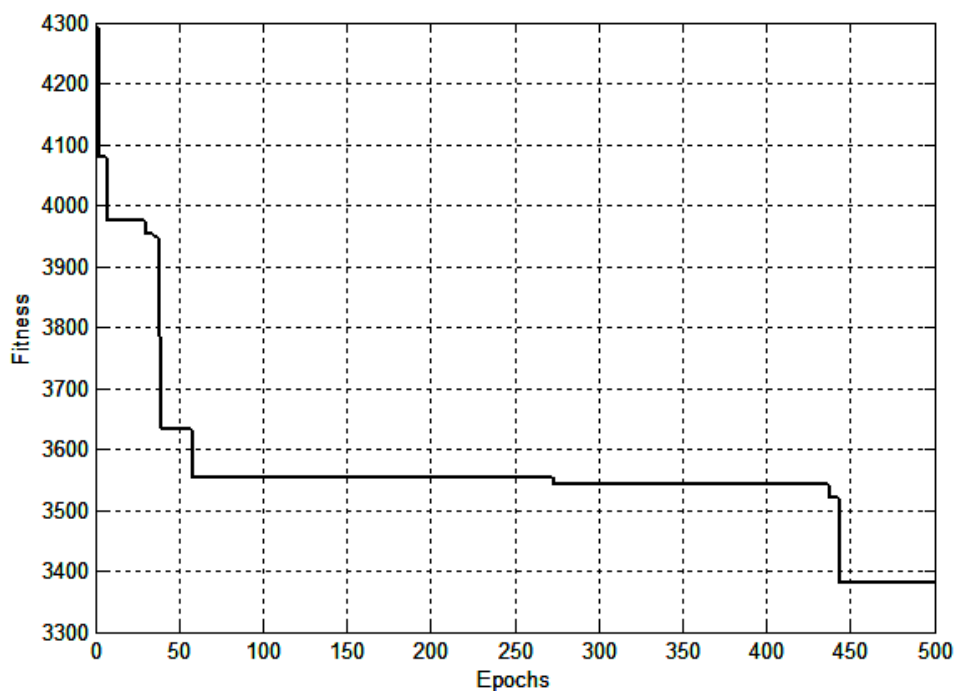


Рисунок 3.3 – Графік залежності функції придатності від епохи

Під час оптимізації використовувалися такі параметри генетичного алгоритму:

- Кросинговер – двоточковий (до нової популяції потрапляють обидва нащадки);

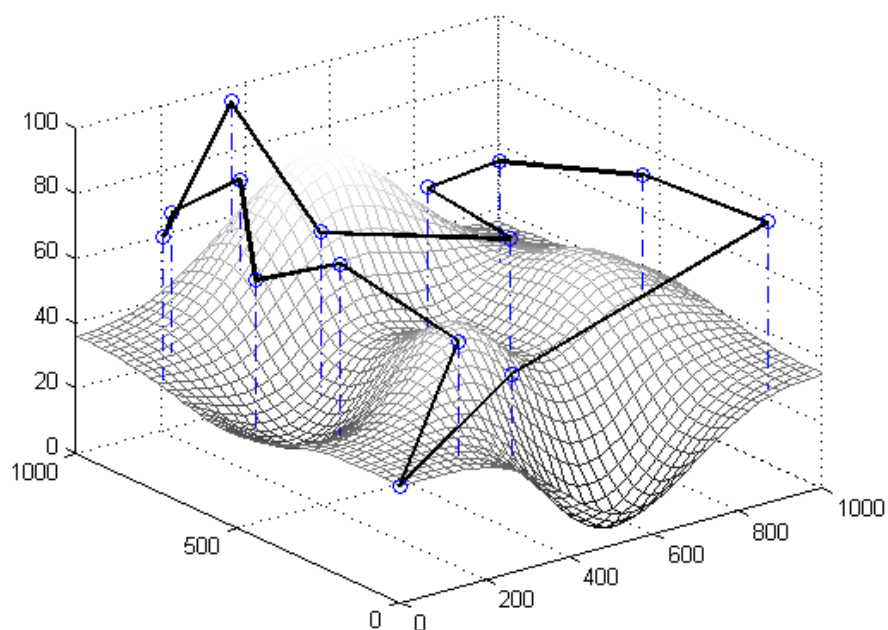


Рисунок 3.4 – Графічне відображення найбільш пристосованої особини популяцій, отриманої на 500 епосі

- Імовірність мутації – 0,05 (ген, для якого проводиться мутація, міняється місцями з іншим випадково обраним геном);
- Критерій зупину – досягнення 500 епохи.

Аналіз графіка показує, що на 500 епосі найбільш пристосована особина в популяції мала таку хромосому, графічне відображення якої наведене на рис. 3.4:

1 10 4 8 2 7 5 6 9 3 1

а, значення функції придатності знизилося до 3381.6.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було розроблено комплекс моделей, методів та програмних засобів інформаційної технології інтелектуального планування маршруту безпілотного літального апарату (БПЛА). У процесі дослідження було розв'язано такі основні задачі.

1. У роботі проведено детальний аналіз сучасних підходів до планування маршрутів безпілотних літальних апаратів, зокрема детермінованих, евристичних та метаевристичних методів. Розглянуто класичні алгоритми оптимізації, такі як повний перебір, жадібні алгоритми, методи динамічного програмування, а також їх обмеження при зростанні розмірності задачі. Окрему увагу приділено задачі комівояжера як одній з базових NP-складних задач комбінаторної оптимізації, що широко застосовується для моделювання маршрутів руху. Аналіз показав, що для практичних задач маршрутизації БПЛА з великою кількістю точок та обмежень доцільно застосовувати еволюційні та популяційні алгоритми, зокрема генетичні алгоритми, які забезпечують прийнятну якість розв'язку за обмежений час.

2. У межах кваліфікаційної роботи сформульовано математичну модель задачі планування маршруту БПЛА, яка враховує множину контрольних точок, відстані між ними, початкову та кінцеву позиції апарата, а також систему обмежень, пов'язаних з тривалістю польоту, дальністю, енергетичними ресурсами та особливостями експлуатації БПЛА. Модель дозволяє формалізувати задачу мінімізації загальної довжини маршруту або часу польоту при збереженні допустимості всіх обмежень. Запропонований підхід створює основу для подальшого застосування алгоритмів оптимізації та програмної реалізації.

3. На основі розробленої математичної моделі здійснено зведення задачі планування маршруту БПЛА до задачі комівояжера. Контрольні точки маршруту інтерпретуються як вершини графа, а можливі траєкторії польоту — як ребра з відповідними вагами, що відображають відстань або інші критерії оптимальності. Таке представлення дозволило застосувати відомі методи та алгоритми для розв'язання задачі комівояжера з урахуванням специфіки польоту БПЛА. Модель

забезпечує гнучкість у додаванні нових обмежень і критеріїв оптимізації, що є важливим для реальних прикладних задач.

