МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ СИСТЕМ ТА МЕРЕЖ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри
_____ І.А. ЖУКОВ
«____» _____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ МАГІСТР

ЗА СПЕЦІАЛЬНІСТЮ 123 «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ»

Тема: «Топологія локальної комп'ютерної мережі стандарту *IEEE* 802.5 для наземного сегменту системи безпілотних авіаційних апаратів»

Виконавець: Ангеліна ЦЕЗАР

Керівник: Микола ПЕЧУРІН

Нормоконтролер: Василь МАЛЯРЧУК

Засвідчую, що у кваліфікаційній роботі
немає запозичень із праць інших авторів
без відповідних посилань
Студент: _____Ангеліна ЦЕЗАР

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет комп’ютерних наук та технологій
Кафедра комп’ютерних систем та мереж
Напрям (спеціальність) 123 «Комп’ютерна інженерія»
(шифр, найменування)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____І.А. ЖУКОВ
«____» _____2023 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

Цезар Ангеліни Юріївни
(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема роботи: «Топологія локальної комп’ютерної мережі стандарту *IEEE* 802.5 для наземного сегменту системи безпілотних авіаційних апаратів» затверджена наказом ректора від «29» серпня 2023р. №1521/ст;

2. Термін виконання роботи: з 02.10.2023 до 31.12.2023;

3. Вихідні дані до роботи: клас комп’ютерної мережі – локальна, сегмент БАК - наземний, стандарт – *IEEE* 802.5, метод вибору – бінарні порівняння;

4. Зміст пояснювальної записки (перелік питань, що підлягають розробці): Вступ; Перелік умовних позначень, скорочень, термінів; Розділ 1 Системний аналіз локальних комп’ютерних мереж стандарту *IEEE* 802.5; Розділ 2 Формалізація задачі проєктування топологій ЛКМ стандарту *IEEE* 802.5; Розділ 3 Вибір і розробка інструментарію дослідження розробленої моделі проєктування топології ЛКМ стандарту *IEEE* 802.5; Розділ 4 Дослідження розробленої моделі на умовному прикладі на предмет визначення ефективної топології ЛКМ для наземного сегменту системи безпілотних авіаційних апаратів; Висновки; Список бібліографічних посилань використаних джерел;

5. Перелік обов'язкового графічного матеріалу: презентація.

6. Календарний план-графік

№ з/п.	Завдання	Термін виконання	Підпис керівника
1	Узгодження технічного завдання	02.10.2023	
2	Підбір та опрацювання теоретичного матеріалу	02.10.2023 -12.10.2023	
3	Системний аналіз архітектурних рішень ЛКМ стандарту *IEEE* 802.5	12.10.2023 -19.10.2023	
4	Визначення критеріїв ефективності архітектурних рішень ЛКМ	19.10.2023 - 24.10.2023	
5	Розробка набору моделей задачі проєктування топологій ЛКМ стандарту *IEEE* 802.5	24.10.2023 - 27.10.2023	
6	Вибір базової моделі для дослідження	27.10.2023 - 03.11.2023	
7	Вибір інструментарію дослідження розробленої моделі	03.11.2023 -10.11.2023	
8	Розробка налаштування та структурної схеми генетичного алгоритму	10.11.2023 - 20.11.2023	
9	Розробка комп'ютерної програми на мові *Python*	20.11.2023 -10.12.2023	
10	Розрахунок топології ЛКМ стандарту *IEEE* 802.5 на умовному прикладі	10.12.2023 - 14.12.2023	
11	Оформлення пояснювальної записки та графічного матеріалу	14.12.2023 - 21.12.2023	
12	Подання матеріалів роботи на кафедру	22.12.23 – 31.12.2023	

7. Дата видачі завдання: «02» жовтня 2023 р.

Керівник кваліфікаційної роботи: _____ Микола ПЕЧУРІН

(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання: _____Ангеліна ЦЕЗАР

(підпис студента) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи «Топологія локальної комп'ютерної мережі стандарту *IEEE* 802.5 для наземного сегменту системи безпілотних авіаційних апаратів» 95 ст., 57 рис., 25 табл., 33 літературних джерел, 1 додаток.

TOKEN RING, IEEE 802.5, ЛОКАЛЬНА КОМП'ЮТЕРНА МЕРЕЖА, ГЕНЕТИЧНИЙ АЛГОРИТМ, СХРЕЩУВАННЯ, ВІДБІР, СЕЛЕКЦІЯ, МУТАЦІЯ

Мета кваліфікаційної роботи: дослідити способи розрахунку топологій ЛКМ стандарту *IEEE* 802.5, організованих для реалізації функцій наземного сегменту БАК.

Об'єкт дослідження: локальна комп'ютерна мережа стандарту *IEEE* 802.5.

Предмет дослідження: топологія локальної комп'ютерної мережі стандарту *IEEE* 802.5.

Результати дослідження: розроблена програма не лише виявляє найбільш оптимальні рішення для побудови топології, а й значно зменшує час, необхідний для пошуку цього рішення. Головна її ціль – зменшити фінансові витрати на кабельну інфраструктуру, що включає в себе витрати на сам кабель та його розведення.

Наукова новизна отриманих результатів полягає у тому, що для вирішення нової проблеми, а саме створення оптимальної топології локальної комп'ютерної мережі згідно до стандарту *IEEE* 802.5 для наземного сегменту системи безпілотних авіаційних апаратів, було запропоновано концепцію зведення задачі мінімізації фінансових витрат на кабель до відомої задачі комівояжера.

Програмні засоби, задіяні в спроєктованому об'єкті: *Python*, бібліотеки *numpy*, *math*, *geopy*, та *random*.

Прогнозні припущення про подальший розвиток: в комп'ютерну програму можна додати інтерактивний інтерфейс, що дозволить користувачу комбінувати різноманітні налаштування генетичного алгоритму з детальною інструкцією до кожного з них з метою покращення отриманих результатів, вбудувати в нього інтерактивні карти та графік з відображенням найкращих кандидатів для вирішення поставленої задачі в кожному поколінні.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

ACB - *Access Control Byte*

AHP - *Analytic Hierarchy Process*

DA - *Destination Address*

ED - *End Delimiter*

FCB - *Frame-control bytes*

FCS - *Frame-check sequence*

FS - *Frame Status*

IBM - *International Business Machines*

IEEE - *Institute of Electrical and Electronics Engineers*

LAN - *Local Area Network*

MAC - *Media Access Control*

MAU - *Multi-station Access Unit*

NAUM - *Nearest Active Upstream Neighbour*

SA - *Source Address*

SD - *Start Delimiter*

STP - *Shielded Twisted Pair*

THT - *Token Holding Time*

UTP - *Unshielded Twisted Pair*

ВСТУП

Безпілотний літальний апарат (БПЛА) в військовій сфері – це тип літального апарату без пілота на борту, який може керуватися як автономно, так і дистанційно, має в своєму складі цілевказівники, датчики, електронні та наступальні передавачі, з використанням яких можна перешкоджати роботі чи навіть знищувати ворожі цілі.

Автономне керування здійснюється з використанням бортових комп'ютерів, в які попередньо завантажують запрограмовані інструкції для виконання польоту або цілої місії. Дистанційне керування, в свою чергу, виконується за допомогою пілота на будь-які дистанції, основною вимогою є лише належний канал зв'язку з БПЛА. Водночас з цим такий вид керування може здійснюватися як на землі, так і з підводної або ж повітряної платформи.

БПЛА можна класифікувати по радіусу дії:

- БПЛА дуже близького радіусу можуть подорожувати на відстань до 5 км;

- БПЛА близького радіусу – до 48 км;

- БПЛА короткого радіусу – до 145 км. Цей тип БПЛА зазвичай використовують для збору розвідувальної інформації або ж шпигунства;

- БПЛА середнього радіусу – до 644 км. Можуть бути використані як для наукових досліджень, так і для метеорологічних спостережень.

- БПЛА дального радіусу можуть виходити за межі 644 км та підніматися на висоту до 914 метрів.

Важливість безпілотних авіаційних апаратів зростає з кожним роком, оскільки вони інтегруються в усі сфери життя. В основному вони використовуються в таких областях:

- військові операції:

- проведення розвідки;

- виконання бойових операцій, особливо для завдання ударів по морських і наземних цілях;

- доставка гуманітарних вантажів, боєприпасів і медикаментів у важкодоступні та віддалені райони);

- переміщення озроєння;

- евакуація поранених солдат з лінії фронту.

- сільськогосподарський сектор:

- вимір обсягів врожаю;

- обробка полів;

- створення електронних карт земельних ділянок;

- моніторинг стану ґрунту тощо.

- розважальна індустрія (створення фото та відеоконтенту з повітряних ракурсів на різноманітні заходи, такі як фестивалі, концерти та інше)

- дослідницька робота (для збору даних про клімат, геологічні зміни, природні явища тощо).

Окрім цього дрони використовуються:

- для збереження дикої природи, оскільки дозволяють наглядати за бродячими групами твар та відстежувати стан їх здоров'я. Також їх використовують для боротьби з браконьєрством в Африці та Азії;

- для відновлення лісів: дрони очищають підстилку знищених під час пожежі лісів, скидають з повітря добрива, насіння та поживні речовини;

- для доставки медичних засобів та навіть донорських органів;

- для пошуку людей після природних катаклізмів, таких як землетруси, урагани, снігові лавини тощо.

Причини використання безплілотних авіаційних апаратів не завершується на різноманітті сфер їх використання.

До додаткових переваг можна віднести:

- зменшення фінансових витрат на навчання та утримання пілотів, оскільки у них немає необхідності;

- мінімальні затрати на створення та експлуатацію;

- можливість працювати в небезпечних районах без загрози людському життю;

- виконання маневрів та задач з високою точністю у місцях, де складно або ж неможливо їх виконати при використанні звичайних літальних апаратів;

- зменшення вразливості до атак в порівнянні з традиційними ЛА.

Термін «безпілотний авіаційний комплекс» підкреслює важливість, не тільки літального апарату, а й інших його елементів. Безпілотний авіаційний комплекс складається з наземного та повітряного сегментів. Роль повітряного сегменту полягає в польоті по певному маршруту, отриманому від наземного сегмента, який в свою чергу складається з таких компонентів як обладнання запуску та обслуговування, системи управління, обладнання зв’язку та самого персоналу.

Наземні станції управління можуть бути як столом з декількома екранами, так і портативним контролером або ж взагалі програмним додатком. З використанням цієї станції здійснюється керування польотом, зчитування даних про стан, контроль датчиків корисного навантаження, планування місії і т.д. Для зв’язку між наземним та повітряним сегментом зазвичай використовують радіочастотну технологію, яка дозволяє надати оператору такі дані як відстань до оператора, висоту польоту, час польоту, відстань до цілі та тому подібне.

Актуальність теми

Військові умови, які наразі тривають у нашій країні, вимагають особливої уваги до економії засобів при проєктуванні топології ЛКМ для наземного сегменту системи безпілотних авіаційних апаратів. Ця проблематика потребує надзвичайної уваги, оскільки фінансові ресурси є обмеженими, а критичні потреби є нагальними. В результаті заощаджені кошти можна використати на інші стратегічно важливі потреби.

Крім того, збільшення захищеності та надійності зв'язку між наземним та повітряними сегментами допоможе у проведенні успішних та оперативних дій. Це збільшить шанси на успішне виконання військових завдань та прийняття вдалих стратегічних рішень. Покращений зв'язок між сегментами дозволить більш оперативно та ефективно реагувати на зміни у військовій обстановці, що в свою чергу підвищить шанси на реалізацію стратегічних цілей та посприяє швидшому просуванню до досягнення поставлених завдань у цілому.

Мета кваліфікаційної роботи: дослідження методів розрахунку топології ЛКМ стандарту *IEEE* 802.5, організованих для реалізації функції наземного сегменту безпілотних авіаційних комплексів.

Завдання кваліфікаційної роботи:

- розглянути ключові особливості та характериситики ЛКМ стандарту *IEEE* 802.5 шляхом проведення системного аналізу;

- визначити основні критерії ефективності архітектурних рішень ЛКМ стандарту *IEEE* 802.5;

- розробити набір моделей задачі проектування топологій ЛКМ стандарту *IEEE* 802.5, та серед яких обрати базову модель для дослідження з використанням методу аналізу ієрархій;

- визначити набір інструментів для виконання етапу пошуку способів покращення вибраного об'єкту комп'ютерної інженерії, а саме метод повного перебору, метод гілок та меж, метод Літтла, алгоритм Хелда-Карпа, метод найближчого сусіда, метод найбільш дешевого включення, дерев'яний алгоритм, мурашиний алгоритм, генетичний алгоритм та алгоритм імітації відпалу;

- обрати найбільш ефективний інструмент серед представленого набору на основі таких параметрів, як розмір набору вхідних даних, час на виконання, точність результату та доступна продуктивність;

- розробити налаштування генетичного алгоритму для дослідження розробленої моделі з попереднім дослідженням основних методів відбору, схрещування, мутації та селекції;

- розробити комп'ютерну програму на мові *Python* для знаходження найбільш ефективного шляху з'єднання елементів комп'ютерної мережі для наземного сегменту системи безпілотних авіаційних апаратів;

- провести тестування розробленої програми шляхом розрахунку топології ЛКМ стандарту *IEEE* 802.5 на умовному прикладі;

Об'єкт дослідження: локальна комп'ютерна мережа стандарту *IEEE* 802.5.