4. У роботі розроблено генетичний алгоритм, адаптований до задачі планування маршруту БПЛА. Визначено спосіб кодування особин популяції, функцію пристосованості, оператори селекції, схрещування та мутації. Особливу увагу приділено врахуванню обмежень задачі при оцінюванні якості рішень та формуванні нових поколінь. Запропонований алгоритм дозволяє знаходити ефективні маршрути навіть у випадках великої кількості контрольних точок, забезпечуючи баланс між якістю розв'язку та обчислювальними витратами. Результати експериментів підтвердили доцільність використання генетичних алгоритмів для інтелектуального планування маршрутів.

5. На основі розроблених моделей та алгоритмів реалізовано програмну інформаційну систему, яка забезпечує автоматизоване формування маршруту польоту БПЛА. Система дозволяє задавати початкові параметри задачі, формувати множину контрольних точок, виконувати оптимізацію маршруту та отримувати результат у зручному для користувача вигляді. Програмна реалізація є модульною, що спрощує її подальше розширення та адаптацію до різних типів БПЛА і умов експлуатації. Розроблена система може бути використана як основа для створення прикладних рішень у сфері моніторингу, картографування та спостереження.

6. Для оцінювання ефективності та коректності роботи розробленої інформаційної системи проведено серію тестових експериментів. У ході перевірки проаналізовано здатність системи формувати допустимі маршрути, дотримуючись заданих обмежень, а також стабільність роботи генетичного алгоритму при зміні параметрів задачі. Отримані результати свідчать про працездатність розробленої системи та її придатність для практичного використання. Проведені експерименти підтвердили, що запропонований підхід дозволяє ефективно розв'язувати задачу інтелектуального планування маршруту БПЛА та може бути використаний у подальших наукових дослідженнях і прикладних розробках.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Arshi Oroos et al (Editor). Unmanned Aerial Vehicles Swarm for Protecting Smart Cities: Future Trends and Challenges. – Oroos Arshi, Inam Ullah Khan, Keshav Kaushik, Nadeem Iqbal, Inam Ullah, Khadija Slimani. – Apress Media, LLC., 2025. – 867 p.
2. Bain Geoff, Blaney Mark. Managing Safety in the Drone Industry: A Practical Guide.– CRC Press, 2025. – 295 p.
3. Bankes Sarah (ed.) Drones. The Complete Manual. 14th Edition. – Future Publishing Limited, 2025. – 132 p.
4. Saini H.K., Islam S.N.M., Kumar T.A. Quantum Unmanned Aerial Vehicle: Achieving Cybersecurity and AI Strategies. – Berlin: de Gruyter, 2025. – 530 p.
5. Victoria A. The Drones New Era. – Independently published, 2025. – 382 p.
6. Zhang Z. et al. Wireless Power Transfer for Unmanned Aircraft: Fundamentals, Design, and Control. – Cambridge University Press, 2025. – 140 p.
7. White Jon (ed.) The Drones Book.14rd Edition. – Future Publishing Limited, 2024. – 132 p.
8. Serafinelli Elisa. Theorising Drones in Visual Culture: Views from the Blue. – Palgrave Macmillan, 2024. – 228 p.
9. Sadraey M.H. Unmanned Aircraft Design: A Review of Fundamentals. 2nd Edition. – Springer Cham, 2024. – 223 p.
10. Rogers James P. (Ed.). De Gruyter Handbook of Drone Warfare. – De Gruyter, 2024. – 5506 p.
11. Mehta Axaykumar, Modi Akash. Robust Sliding Mode Protocols for Formation of Quadcopter Swarm. – Springer, 2024. – 165 p.
12. L’Afflitto A. Inalhan G. Shin H. (eds.) Control of Autonomous Aerial Vehicles. Advances in Autopilot Design for Civilian UAVs. –Springer, Switzerland, 2024. – 363 p.
13. Karakoc T.H. Ozbek E. Unmanned Aerial Vehicle Design and Technology. – Springer, Switzerland, 2024. – 197 p.
14. Bor-Sen Chen. Robust H_∞ Team Formation Tracking Design Methods of Large-Scale UAV Networked Control Systems. – Boca Raton: CRC Press, 2024. – 278 p.

15. Bai L. et al. Propagation Characterization and Channel Modeling for UAV Communications. – Springer, 2024. – 400 p.
16. Bielefeldt B.R., Hartl D.J., Kobayashi M.H. Topology Optimization Via L-systems and Genetic Algorithms: Bioinspired Encoding for Generative Design. – Cambridge: Cambridge University Press, 2025. – 303 p.
17. Brownlee Jason. Genetic Algorithm Afternoon: A Practical Guide for Software Developers. – AlgorithmAfternoon.com, 2024. – 226 p.
18. Chemin Yann-Henri, Engelbrecht Andries (eds.) Genetic Algorithms: Theory, Design and Programming. – ITeXLi, 2024. – 79 p.
19. Dey N. (ed.) Applied Genetic Algorithm and Its Variants: Case Studies and New Developments. – Singapore: Springer, 2023. – 254 p.
20. Trujillo L., Winkler S.M., Silva S., Banzhaf W. (eds.) Genetic Programming Theory and Practice XIX. – Springer, 2023. – 272 p.
21. Greiner D., Gaspar-Cunha A., Hernandez-Sosa D. Evolutionary Algorithms in Engineering Design Optimization. – MDPI, 2022. – 316 p.
22. Rocha Luciana. Evolutionary Computing. – Intelliz Press, 2022. – 312 p.
23. Ventura Sebastian, Luna José, Moyano Jose M. (eds.) Genetic Algorithms. – ITeXLi, 2022. – 156 p.
24. Mustafy T., Rahman T., Siddiqui N. Engineering Data Analysis with MatLAB. – CRC Press, 2024. – 776 p.
25. Dingyu X., Feng P. MatLAB and Simulink in Action: Programming, Scientific Computing and Simulation. – Springer, 2024. – 472 p.
26. Edet Theophilus. MatLAB Programming: Advanced Data Analysis, Visualisation, and Large-Scale Applications for Research and Development. – CompreQuest Books, 2024. – 734 p.
27. Palm III William J. MatLAB for Engineering Applications. 5th Edition. – McGraw-Hill LLC, 2023. – 656 p.
28. Tranquillo Joe. MatLAB for Engineering and the Life Sciences. 2nd Edition. – Springer, 2023. – 133 p.

29. Шелехов І., Доля Є. Інтелектуальне планування маршруту безпілотного літального апарата. – Матеріали та програма міжнародної наукової конференції молодих учених «Інформатика, математика, автоматика : 2025» (Суми – Астана, 21–25 квітня 2025 року). – С. 201-202.

30. Доля Є. Еволюційна оптимізація траєкторії польоту безпілотного літального апарата. – Матеріали Всеукраїнської наукової конференції студентів і аспірантів, присвяченої Міжнародному дню студента (Суми, 17-21 листопада 2025 року). – С. 346.