Предмет дослідження: топологія локальної комп'ютерної мережі стандарту *IEEE* 802.5.

Методи дослідження: аналіз, індукція, порівняння, класифікація, абстрагування.

Наукова новизна отриманих результатів полягає у тому, що для вирішення нової проблеми, а саме створення оптимальної топології локальної комп'ютерної мережі згідно до стандарту *IEEE* 802.5 для наземного сегменту системи безпілотних авіаційних апаратів, було запропоновано концепцію зведення задачі мінімізації фінансових витрат на кабель до відомої задачі комівояжера. Цей підхід передбачає використання теорії графів та математичних моделей для знаходження оптимальних маршрутів, що сприятиме ефективному розташуванню елементів мережі з мінімальними фінансовими витратами, а також полегшить планування кабельної інфраструктури системи безпілотних авіаційних апаратів.

Практичне значення отриманих результатів: розробка комп'ютерної програми відображає ключовий крок в оптимізації процесу проєктування локальних комп'ютерних мереж за стандартом *IEEE* 802.5 для наземного сегменту системи безпілотних авіаційних апаратів. Створена програма не лише виявляє найбільш ефективні рішення побудови топології, а й зводить часові витрати, які необхідні для її проєктування, до мінімуму. Її основна ціль полягає в мінімізації фінансових витрат витрат на кабельну інфраструктуру, які включають в себе фінансові витрати на сам кабель та його розведення.

РОЗДІЛ 1
СИСТЕМНИЙ АНАЛІЗ ЛОКАЛЬНИХ КОМП'ЮТЕРНИХ МЕРЕЖ СТАНДАРТУ *IEEE* 802.5

1.1. Суть системного аналізу із визначенням особливостей його застосування для об'єкту аналізу – архітектурних рішень ЛКМ стандарту *IEEE* 802.5

Мережа *Token Ring* – основна технологія локальної комп'ютерної мережі (*LAN*), яка була розроблена *IBM* в 1970-х роках. Специфікація *IEEE* 802.5 була створена за моделлю *IBM Token Ring* та є повністю сумісною з нею [1].

Різниця між ними полягає в тому, що в специфікації *IEEE* 802.5 не визначено топологію та тип середовища передачі. В той час як для технології *Token Ring* передбачено використання топології «зірка» та витої пари.

Також існує різниця між кількістю станцій в сегменті: в *Token Ring* все залежить від наявності екрану на витій парі: при використанні екранованого кабелю – 260 станцій, в іншому випадку – 72. В *IEEE* 802.5 визначено використання 250 станцій. Як *IEEE 802*.5, так і *IBM Token Ring* підтримують однакові швидкості передачі даних, а саме 4 або 16 Мбіт/с [2].

До того ж існує модернізований варіант з підтримкою 100 та 155 Мбіт/с зі збереженням особливостей технології 16 Мбіт/с. Передача сигналів виконується в основній смузі часот [2].

Мережі *Token Ring* побудовані з використанням топології «кільце», яка логічно реалізована всередині багатостанційного блоку доступу (англ. *Multi-station Access Unit (MAU)*), який є еквівалентом концентраторів *Ethernet* з деякими відмінностями. Зокрема, вхідні дані відправляються не на всі порти, а лише на наступний [3].

На рис. 1.1 представлено *IBM* 8228 *Multistation Access Unit*.

Рис. 1.1. *IBM* 8228 *Multistation Access Unit*

Всі станції підключаються до багатостанційного блоку доступу по топології «зірка», в свою чергу *MAU* об'єднуються між собою та утворюють магістральне фізичне кільце. Для об'єднання *MAU* з іншими сусідніми *MAU* використовуються порти *Ring In* та *Ring Out* (рис. 1.2).

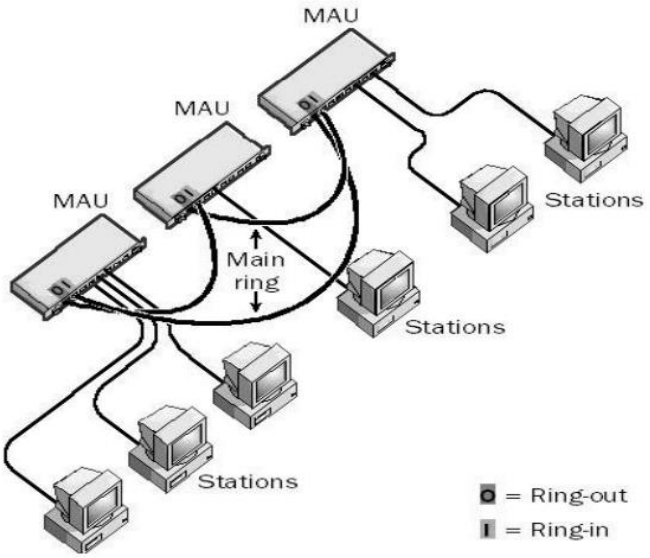


Рис. 1.2. Підключення *MAU* один до одного з використанням портів *Ring In* та *Ring Out*

Існує дві найбільш використовувані специфікації фізичного рівня під назвою *Type* 1 та *Type* 3 [3].

Type 1 – екранована вита пара (*STP*). Роз'єм на кінці багатостанційного блоку доступу називають *IBM Data Connector*. Для побудови мережі необхідно використати два види кабелю: пелюсткові (*IDC – DB*-9) (рис. 1.3) та патч-кабелі (*IDC-IDC*). Перші використовуються для з'єднання комп'ютерів та *MAU*, а другі – для підключення *MAU* між собою.

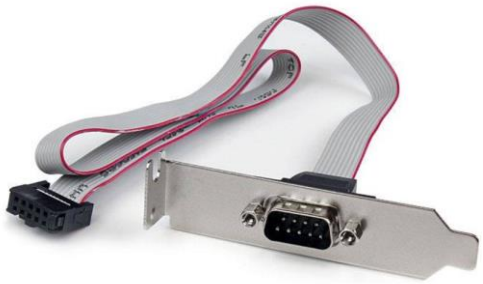


Рис. 1.3. Кабель *IDC* – *DB*-9

Type 3 – *UTP* категорії 5 з роз'ємами *RJ*-45 (рис. 1.4). Тобто, як для комп'ютерів, так і для *MAU* використовуються роз'єми *RJ*-45.

Рис. 1.4. *UTP* категорії 5 з роз'ємами *RJ*-45

Різниця між двома специфікаціями полягає в довжині кабелю та кількості підключених робочих станцій. Довжина *Type* 1 обмежена 100 метрами, а *Type* 3 – 45. Такі ж обмеження довжини діють і на з'єднання пасивних *MAU*, а при використання активних – довжина збільшується до 730 та 365 м. Мережі, які використовуються *Type* 1, можуть задіяти до 260 робочих станцій, а *Type* 3 – лише до 72. Максимальна довжина кільця складає 4 км.

Задані обмеження на довжину та кількість станцій зумовлені часом обороту токену по кільцю. Наприклад, при використання 260 станцій, маркер повернеться до активного монітора через 2.6 с згідно до *Token Holding Time* (*THT*), рівному 10 мс. Отримане значення складає інтервал контролю обороту токена. Обмеження досить гнучкі, оскільки при налаштуванні значення тайм-аутів існує можливість збільшення як кількості кінцевих станцій, так і довжини кільця [4].

Також можна використати волоконно-оптичний кабель для з'єднання *MAU*, що дозволить розширити ЛОМ на більшу відстань [3].

Для доступу до загального розділеного ресурсу, яким є кільце, застосовують детермінований алгоритм. По мережі виконується передача маркеру, володіння яким дає право на передачу, яке передається циклічно між станціями по логічному кільцю. Будь-яка станція отримує дані від попередньої станції, яка ще називається *Nearest Active Upstream Neighbour* (*NAUM*) та передає її до вузла, який знаходиться нижче по потоку [3].

Принцип маркерного доступу представлений на рис. 1.5.

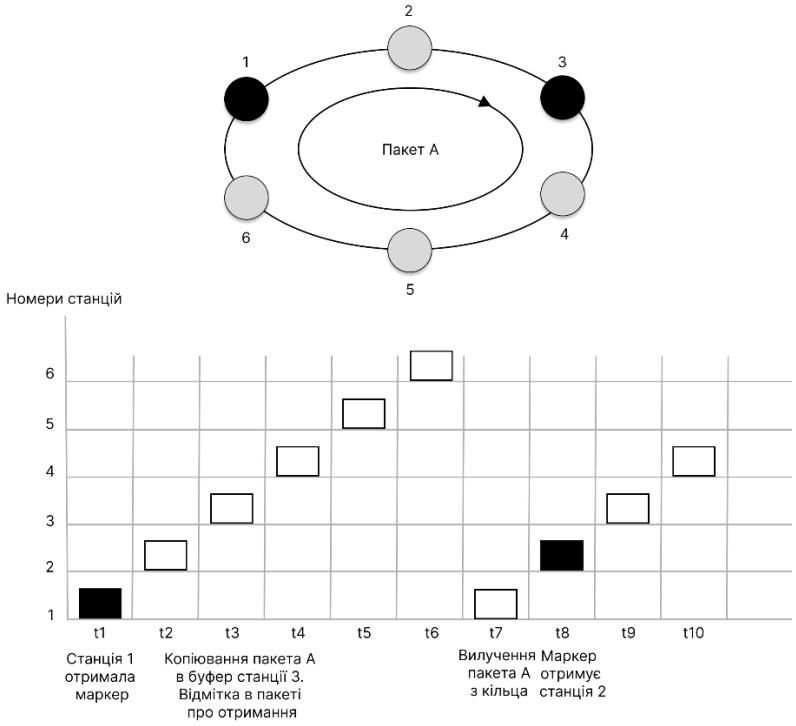


Рис. 1.5. Принцип маркерного доступу

Маркер – повідомленням довжиною 24 біта, який служить для надання права на відправку даних вузлу. Кожна кінцева станція, отримавши токен, може утримувати його максимальний період часу і, якщо вона не має даних до відправки, він передається до наступного вузла.

Тобто кожен вузол може або відправляти свої дані, або ретранслювати дані, які він отримав від *NAUM*, але обидві операції виконувати він не в змозі. *Token Holding Time* (*THT*) за замовчанням складає 10 мс, а максимальний розмір фрейма для мереж 16 Мбіт/с рівний 16 Кбайт, відповідно в мережах 4 Мбіт/с – 4 Кбайт. За час утримання

маркера станції необхідно встигнути передати хоча б один кадр. При використанні швидкості 4Мбіт/с можна передати 5000 байт, а при 16Мбіт/с – 200000 байт [4].

У випадку, коли інформація все таки існує, станція захоплює маркер та змінює в ньому 1 біт, що зумовлює його перетворення на послідовність «початок кадру». Після чого додається інформація та здіснюється її відправка на наступні станції по кільцю. Коли фрейм досягає станції-отримувача, він видаляється. Оскільки поки виконується передача даних, в мережі нема маркера, кінцеві мережі повинні очікувати її завершення для того, щоб здійснити нову передачу. При підтримці раннього звільнення маркера, він стає доступним після завершення передачі кадру [4].

Для надання деяким станціям можливості використовувати мережу частіше, в *Token Ring* передбачено використанння системи пріорітетів. Для контролю пріоритету в кадрах використовується два поля: поле пріорітету та поле резервування. Кінцеві станції, які мають пріоритет рівний або вище заявленого в маркері, можуть його заняти. Резервування маркера виконується лише після перетворення його в інформаційний кадр та для кінцевих станцій, які мають пріорітет вище, ніж у передаваючої станції. Коли відбувається генерація нового маркера, він включає в себе більш високий пріорітет станції, що його резервує. Після завершення передачі станції, які підвищували пріорітет маркера, повинні відновити свій попередній пріоритет [2].

Для виявлення та компенсації помилок в *Token Ring* використовуються декілька механізмів. Наприклад, будь-яка одна станція в мережі може бути обрана у якості активного монітору на основі найбільшого значення *MAC*-адреси. Він слугує для надання часової інформації та обслуговування кільця. Однією з функцій є видалення постійно циркулюючих кадрів з кільця при виході з ладу передавального вузла та генерація нового маркеру [5].

У додатку до цього існує алгоритм *beaconing*, який дозволяє виявити та усунути проблеми в мережі. У випадку обриву кабелю, станція, яка виявила цю проблему, посилає кадр маяка. Цей кадр визначає область відмови, в яку входить станція, яка виявила збій, найближча активна станція, та всі станції, що знаходяться між ними. *Beaconing* запускає автоконфігурацію, в процесі виконання якої вузли в цій області

проводять автоматичну діагностику для переналаштування мережі навколо ділянок відмови [2].