ДОДАТКИ

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ

**На розробку програмної реалізації у вигляді скриптів MATLAB для
інформаційної технології інтелектуального планування маршруту
безпілотного літального апарату**

Погоджено:

Кандидат технічних наук, доцент кафедри
кібернетики та інформатики

Ігор ШЕЛЕХОВ

Суми 2025

1. Призначення й мета створення програмної реалізації

1.1 Призначення програмної реалізації

Програмна реалізація у вигляді набору скриптів MATLAB призначена для автоматизованого формування, оптимізації та візуалізації маршрутів безпілотного літального апарату (БПЛА) шляхом зведення задачі планування маршруту до задачі комівояжера (TSP) та застосування генетичних алгоритмів (ГА) для пошуку оптимального або близького до оптимального маршруту з урахуванням заданих обмежень (зони заборони, дальність польоту, часові вікна, пріоритети точок тощо).

1.2 Мета створення програмної реалізації

Метою є розробка робочого програмного комплексу в MATLAB, який забезпечить:

- формалізацію та моделювання задачі інтелектуального планування маршруту БПЛА;
- впровадження та налаштування генетичного алгоритму для ефективного пошуку траєкторій;
- зручні інтерфейси для вводу вихідних даних, візуалізації маршрутів та оцінювання якості рішень;
- проведення тестування та експериментальної валідації алгоритмів на наборі тестових прикладів.

1.3 Цільова аудиторія

Ціловою аудиторією є

- наукові та інженерні працівники, що займаються розробкою алгоритмів планування маршрутів;
- студенти та здобувачі, які вивчають методи оптимізації та еволюційні алгоритми;
- розробники програмного забезпечення для БПЛА, що потребують прототипів та інструментів для тестування маршрутів.

2. Вимоги до програмної реалізації

2.1 Вимоги до структури й функціонування програмної реалізації

Програмний пакет повинен мати модульну структуру і складатися з окремих скриптів/функцій MATLAB, що виконують наступні ролі:

- модуль підготовки даних (генерація/завантаження координат точок, зон обмежень, параметрів БПЛА);
- модуль математичного опису задачі (обчислення матриці відстаней/вартостей, задання обмежень);
- модуль представлення маршруту у вигляді задачі комівояжера;
- модуль генетичного алгоритму (ініціалізація популяції, оператори відбору, кросовера, мутації, елітарність, критерії зупинки);
- модуль локальної оптимізації (опціонально: 2-opt, 3-opt або інші покращувальні процедури);
- модуль візуалізації (побудова карти/плану, відображення зон обмежень, анімація траєкторії, порівняння варіантів);
- модуль експериментального тестування та оцінки (логування результатів, побудова графіків збіжності, обчислення метрик якості);
- модуль збереження/завантаження результатів (файли .mat, CSV, графічні зображення).

2.2 Вимоги до інтерфейсів вводу/виводу

- Вхідні дані: файл .mat або .csv з координатами точок (ID, X, Y, опціонально висота), параметри обмежень (радіуси зон, полілінії зон заборони), характеристики БПЛА (макс. дальність, швидкість), параметри ГА (розмір популяції, ймовірності кросовера/мутації, критерій зупинки).
- Вихідні дані: оптимальний маршрут (послідовність точок), довжина маршруту, час виконання, метрики (помилка від нижньої межі, кількість ітерацій), графічні файли (PNG), файл з результатами (.mat/.csv) для подальшої обробки.

- Комунікація між модулями реалізується через функції з чітко описаними аргументами та поверненими значеннями.

2.3 Вимоги до документації

- Кожний скрипт/функція повинен містити заголовковий блок з описом призначення, вхідними параметрами, вихідними значеннями та прикладом виклику.
- Загальна користувацька інструкція (README) з описом встановлення, порядку запуску, формату вхідних файлів та прикладами експериментів.
- Коментарі в коді мають бути інформативними та структурованими.

3 Технічні вимоги

3.1 Системні вимоги

- Платформа: Windows / Linux / macOS;
- ПЗ: MATLAB (рекомендована версія R2021a або новіша);
- Обов'язкові тулбокси: Optimization Toolbox (опціонально), Statistics and Machine Learning Toolbox (опціонально). Реалізація має бути можливою й без спеціальних тулбоксів (чистий MATLAB), але з використанням стандартних засобів MATLAB.
- Мінімальні ресурси: 8 ГБ оперативної пам'яті (рекомендується 16 ГБ при великих задачах), багатоядерний процесор для прискорення обчислень (обробка паралельних сценаріїв опціонально).

3.2 Вимоги до якості реалізації

- Коректність обчислень (перевірка на контрольних прикладах);
- Стабільність роботи при різних обсягах вхідних даних;
- Час виконання: алгоритми мають бути оптимізовані з урахуванням можливості запуску на настільних ПК;

- Розширюваність: код повинен дозволяти легко підставляти інші оператори ГА, локальні оптимізатори або метрики.

3.3 Вимоги до безпеки та надійності

- Захист від некоректних вхідних даних (перевірки, валідація);
- Обробка винятків та інформативні повідомлення про помилки;
- Можливість відновлення проміжних результатів (checkpointing) при тривалих експериментах.

4 Функціональні вимоги

4.1 Основні функції

- Зчитування/генерація координат цілей та зон обмежень;
- Побудова матриці відстаней та перетворення задачі планування в TSP-формат;
- Запуск генетичного алгоритму з параметричними налаштуваннями;
- Виконання локального покращення для обраних індивідів;
- Візуалізація остаточного маршруту та проміжних ітерацій (опціонально анімація);
- Порівняння декількох запусків з різними параметрами, побудова графіків збіжності (значення цільової функції від ітерації);
- Збереження детальних логів експериментів.

4.2 Додаткові (опціональні) функції

- Паралельне виконання кількох запусків ГА з різними початковими умовами;
- Інтеграція з картографічними бекендами (імпорт реальної карти — опціонально);
- Підтримка обмежень на час / часових вікон відвідування точок;
- Побудова сценаріїв „що-якщо” для аналізу чутливості параметрів.

5 Вимоги до тестування та валідації

5.1 Вид тестів

- Функціональні тести для кожного модуля (утилітні скрипти з тестовими наборами даних);
- Тестування на стандартних TSP-прикладках (малий набір, середній, великий) для перевірки коректності та порівняння з базовими рішеннями (наприклад, випадкова реалізація, жадібний алгоритм);
- Навантажувальні тести (збільшення кількості точок до 100–1000 для оцінки масштабованості);
- Тестування на стійкість до аномальних вхідних даних.

5.2 Критерії приймання

- Коректність вихідного формату та сумісність результатів із заданими специфікаціями;
- Досягнення прийнятного рівня якості рішення: наприклад, відносне відхилення від простих базових рішень не більше заданої межі в типовому наборі тестів;
- Наявність повної документації та прикладів запуску;
- Відсутність критичних помилок при типових сценаріях використання.

6 Склад і зміст робіт зі створення програмної реалізації

Докладний опис етапів роботи зі створення застосунку наведено в таблиці А.1.