В *Token Ring* та *IEEE* 802.5 підтримується два типи кадрів: токени та кадри даних/команд. Маркеми мають довжину 3 байта, а довжина кадрів даних-команд залежить від розміру інформаційного поля. Токени складаються з початкового роздільника (*Start Delimiter*), байта контролю доступу (*Access-control bite*) та кінцевого роздільника (*End Delimiter*) [7].

Start delimiter (*SD*), поле довжиною 1 байт, яке використовується для того, щоб повідомити кожну станцію про надходження маркера або ж кадру даних/команд [7]. Воно є закодованою манчестерським кодом послідовністю символів: *JKOJKOOO*.

Access-control bite (*ACB*) (1 байт) складається з поля пріорітету та поля резервування (рис. 1.6) [7]. Обидва поля займають по три біти. Для того, щоб відрізнити маркер від кадру даних/команд, використовується біт маркера. Якщо він встановлений на 1, то це маркер доступу. Біт моніторингу призначений для виявлення нескінченної передачі кадру по кільцю. Активний монітор встановлює для цього біта значення 1, інші станції – 0. В тому випадку, коли кадр або маркер обійшов кільце та все ще має 1 в цьому біті, активний монітор на це реагує. У випадку, коли це був кадр, він видаляється. А якщо це був маркер, він повторно відправляється по кільцю.

Рис. 1.6. Формат поля *Access Control*

End delimiter (*ED*) (1 байт) використовується для сповіщення про кінець маркеру або кадру даних/команд [7]. Це поле також містить біти для визначення пошкодженого фрейму та останнього фрейму у логічній послідовності. Признак помилки *E* встановлюється 1, в тому випадку, якщо одна зі станцій виявить некоректність у кадрі або помилку в контрольній сумі. Признак *I* вказує, чи є кадр проміжним або останнім в серії фреймів.

Кадри даних містять інформацію для протоколів верхнього рівня, тоді як кадри команд містять керуючі дані.

Вони складаються з тих же полів, що і кадр маркеру, але з доповненням декількох інших, таких як: контрольні байти кадру, адреса призначення та джерела, дані, послідовність перевірки кадру, статус кадру.

Кадр маркеру та кадр даних/команд представлено на рис. 1.7.

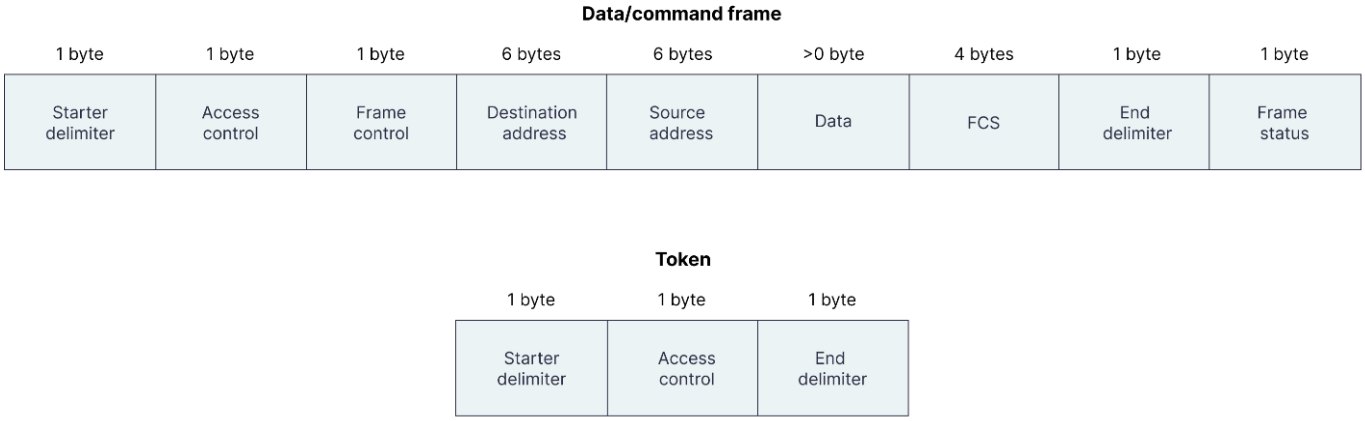


Рис. 1.7. Формати кадру маркеру та кадру даних/команд

Розглянемо їх детальніше:

Frame-control bytes (*FCB*) (1 байт) – призначений для визначення вмісту кадру, тобто чи містить кадр керуючу інформацію чи дані. Цей байт слугує для визначення типу керуючої інформації у випадку використання керуючого кадру.

Destination and source addresses – для ідентифікації відповідних адрес, містить два поля довжиною 2 або 6 байтів. Розглянемо призначення перших двох бітів адреси призначення. Перший призначений для визначення індивідуальної або групової адреси, другий – яким чином призначено адресу: глобально чи локально. При наявності 1 в першому біті адреси джерела повідомляється про наявність *Routing Information Field* (*RIF*). Маршрутна інформація використовується у випадку роботи мостів в режимі маршрутизації від джерела.

Дані – призначений для того, щоб вказати, що довжина поля залежить від часу утримання маркеру. Максимальна довжина цього поля не визначена, але існує обмереження на базі часового відношення між *THT* та часом передачі фрейма.

Frame-check sequence (*FCS*) (4 байти) – для перевірки наявності пошкодження кадру під час передачі. Це значення обчислене станцією-джерелом та залежить від

вмісту кадру. Після отримання кадру станцією призначення виконується повторне обчислення цього значення.

Frame Status (*FS*) (1 байт) – поле, яке завершує кадр команд/даних та містить індикатори копіювання кадру та розпізнавання адреси [7].

Існують наступні типи кадрів команд *MAC*-рівня:

- *Duplicate Address Test* (*DAT*) використовується для перевірки унікальності адреси станції;

- *Active Monitor Present* (*AMP*) - для сповіщення про роботоспроможність активного монітору іншим станціям. Цей кадр відправляється активним монітором кожні 3 секунди, при його відступності протягом більше 7 секунд відбувається процедура пошуку заміни;

- *Standby Monitor Present* (*SMP*) відправляється будь-якою станцією, окрім активного монітору;

- *Claim Token* (*CT*) – відправляється в тому випадку, коли є підозра на відмову активного монітора. Після його відправки резервні монітори домовляються про те, який з них його замінить;

- *Beacon* (*BCN*) – відправляється у випадку мережевих проблем, наприклад, відмови кінцевої станції або обриву кабелю;

- *Purge* (*PRG*) – для очищення кільця від невилучених кадрів та переведення станцій в початковий стан [8].

1.2. Критерії ефективності архітектурних рішень локальних комп'ютерних мереж стандарту *IEEE* 802.5 для умов використання в системах управління безпілотними апаратними комплексами

Для оцінки ефективності архітектурних рішень ЛКМ можна скористатися такими критеріями, як продуктивність, вартість, масштабованість, надійність, безпека та інші.

Продуктивність є одним з основних критеріїв оцінки ефективності комп’ютерної мережі. Продуктивність вимірюється з використанням наступних показників: часу відгуку, пропускної спроможності та затримки [8].

Час реакції визначає часовий інтервал між створенням запиту і отриманням відповіді на нього. На його значення впливають наступні фактори: тип служби, поточний стан компонентів мережі та інше [8].

Пропускна здатність відповідає за визначення об’єму даних, які можна передати по каналу зв’язку за певний часовий проміжок. Вимірюється в бітах за секунду або пакетах за секунду. Існують наступні види пропускної здатності: максимальна, середня та миттєва. Для їх обчислення необхідно поділити загальний об’єм даних на час їх передачі [8].

Різниця між середньою та миттєвою пропускною здатністю полягає лише в виборі інтервалу часу: при обчисленні першої обираються найбільш довгі проміжки часу, наприклад, день чи тиждень, у той час як при обчисленні другої – найменший проміжок, наприклад, 1 секунда.

Максимальна пропускна здатність – найбільше значення миттєвої пропускної здатності, отримане за відповідний проміжок часу. Значення цього показника показує наскільки мережа здатна справлятися з максимальними навантаженнями, спровокованими використанням ресурсів мережі великою кількістю користувачів в особливі періоди роботи [8].

Затримка передачі схожа на час реакції, тобто визначає час, необхідний для передачі пакету від відправника до отримувача з єдиною відмінністю, яка полягає в тому, що цей показник не бере до уваги затримки обробки комп’ютерами. Для оцінки якості мережі використовується максимальна затримки передачі та джиттер. Зазвичай затримка передачі не перевищує сотень мс, в деяких випадках – декількох секунд. Чутливість до цих затримок залежить від типу трафіку в мережі. Наприклад, при використанні служб електронної пошти або ж друку, дані затримки не відчуваються користувачами. Такі ж затримки при передачі відео або голосових даних можуть призводити до зниження якості цієї інформації, наприклад, нерозбірливість у розмові, відсутність синхронізації між аудіо та відео [8].

Останні два параметри є незалежними, тобто мережа одночасно може мати як велику пропускну здатність, так і великі затримки передачі. Продуктивність мережі залежить від ряду аспектів: кількості активних користувачів, типу середовища передачі даних, відстані тощо.

Надійність визначає здатність комп'ютерної мережі передавати дані, уникаючи помилок та втрат. Для її вимірювання можна скористатися такими метриками, як середній наробіток між відмовами, частота втрати пакетів, частота помилок та час безвідмовної роботи. Ці показники доцільно використовувати для простих елементів, які описуються двома станами: неробоспроможним та роботоспроможним. Для більш складних систем, в арсеналі яких більше двох станів, необхідно використовувати інший набір показників, наприклад, коефіцієнт готовності, ймовірність доставки пакету без спотворень, ймовірність втрати пакету, відношення втрачених пакетів до доставлених та інші [11].

Безпека – здатність мережі до захисту даних від несанкціонованого доступу, перехоплення та модифікації. На безпеку можуть впливати наступні загрози: шкідливе програмне забезпечення, атаки на відмову, перехват мережевого трафіку, соціальна інженерія та інше. Включає в себе шифрування, авторизацію, автентифікацію та контроль доступу [8].

Відмовостійкість – поняття, яке відображає здатність системи обмежити вплив відмови чи збою певних компонентів на користувачів. Таким чином відмова не зумовлює повну зупинку системи, а лише знижує якість роботи. Вона допомогає звести до мінімуму простої, а також зберегти цілісність та доступність даних [8].

Розширюваність представляє собою здатність до відносно легкого внесення змін та доповнень до системи. Наприклад, нарощування довжини сегментів, кількості комп'ютерів, користувачів, служб або додатків, оновлення мережевого обладнання та ін. Таким чином, система може адаптуватися до нових вимог, додаючи або модифікуючи компоненти з мінімальними витратами ресурсів та часу [8].

Масштабованість – здатність мережі зберігати якість роботи і продуктивність при зростаючих вимогах, таким як збільшення кількості користувачів та обсягу даних, або ж при зміні показників завдань.

Існує два види масштабування: вертикальне та горизонтальне. Вертикальне – передбачає нарощування потужності наявного вузла шляхом покращення апаратної частини. Горизонтальне – заключається в додаванні нових вузлів чи серверів до системи.

Додаткова відмінність між ними заключається в тому, що вертикальне масштабування легше реалізувати, оскільки немає необхідності розподілу навантаження та синхронізації даних між вузлами. Але вертикальне масштабування має обмеження в розширенні, тобто існує гранична межа в тому, наскільки потужне обладнання можна встановити на вузол, тоді як горизонтальне масштабування не має жорстких обмежень в розширенні, все залежить від фінансових можливостей [10].

Прозорість означає, що мережа спроектована так, щоб спрости взаємодію користувача з нею шляхом приховування складності та деталей свого внутрішнього функціонування. Таким чином, робота в мережі стає зручнішою та інтуїтивно зрозумілою, що дозволяє користувачам не вдаватися в технічнічні деталі та складності мережевої інфраструктури, а фокусуватися лише на своїх завданнях [9].

Керованість визначає можливість централізованого спостереження, управління та контролю за кожним елементом мережі. Ефективна система управління повинна мати зручний інтерфейс та вести непреривний моніторинг. У випадку, коли виникають помилки, вона повинна їх виявити, усунути та повідомити про це системного адміністратора, представивши йому дані про помилку та шляхи її вирішення. Також вона повинна реалізовувати накопичення цих даних, яке буде допомагати у плануванні розвитку мережі. Системи управління приносять найбільшу користь для великих мереж, таких як, наприклад, корпоративні, оскільки замінюють велику кількість спеціалістів, яких необідно було б розмістити в кожному приміщенні з мережевим обладнанням [9].

Сумісність або інтегрованість означає здатність мережі приймати та об’єднувати в єдине ціле різноманітне апаратне та програмне забезпення. Таким чином не виникає проблем з використанням різним операційних систем або ж додатків та пристроїв від різних виробників. Гетерогенною мережею називається

мережа, яка містить в собі різнотипні елементи. Інтегрованою називається гетерогенна мережа, яка функціонує без помилок [9].