7 Вимоги до введення в експлуатацію

7.1 Перенесення та запуск проекту

- Надати інструкції з розгортання проекту в MATLAB (вказати версію);

- Перевірити роботу на тестовому наборі даних;
- Надати приклади запуску (script_run_examples.m) і файли з результатами.

Таблиця А.1 – Етапи створення програмної реалізації (скрипти MATLAB)

№	Склад і зміст робіт	Строк розробки (дні)
1	Формулювання технічних вимог і планування розробки	2
2	Підготовка тестових наборів даних (генератори, приклади)	3
3	Розробка структури проекту та шаблонів функцій	2
4	Реалізація модуля підготовки даних	3
5	Розробка модуля побудови матриці відстаней і формалізації TSP	3
6	Розробка базового генетичного алгоритму (ядро)	6
7	Додавання операторів кросовера/мутації та параметричних налаштувань	4
8	Реалізація локальних методів покращення (2-opt та ін.)	3
9	Модуль візуалізації та анімації маршрутів	4
10	Модуль логування та збереження результатів	2
11	Розробка скриптів для експериментів та автоматичних прогонів	3
12	Оптимізація продуктивності та рефакторинг коду	4
13	Написання документації (README, приклади, опис функцій)	3
14	Функціональні та навантажувальні тести	4
15	Підготовка звіту про експерименти та демонстраційних матеріалів	3
—	Загальна тривалість робіт	47 днів

7.2 Підготовка користувацької документації

- Короткий посібник користувача з прикладами запуску;
- Технічний опис архітектури кодової бази;
- Опис можливих розширень та рекомендації з оптимізації.

8 Примітки

- Джерело алгоритмів та методик — авторська розробка з опорою на загальноприйняті підходи до TSP та генетичних алгоритмів.
- Код має бути придатним для подальшої інтеграції до більш складної інформаційної системи (наприклад, веб-інтерфейсу або вбудованого контролера).

ДОДАТОК Б

ПЛАНУВАННЯ РОБІТ

Планування робіт із розроблення та впровадження застосунку здійснюється з використанням календарно-ресурсного підходу. Для візуалізації тривалості етапів, їх послідовності та взаємозв'язків використовується діаграма Ганта. Такий метод дає змогу структуровано організувати процес виконання проекту, визначити ключові завдання, оцінити строки та забезпечити контроль виконання. На рисунку Б.1 зображено Діаграму Ганта, у вигляді календарю

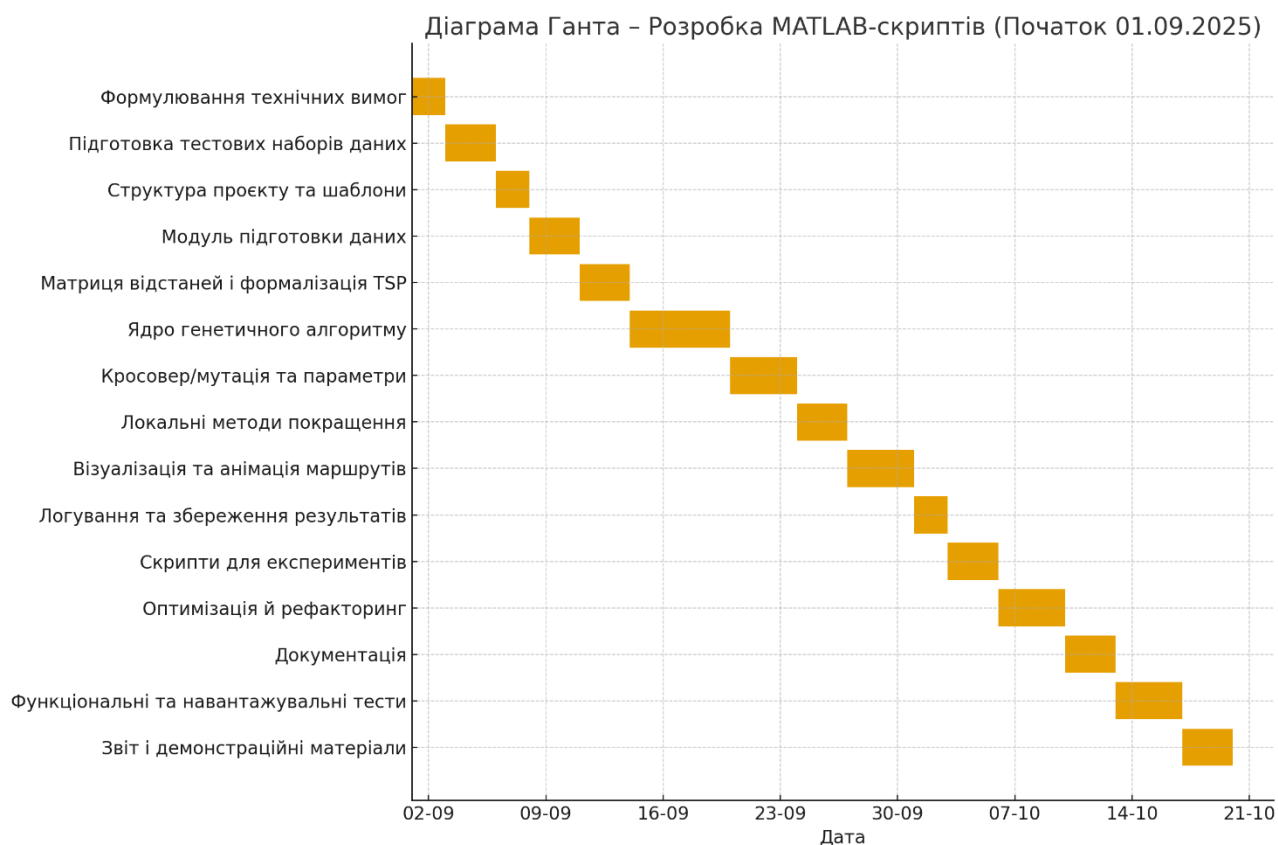


Рисунок Б.1 – Діаграма Ганта

Джерело: запропоновано автором

Зростання потреб у використанні безпілотних літальних апаратів для моніторингу, логістики, агротехнологій та інших сфер обумовлює актуальність створення інтелектуальних систем автоматизованого планування маршрутів.

Розроблення програмної реалізації, що поєднує математичні моделі задачі комівояжера та методи еволюційних обчислень, вимагає ретельного планування етапів, визначення строків і відповідальних виконавців.

Для забезпечення високої якості кінцевого продукту важливим елементом управління проектом є формування реєстру ризиків, який дозволяє завчасно ідентифікувати потенційні загрози, оцінити їх вплив на строки та результати розробки, а також визначити ефективні стратегії управління.

Основні компоненти реєстру ризиків включають: опис ризику, ймовірність виникнення, потенційний вплив на проєкт, стратегії мінімізації та відповідальних осіб. У таблиці Б.1 наведено сформований реєстр ризиків, актуальний для розробки системи інтелектуального планування маршруту БПЛА.