Вартість – фінансові витрати, необхідні при проєктуванні, розгортанні та експлуатації комп’ютерної мережі [11].

До цих витрат можна віднести:

- вартість апаратного обладнання: комп’ютерів, маршрутизаторів, серверів, комутататорів та іншого;

- вартість програмного забезпечення, до якого входять операційні системи, антивірусне ПЗ, програми управління мережею, додатки, які використовуються в мережі;

- вартість обслуговування та підтримки – витрати, пов'язані з забезпеченням нормальної функціональності і безпеки комп’ютерної мережі, а також наданням технічної підтримки користувачам і вирішенням різних питань, які виникають під час її експлуатації;

- вартість простою, порушень безпеки тощо.

Висновки за розділом

В першому розділі були розглянуті основні аспекти специфікації *IEEE* 802.5 та технології *Token Ring*, зокрема: особливості специфікації *IEEE 802.5* та *Token Ring*: використовувані топології, кількість станцій в сегменті, шифрування, швидкость передачі та середовища передачі; принцип роботи мережі *Token Ring*; механізми виявлення та компенсації помилок; типи кадрів: кадр маркеру та кадр даних/команд.

Також були визначені критерії для оцінки ефективності архітектурних рішень ЛКМ, зокрема: продуктивність, надійність, безпека, відмовостійкість, розширюваність, масштабованість, керованість, прозорість, сумісність/інтегрованість та вартість.

РОЗДІЛ 2
ФОРМАЛІЗАЦІЯ ЗАДАЧІ ПРОЄКТУВАННЯ ТОПОЛОГІЙ ЛКМ СТАНДАРТУ *IEEE* 802.5

2.1. Розроблення набору моделей задачі проєктування топологій ЛКМ стандарту *IEEE* 802.5

Для вирішення задачі звернемося до теорії графів та представимо комп'ютерну мережу у вигляді графу. Вершини графу – компоненти ЛКМ наземного сегменту БАК, а дуги – канали зв'язку між ними.

Перед розробленням набору моделей визначимося з критеріями:

- кожна пара вершин повинна бути зв'язана одна з одною, тобто ступінь зв'язності повинен бути рівний двум. Це обмеження накладається з урахування використання саме технології *IEEE* 802.5 для побудови топології ЛКМ;

- зробимо припущення про те, що збільшення сумарної довжини кабелю призводить до збільшення випромінювання, що призводить до зменшення захищенності мережі. Таким чином, цільова функція набуває наступного вигляду

$$f = \sum l_{ij} -> min$$

Виходячи з цих двох критеріїв, перед нами постає задача знаходження маршруту з мінімальною довжиною, який проходить через низку компонентів та повертається в початкову точку, яка має назву «задача комівояжера».

Різниця між орієнтованим та неорієнтованим графами полягає в тому, що перший має направлені (рух по ребру лише в один бік) ребра, в той час як другий – ні [12]. Змішаний граф має як направлені, так і ненаправлені ребра. Оскільки перед нами постає задача представлення топології мережі, можна сказати про те, що напрямок для нас не є важливим фактором, тому доцільно використати саме неорієнтований граф. Таким чином одразу відкидаємо ідею використання орієнтованих та змішаних графів.

Виходячи з першого критерію, граф повинен бути зв’язним, оскільки між кожною парою вершин повинна бути одна дуга.

Так як головною метою є створення топології із найменшою загальною довжиною каналів зв'язку, оберемо саме зважений граф, оскільки кожному ребру буде призначено відстань між вершинами, на основі значень яких буде обрано оптимальний маршрут.

Використання повного графу, тобто графу, в якому кожна вершина зв’язана з кожною іншою, відкидається з урахуванням другого критерію [13]. Додатково до цього, побудова даної мережі є фінансово витратною та складною як при початковій реалізації, так і при її масштабуванні. Повний граф представлено на рис. 2.1.

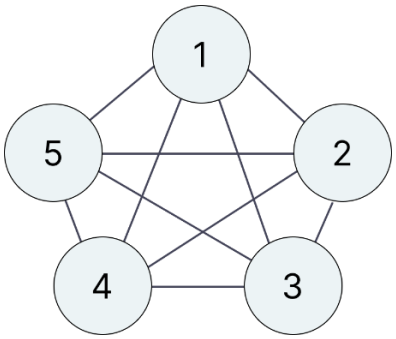


Рис. 2.1. Повний граф

Також відкинемо ідею використання дерева, оскільки цей граф не утворює замкнутого циклу, тобто не має можливості повернутися в початкову вершину, не пройшовши по одному ребру декілька разів [12]. Таким чином, порушується перший критерій. Дерево представлено на рис. 2.2.

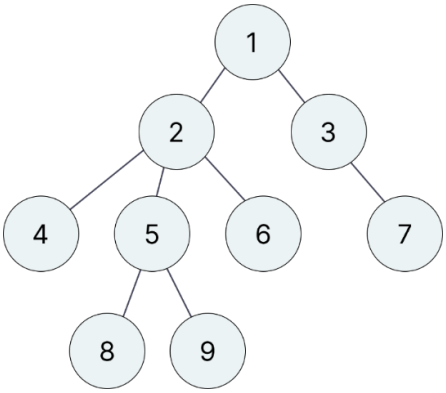


Рис. 2.2. Дерево

Ейлерів граф – граф, що містить ейлерів цикл, який в свою чергу проходить всі ребра по одному разу та повертається в початкову точку, а також містить в собі всі вершини, можливо навіть по декілька разів [15]. Ейлерів граф обов’язково має бути зв’язним, всі вершини повинні мати парну степінь зв’язності. Ейлерів граф представлено на рис. 2.3.

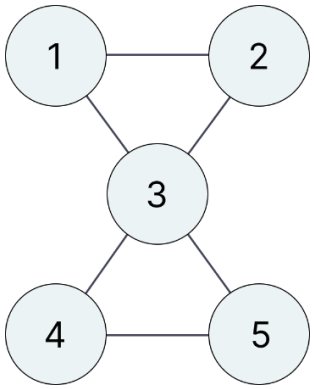


Рис. 2.3. Ейлерів граф

Гамільтонів граф - граф, що містить гамільтонів цикл, який в свою чергу проходить кожну з вершин по одному разу та повертається в початкову точку [16], утворюючи замкнутий цикл. Обов’язкова умова – граф повинен бути зв’язним. Гамільтонів граф представлено на рис. 2.4.