Приклади потенційних ризиків для проєкту

1. Технічні ризики

1.1. Помилки у реалізації генетичного алгоритму

Причини: неправильний вибір операторів, некоректна селекція, помилки у параметрах.

Наслідки: нерелевантні або нестабільні маршрути, неможливість знайти оптимальне рішення.

Запобігання: тестування на контрольних наборах, порівняння з еталонними TSP-розв'язувачами.

1.2. Відмова або помилки у побудові матриці відстаней

Причини: логічні помилки, некоректні координати, неправильне округлення.

Наслідки: неправильний пошук маршруту, невідповідність оптимізаційних результатів.

Запобігання: перевірка функцій генерації, дублювання розрахунків різними методами.

1.3. Низька продуктивність алгоритму при великомасштабних даних

Причини: висока складність ГА, надто велика популяція або кількість ітерацій.

Наслідки: тривалий розрахунок, неможливість роботи в реальному часі.

Запобігання: оптимізація ГА, параметричне налаштування, профілювання.

Таблиця Б.1 – Реєстр ризиків проєкту

Опис ризику	Імовірність	Вплив	Стратегії управління	Відповідальні
Помилки у реалізації генетичного алгоритму (некоректна еволюція, локальні мінімуми)	Середня	Високий	Модульне тестування, валідація результатів на штучних даних, код-рев'ю	Розробник
Некоректна побудова матриці відстаней (помилки у розрахунках)	Середня	Високий	Перевірка формул, подвійна валідація даних, тестові набори	Розробник
Надмірний час обчислень при великій кількості точок маршруту	Висока	Середній	Оптимізація ГА, кешування, паралельні обчислення	Розробник
Збої MATLAB або середовища виконання	Низька	Середній	Збереження станів, регулярні бекапи, використання стабільних версій	Розробник
Невідповідність кінцевої реалізації технічним вимогам	Середня	Високий	Узгодження етапів розробки, контрольні точки, проміжні демонстрації	Керівник проєкту
Затримка у виконанні окремих етапів робіт	Висока	Середній	Перепланування, буфер часу, контроль виконання щотижневих задач	Керівник проєкту
Нестача або некоректність тестових даних	Середня	Середній	Створення генераторів тестових наборів, штучних даних	Розробник
Продуктивність системи нижча за очікувану	Середня	Середній	Профілювання коду, оптимізація критичних ділянок	Розробник
Відсутність повної документації для користувачів	Середня	Низький	Підготовка README, коментарів до коду, прикладів використання	Розробник
Непередбачувані зміни вимог під час розробки	Низька	Високий	Чітка фіксація вимог, гнучке планування, регулярні консультації	Керівник проєкту
Ризики, пов'язані з апаратними платформами БПЛА (формати, обмеження енергоспоживання)	Низька	Середній	Універсальні формати даних, параметризація моделі	Розробник
Неможливість інтеграції модулів у єдину систему	Низька	Середній	Розробка уніфікованих API, тестування інтеграції після кожного етапу	Розробник

Джерело: запропоновано автором.

1.4. Критичні помилки у коді MATLAB

Причини: недопрацьовані модулі, неправильні залежності.

Наслідки: аварійне завершення роботи, некоректні результати.

Запобігання: модульні тести, код-рев'ю.

2. Організаційні ризики

2.1. Затримки у виконанні етапів розробки

Причини: технічні труднощі, непередбачувані проблеми.

Наслідки: порушення календарного графіка, зсув кінцевих строків.

Запобігання: буфер часу, щотижневий контроль завдань.

2.2. Зміна вимог у процесі

Причини: уточнення потреб користувачів, зміна контексту застосування БПЛА.

Наслідки: необхідність перероблення модулів.

Запобігання: чітка фіксація ТЗ, гнучке управління.

3. Ризики планування

3.1. Недооцінка складності розробки

Причини: складність алгоритмів, високі вимоги до оптимізації.

Наслідки: збільшення часу, перевищення навантаження.

Запобігання: декомпозиція задач, аналіз складності.

3.2. Вихід за межі строків або ресурсів

Причини: труднощі інтеграції модулів, помилки в коді.

Наслідки: ризик несвоєчасного завершення.

Запобігання: регулярний моніторинг прогресу.

4. Ризики, пов'язані з практичним використанням системи

4.1. Некоректні вхідні дані від операторів

Причини: помилки у введенні координат, відсутність валідації.

Наслідки: неправильний маршрут, аварійна поведінка БПЛА.

Запобігання: валідація даних, підказки та попередження.

4.2. Невідповідність результатів можливостям конкретного БПЛА

Причини: перевищення дальності, зневага до перешкод.

Наслідки: невиконання місії, ризики для апаратури.

Запобігання: урахування технічних параметрів дронів.

5. Зовнішні ризики

5.1. Нестабільність зовнішніх сервісів (карти, метеодані)

Причини: недоступність API, оновлення форматів.

Наслідки: неповні дані для планування.

Запобігання: fallback-механізми, дублюючі джерела.

5.2. Надзвичайні ситуації (блекаути, кібератаки)

Причини: зовнішні фактори.

Наслідки: тимчасова зупинка розробки або тестування.

Запобігання: резервні середовища, шифрування даних.

Аналіз ризиків показав, що найбільш критичними для розробки системи інтелектуального планування маршруту БПЛА є технічні ризики, пов'язані з реалізацією генетичного алгоритму, побудовою матриці відстаней, продуктивністю та можливими помилками в коді. Значну загрозу становлять також організаційні ризики — зміни вимог, затримки виконання етапів та похибки у вхідних даних. Формування реєстру ризиків дає змогу завчасно розробити механізми реагування і забезпечити стабільність розробки, підвищити надійність і якість програмної реалізації.

ДОДАТОК В
КОПІЇ ПУБЛІКАЦІЙ



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
МІЖНАРОДНИЙ УНІВЕРСИТЕТ «АСТАНА»



ІНФОРМАТИКА, МАТЕМАТИКА, АВТОМАТИКА

ІМА - 2025

**МАТЕРІАЛИ
та програма**

**МІЖНАРОДНОЇ
НАУКОВОЇ КОНФЕРЕНЦІЇ
МОЛОДИХ ВЧЕНИХ**

**(Суми-Астана,
21-25 квітня 2025 року)**

**Суми,
Сумський державний університет
2025**

106. Інтерактивний віджет для автоматизації формування PDF-документів на основі даних про угоди та їх інтеграції до записів клієнтів у Zoho CRM

Автор: здобувач Карпенко А.М.,
Керівник – доцент Боровик В.О.

107. Машинне навчання для симетричних шифрів

Автор: здобувач Мельник В.В.,
Керівник – старший викладач Лаврик Т.В.

108. Наскрізне шифрування в месенджерах

Автор: здобувач Одарченко Д.Д.,
Керівник – старший викладач Лаврик Т.В.

109. Інтелектуальне планування маршруту безпілотного літального апарата

Автори: здобувач Доля Євген,
доцент Шелехов Ігор

110. Інформаційна технологія оцінки архітектурних паттернів з використанням показників безпеки вебдодатків Java

Автор: здобувач Савостян В.В.,
Керівник – доцент Любчак В. О.

111. Гібридний цифровий двійник розумного будинку

Автор: здобувач Гонор І. С.

112. Методи машинного навчання в системах розпізнавання наземних об'єктів автономними БПЛА

Автор: здобувач Алещенко М.М.,
Керівник – професор Довбиш А.С.

113. Інформаційно-екстремальна інтелектуальна технологія комп'ютерного діагностування остеологічних захворювань та патологій людини

Автор: здобувач Колісник О.

Інтелектуальне планування маршруту безпілотної літальної апарата

Доля Євген¹, ІСТ-2401м, здобувач;
Шелехов Ігор^{1,2}, доцент

¹Кафедра кібернетики та інформатики Сумського національного аграрного університету, Суми, Україна

² Кафедра комп'ютерних наук Сумського державного університету, Суми, Україна

Сучасний стан розвитку технологій керування безпілотними літальними апаратами (БПЛА) передбачає активне впровадження програмних рішень для автономної навігації та планування маршрутів. З огляду на широке використання БПЛА в таких галузях як оборона, сільське господарство, екологічний моніторинг, геодезія, актуальним є створення інформаційної системи, яка дозволяє ефективно будувати оптимальні маршрути польоту з урахуванням рельєфу місцевості, обмежень та цільових точок.

Метою дослідження є розробка інформаційної системи планування маршруту БПЛА з використанням генетичного алгоритму, що дозволяє вирішити задачу комівояжера у тривимірному просторі. Об'єкт дослідження – процес планування маршруту БПЛА з мінімізацією довжини або іншого критерію оптимальності. Предмет дослідження – генетичні алгоритми як метод глобальної оптимізації. Методологічною основою дослідження є еволюційний підхід до пошуку рішень у багатовимірному просторі можливих маршрутів.

У межах дослідження проаналізовано поняття БПЛА, здійснено їх класифікацію за функціональними та конструктивними ознаками, а також розглянуто технічні характеристики та структуру типової системи керування. Особливу увагу приділено ролі автоматизованого планування маршрутів, яке виступає ключовим елементом автономної навігації. Встановлено, що ефективність навігації значною мірою залежить від обраного алгоритму маршрутизації, який повинен враховувати просторову конфігурацію цільових точок, геометрію місцевості, обмеження по висоті польоту, енергоспоживанню та специфіку виконуваних завдань.

Оснoву інформaційної системи становить генетичний алгоритм, до складу якого входять: генерація початкової популяції маршрутів, обчислення функції пристосованості (довжини маршруту), застосування операторів кросинговеру (частково відображене схрещування), мутації (випадкова перестановка точок) та селекції (вибір найкращих рішень). Для запобігання передчасної конвергенції реалізовано перевірку на ідентичність особин та адаптивну мутацію.

Програмну реалізацію системи виконано в середовищі MATLAB. Застосування цієї платформи обумовлено її потужними засобами для чисельного моделювання, візуалізації та інтеграції з іншими програмними системами. Розроблено окремі функціональні модулі: генерація координат опорних точок, створення початкової популяції, обчислення матриці відстаней, реалізація генетичних операторів, графічне відображення маршруту.

У рамках моделювання використано 10 випадково згенерованих точок у тривимірному просторі з координатами (x, y, z), які були нанесені на карту рельєфу місцевості. В результаті 500 ітерацій алгоритму спостерігалось зменшення довжини маршруту з понад 6900 до 3381.6 умовних одиниць. Найбільш пристосовані особини демонстрували стабільну еволюцію маршрутів із поступовим покращенням показників. Графіки залежності значень функції пристосованості від епох ілюстрували ефективну роботу алгоритму.

Важливою особливістю підходу стало те, що оптимізація проводилася не лише у площині (x, y), як у класичних задачах комівояжера, а з урахуванням висоти (координати z), що суттєво ускладнює розв'язання задачі, але наближає її до реальних умов експлуатації БПЛА.

Практичне значення дослідження полягає в можливості застосування запропонованої інформаційної системи для планування маршрутів автономних БПЛА при виконанні різнопланових завдань: аерофотозйомки, моніторингу інфраструктури, пошуку та рятування, дослідження важкодоступних територій. Гнучкість параметрів алгоритму дозволяє адаптувати його до конкретних сценаріїв і середовищ функціонування.

Подальші напрями досліджень включають розширення функціоналу за рахунок гібридизації з іншими методами оптимізації, такими як мурашині алгоритми, диференціальна еволюція, а також дослідження стійкості алгоритму до випадкових збурень.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ

МАТЕРІАЛИ
ВСЕУКРАЇНСЬКОЇ НАУКОВОЇ
КОНФЕРЕНЦІЇ СТУДЕНТІВ
ТА АСПІРАНТІВ, ПРИСВЯЧЕНОЇ
МІЖНАРОДНОМУ ДНЮ СТУДЕНТА

(17-21 листопада 2025 р., м. Суми)

ЕВОЛЮЦІЙНА ОПТИМІЗАЦІЯ ТРАЄКТОРІЇ ПОЛЬОТУ БЕЗПІЛОТНОГО ЛІТАЛЬНОГО АПАРАТА

Доля Є. В., студ. 2 (магістерського) курсу ФЕІМ

Науковий керівник: доц. І.В. Шелехов

Сумський НАУ

У розробленій інформаційній системі планування маршруту безпілотного літального апарата (БПЛА) генетичний алгоритм (ГА) реалізовано як ядро оптимізаційного процесу у середовищі MATLAB, що забезпечує зручне оперування матрицями, візуалізацію рельєфу та швидке тестування параметрів еволюційного пошуку. Програмна структура складається з набору m-функцій, кожна з яких відповідає певному етапу ГА: створення популяції (`popul_gen`), обчислення функції пристосованості (`popul_fitness`), операції кросингверу (`popul_cross`), мутації (`popul_mutation`) та формування нового покоління.

Особливістю реалізації є частково відображуваний кросингвер [1], який зберігає топологію маршруту та уникає дублювання точок. Мутація виконується через перестановку двох випадкових генів, що підвищує різноманітність популяції. Функція пристосованості реалізована як обчислення загальної довжини польоту з урахуванням просторових координат, отриманих із матриці відстаней. Система підтримує візуальне відображення поверхні та траєкторій у 3D-просторі, що дає змогу контролювати процес еволюції у реальному часі.

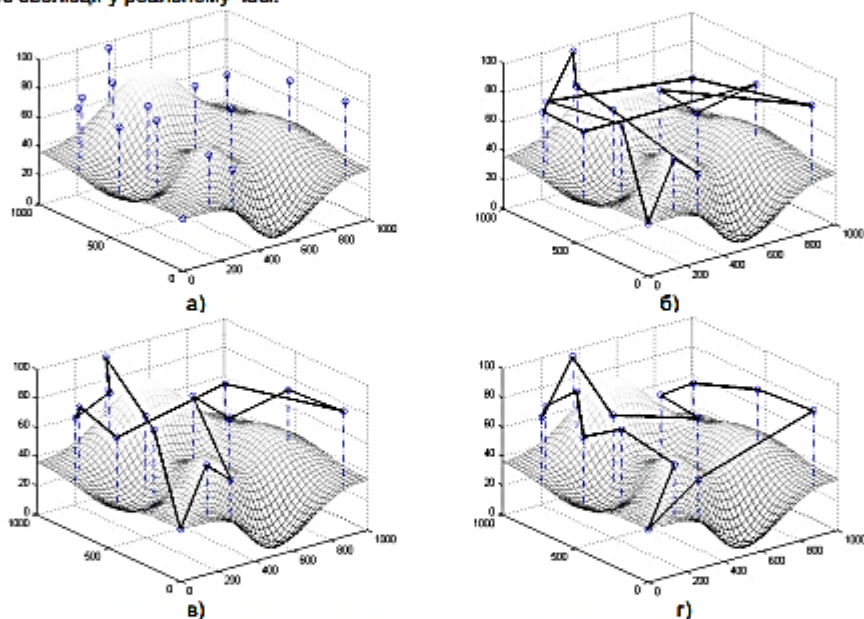


Рисунок 1 – Графічне відображення результатів роботи генетичного алгоритму:
а) згенерована поверхня і опорні точки на ній; б) відображення найбільш пристосованої особи з 10 популяції, в) 100 популяції, г) фінальної популяції

Модель дозволяє гнучко налаштовувати параметри: розмір популяції, імовірність мутації, кількість епох. Завдяки стохастичному характеру пошуку, алгоритм здатен уникати локальних мінімумів, а елітарна стратегія гарантує збереження найкращих маршрутів.

У модифікації для множинних БПЛА кожен апарат може розглядатись як окрема популяція («острівна модель»), між якими відбувається обмін оптимальними рішеннями[2]. Це забезпечує паралельну оптимізацію маршрутів і координоване покриття території без конфліктів між траєкторіями. Таким чином, програмна реалізація демонструє ефективність генетичного підходу як гнучкого інструменту інтелектуального планування місій БПЛА.

Джерела:

- 1.Xin J., Zhong J., Yang F., Cui Y., Sheng J. An Improved Genetic Algorithm for Path-Planning of Unmanned Surface Vehicle // *Sensors*. – 2019. – Vol. 19. – Art. 2640. – DOI: 10.3390/s19112640
- 2.Xu H., Niu Z., Jiang B., Zhang Y., Chen S., Li Z., Gao M., Zhu M. ERRT-GA: Expert Genetic Algorithm with Rapidly Exploring Random Tree Initialization for Multi-UAV Path Planning // *Drones*. – 2024. – Vol. 8. – Art. 367. – DOI: 10.3390/drones8080367

ДОДАТОК Г

ПРОГРАМНИЙ КОД

```

%% main.m
% Структурований і оптимізований генетичний алгоритм для оптимізації
маршруту БПЛА (3D TSP з рельєфом)
% Як користуватися: запустіть main в MATLAB (усі функції з цього
файлу – окремі функції підрозділи).

function main
clear; close all; clc;
rng('shuffle'); % випадкова ініціалізація

%%% Параметри задачі
n_points = 10; % кількість точок (включно зі
стартовою точкою 1)
coords_limits = [0 1000; 0 1000; 0 100]; % просторові межі для
відображення

%%% Параметри GA
pop_size = 100; % розмір популяції
max_gen = 500; % кількість поколінь
p_mut = 0.08; % ймовірність мутації
tourney_k = 3; % розмір турніру для селекції
elite_n = 1; % кількість еліт
use_two_opt = true; % виконувати локальний пошук 2-опт
після породження

%%% Генерація точок та рельєфу
[points, ground_points, ground] = points_gen(n_points,
coords_limits);
figure('Name','Points and ground');
points_show(points, ground_points, ground);

%%% Популяція (перестановки від 2..n); позиція 1 – старт/фініш
pop = popul_gen(n_points, pop_size);

%%% Зручно зберігати матрицю відстаней один раз
D = dist_matrix(points);

best_history = zeros(max_gen,1);
mean_history = zeros(max_gen,1);

for gen = 1:max_gen
    % Обчислити довжину маршруту (фітнес = довжина)

```

```

fit = popul_fitness(pop, D);
[min_fit, idx_min] = min(fit);
[mean_fit] = mean(fit);
best_history(gen) = min_fit;
mean_history(gen) = mean_fit;

% Лог еліту
elite = pop(idx_min,:);

% Нове покоління
new_pop = zeros(size(pop));
% Збереження еліту
new_pop(1:elite_n,:) = repmat(elite, elite_n, 1);
cur = elite_n + 1;

while cur <= pop_size
    % Селекція (турнірна)
    p1 = tournament_select(fit, tourney_k);
    p2 = tournament_select(fit, tourney_k);
    parent1 = pop(p1,:);
    parent2 = pop(p2,:);

    % Кросовер (Order Crossover, OX)
    [child1, child2] = order_crossover(parent1, parent2);

    % Мутація (інверсія) з вказаною ймовірністю
    if rand < p_mut, child1 = mutation_inversion(child1); end
    if rand < p_mut, child2 = mutation_inversion(child2); end

    % Локальний пошук 2-opt
    if use_two_opt
        child1 = two_opt_local(child1, D);
        child2 = two_opt_local(child2, D);
    end

    % Додати в популяцію
    new_pop(cur,:) = child1; cur = cur + 1;
    if cur <= pop_size
        new_pop(cur,:) = child2; cur = cur + 1;
    end
end

pop = new_pop;

% Візуалізація процесу (оновлюється кожні 20 поколінь)
if mod(gen,20)==0 || gen==1 || gen==max_gen

```

```

        fprintf('Gen %d: best = %.3f, mean = %.3f\n', gen,
best_history(gen), mean_history(gen));
        draw_progress(points, ground, pop, D, best_history, gen);
    end
end

% Показати фінальний найкращий маршрут
final_fit = popul_fitness(pop,D);
[best_val, best_idx] = min(final_fit);
best_ind = pop(best_idx,:);
opt_path = [1 best_ind 1];
opt_p = points(opt_path,:);

figure('Name','Final best path');
points_show(points, ground_points, ground); hold on;
plot3(opt_p(:,1), opt_p(:,2), opt_p(:,3), 'b-', 'LineWidth', 2);
plot3(opt_p(:,1), opt_p(:,2), opt_p(:,3), 'ro',
'MarkerFaceColor','r');
title(sprintf('Best path, length = %.3f', best_val));

figure('Name','Convergence'); plot(best_history,'-
b','LineWidth',1.5); hold on; plot(mean_history,'-r');
legend('best','mean'); xlabel('Generation'); ylabel('Path length');
grid on;

end

%% -----
% dist_matrix.m
function D = dist_matrix(points)
% Повертає матрицю евклідовських відстаней між точками (включно з
початковою 1)
% points — N x 3
N = size(points,1);
D = zeros(N);
for i=1:N
    for j=i+1:N
        D(i,j) = norm(points(i,:)-points(j,:));
        D(j,i) = D(i,j);
    end
end
end
end

%% -----
% popul_gen.m
function pop = popul_gen(n, pop_size)

```

```

% Створює популяцію перестановок для точок 2..n. Кожен індивід –
вектор розміру n-1
% Ми зберігаємо лише перестановку для точок 2..n. При формуванні
шляху додаємо 1 на початку і в кінці.
if n<=1, error('n must be >1'); end
base = 2:n;
pop = zeros(pop_size, n-1);
for k=1:pop_size
    pop(k,:) = base(randperm(n-1));
end
end

%% -----
% popul_fitness.m
function fit = popul_fitness(pop, D)
% Обчислює довжину маршруту для кожного індивіда
% pop – pop_size x (n-1), D – N x N матриця відстаней
pop_size = size(pop,1);
n_minus1 = size(pop,2);
fit = zeros(pop_size,1);
for i=1:pop_size
    route = [1 pop(i,:) 1];
    s = 0;
    for j=1:length(route)-1
        s = s + D(route(j), route(j+1));
    end
    fit(i) = s;
end
end

%% -----
% tournament_select.m
function idx = tournament_select(fit, k)
% Турнірна селекція – повертає індекс обраного індивіда (чим менший
fit – краще)
pop_size = length(fit);
cand = randi(pop_size, [1,k]);
[~,best_rel] = min(fit(cand));
idx = cand(best_rel);
end

%% -----
% order_crossover.m
function [c1,c2] = order_crossover(p1,p2)
% OX crossover для перестановок довжини m
m = length(p1);

```

```

%crossover points
pt = sort(randi(m,1,2)); a=pt(1); b=pt(2);
%c1
c1 = zeros(1,m); c1(a:b) = p1(a:b);
pos = b+1; if pos>m, pos=1; end
src = b+1; if src>m, src=1; end
while any(c1==0)
    if ~ismember(p2(src), c1)
        c1(pos) = p2(src);
        pos = pos+1; if pos>m, pos=1; end
    end
    src = src+1; if src>m, src=1; end
end
%c2 (симетрично)
c2 = zeros(1,m); c2(a:b) = p2(a:b);
pos = b+1; if pos>m, pos=1; end
src = b+1; if src>m, src=1; end
while any(c2==0)
    if ~ismember(p1(src), c2)
        c2(pos) = p1(src);
        pos = pos+1; if pos>m, pos=1; end
    end
    src = src+1; if src>m, src=1; end
end
end

%% -----
% mutation_inversion.m
function ch = mutation_inversion(ch)
% Інверсійна мутація: вибираємо два індекси та перевертаємо сегмент
m = length(ch);
if m<2, return; end
p = sort(randi(m,1,2));
ch(p(1):p(2)) = fliplr(ch(p(1):p(2)));
end

%% -----
% two_opt_local.m
function route = two_opt_local(route, D)
% Простий 2-opt локальний пошук, працює на перестановці довжини m
improved = true;
m = length(route);
while improved
    improved = false;
    best_delta = 0;
    best_i = 0; best_k = 0;

```

```

for i=1:m-2
    for k=i+1:m-1
        % оцінити виграш від 2-opt на сегменті i..k
        a = get_idx(route, i);
        b = get_idx(route, i+1);
        c = get_idx(route, k);
        d = get_idx(route, k+1);
        delta = (D(a,c) + D(b,d)) - (D(a,b) + D(c,d));
        if delta < best_delta
            best_delta = delta;
            best_i = i; best_k = k;
            improved = true;
        end
    end
end
if improved
    % виконаємо інверсію сегменту best_i+1..best_k
    i = best_i+1; k = best_k;
    route(i:k) = fliplr(route(i:k));
end
end
end

function idx = get_idx(route,pos)
% отримує реальний вузол у позиції pos у маршруті, з урахуванням
% старту 1 перед перестановкою
% route тут – перестановка для 2..n, позиції від 1 до m
idx = route(pos);
end

%% -----
% points_gen.m
function [points, ground_points, ground] = points_gen(n,
coords_limits)
% Генерація n точок з рельєфом (на основі peaks). Повертає
координати в межах coords_limits.
m = 3; % x,y,z
% Внутрішні локальні межі для побудови рельєфу
limits = [-2 2; -2 2; 5 10];
pts = rand(n,m);
pts(1,:) = zeros(1,m); % старт у (0,0)
[X1,X2] = meshgrid(limits(1,1):0.1:limits(1,2));
X3 = peaks(X1,X2);
for i=1:m
    pts(:,i) = limits(i,1) + pts(:,i).*(limits(i,2)-limits(i,1));
end

```

```

% підняти з точки над рельєфом
ground_points = peaks(pts(:,1), pts(:,2));
pts(2:end,3) = pts(2:end,3) + ground_points(2:end);
pts(1,3) = ground_points(1);
% нормалізація в coords_limits
limits(3,1) = min(min(X3));
limits(3,2) = max(max(X3));
for i=1:m
    ground(:,:,i) = coords_limits(i,1) + ( ( (i==3).*X3 + (i~=3).*X1
) - limits(i,1) ) .* (coords_limits(i,2)-coords_limits(i,1)) ./
(limits(i,2)-limits(i,1));
    points(:,i) = coords_limits(i,1) + (pts(:,i)-limits(i,1)) .*
(coords_limits(i,2)-coords_limits(i,1)) ./ (limits(i,2)-
limits(i,1));
end
% Для узгодженості, ground_points теж масштабуємо
ground_points = coords_limits(3,1) + (ground_points - limits(3,1))
.* (coords_limits(3,2)-coords_limits(3,1)) ./ (limits(3,2)-
limits(3,1));
end
%% -----
% points_show.m
function points_show(points, ground_points, ground)
mesh(ground(:,:,1), ground(:,:,2), ground(:,:,3)); hold on;
plot3(points(:,1), points(:,2), points(:,3), 'bo',
'MarkerFaceColor','b');
for j=1:size(points,1)
    plot3([points(j,1) points(j,1)], [points(j,2) points(j,2)],
[points(j,3) ground_points(j)], '-.');
end
xlabel('X'); ylabel('Y'); zlabel('Z'); grid on; view(3);
end
%% -----
% draw_progress.m
function draw_progress(points, ground, pop, D, best_history, gen)
% Малює найкращий маршрут поточної популяції
fit = popul_fitness(pop,D);
[~, idx] = min(fit);
best_ind = pop(idx,:);
opt_path = [1 best_ind 1];
opt_p = points(opt_path,:);
figure(100); clf; points_show(points,
peaks(points(:,1),points(:,2)), ground); hold on;
plot3(opt_p(:,1), opt_p(:,2), opt_p(:,3), 'b-', 'LineWidth', 1.5);
title(sprintf('Gen %d, best = %.3f', gen, best_history(gen)));
drawnow; end

```