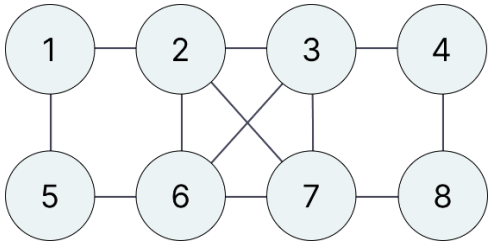


Рис. 2.4. Гамільтонів граф

2.2. Вибір базової моделі для дослідження

Щоб вибрати базову модель, яка підійде для нашого дослідження, скористаємося методом аналізу ієрархій (англ. *Analytic Hierarchy Process* (*AHP*)). Він був започаткований відомим американским вченим Томасом Сааті в 70-80 роках XX століття. Цей метод полягає у ієрархічному розбиттті проблеми на більш прості частини та подальшому аналізі результатів попарних порівнянь, проведених

експертами. Використовується для обумовленості прийняття рішення в умовах широкого набору критеріїв та визначеності.

Цей метод має кілька основних переваг. Перш за все, він пропонує простий процес обчислення та чітку інтерпретацію отриманих даних, що робить складні моделі більш зрозумілими. Крім того, він містить оцінки, засновані як на кількісних, так і на якісних факторах, що забезпечує його адаптивність до різних сценаріїв. Примітно, що *AHP* демонструє стійкість проти невідповідностей в оцінках, забезпечуючи надійні результати [17].

Однак важливо визнати головне обмеження — оптимальна ефективність методу спостерігається при обробці обмеженої кількості альтернатив і критеріїв, в ідеалі не перевищуючи 7-10 загалом.

Містить наступні етапи [17]:

- Визначення проблеми та цілі;

- Визначення критеріїв та альтернатив;

- Побудова ієрархічної структури: ціль – критерії – альтернативи;

Всі вузли та зв’язки утворюють систему, яка з математичної точки зору є графом. Вузлами є головна ціль, критерії та всі можливі альтернативи. Кожен рівень містить групу однотипних вузлів. Вузли одного рівня, які підпорядковуються вузлу іншого рівня, об’єднуються, утворюючи кластер. Найвищий рівень, тобто вершина ієрархії, містить головну ціль, тоді як нижній - різноманітний набір альтернатив. Кількість проміжних рівнів є довільною. Зв’язки зображуються стрілкою та вказують вплив одного вузла на інший. В ієрархії ціль впливає на критерії, а критерії – на альтернативи. Критерії порівнюють за важливістю щодо цілі, а альтернативи – за перевагою щодо кожного окремого критерію. Вплив поширюється від вищого до нижчого рівнів, уникаючи зворотніх та горизонтальних зв’язків.

- Побудова матриці попарного порівняння критеріїв та альтернатив по кожному з критеріїв;

Необхідна кількість матриць залежить від кількості рівнів ієрархічної структури та критеріїв, які розглядаються. Наприклад, при використанні простої трьохрівневої ієрархії кількість матриць рівна числу критеріїв плюс 1. Застосування

попарного порівняльного підходу до оцінювання, як правило, дає більш точні результати порівняно з проведенням оцінювання серед більшого набору альтернатив.

- Формування позитивної обернено-симетричної матриці $a(i,j)$, елементами якої є значення інтенсивності прояву елемента ієрархічної структури i відносно елемента j. Для заповнення матриці необхідно провести $n(n\text{-}1)/2$ суджень, водночас з цим мінімальне значення суджень рівне $(n\text{-}1)$.

Суттєвий недолік, а саме зростання трудомісткості, пов'язаний з процесом попарного порівняння, стає більш помітним зі збільшенням кількості оцінюваних елементів. Крім того, існує точка зору, яка стверджує, що близько 50% винесених суджень можуть бути несуттєвими, служачи більше для підтримки узгодженості матриці, а не безпосередньо сприяючи процесу прийняття рішень. У випадках, коли експерт стикається з труднощами у винесенні точних суджень, можливо заповнити пусті комірки шляхом використання правил початку або правил зупинки [18].

Перш ніж приступити до попарного порівняння, необхідно провести вибір шкали оцінювання та спосіб порівняння.

Наприклад, при використанні порівняння по Стівенсу виконується вибір одного з вузлів кластера у якості еталону, з яким порівнюються всі інші вузли кластеру. В той час, якщо обрати класичне порівняння, кожен окремий вузол порівнюється з рештою вузлів кластера.

Оберемо дев’ятибальну шкалу відношень, запропоновану Т. Сааті, де:

1 – відповідає за рівну важливість критеріїв;

3 – помірна перевага одного елементу над іншим, але міркування не переконливі;

5 – суттєва перевага з наявністю надійних даних або логічних суджень;

7 – очевидна перевага;

9 – абсолютна перевага.

Парні цифри використовуються в проміжних випадках, коли необхідні компромісні рішення [19].

При виконанні порівняння ставиться наступне питання: Який елемент важливіший/вірогідніший/кращий?

Коли елемент *i* вважається більш переважним за елемент *j*, в комірку (*i*,*j*) записується ціле значення, а в комірку (*j*,*i*) обернена цьому значенню величина. У ситуаціях, коли вони рівнозначно важливі, в обидві комірки записується одиниця. По діагоналі матриці розставляються одиниці, оскільки кожний критерій рівний собі ж по важливості. Оцінки елементів нижчого рівня залишаються незалежними від тих, що знаходяться на вищих рівнях [20].

Після заповнення матриці порівняння нам необхідно визначитися з двома термінами: узгодженість та транзитивність [20].

Матриця вважається узгодженою, якщо

$$a_{ij}a_{jk} = a_{ik}$$

Матриця вважається транзитивною, якщо при

$$\begin{cases} a_{ij} < a_{jk} \\ a_{jk} < a_{kl} \end{cases}, \text{ виконується умова } a_{ij} < a_{kl}$$

Якщо матриця є узгодженою, то вона є і транзитивною. Ці вирази будуть вірні для кількісного критерія, але при використанні якісного критерія постають наступні питання: чи обов'язково, щоб матриця відповідала обом виразам, чи достатньо задовольняти лише третій, а можливо, не відповідала жодному з них? У наукових роботах Сааті стверджується, що мінімальна узгодженість є необхідною умовою, тоді як транзитивність має необов'язкову позицію.

Щоб проілюструвати правильність твердження, можна навести переконливий приклад: розглянемо сценарій, де команда *A*1 перемагає команду *A*2, яка, у свою чергу, перемагає команду *A*3, після чого команда *A*3 перемагає команду *A*1. В цьому випадку транзитивність не задоволняється, але ситуація цілком можлива [20].

- Обчислення середнього геометричного кожного рядку матриці, суми всіх отриманих значень та компонентів нормалізованого вектора пріорітетів.

Для цього використаємо наступні формули:

- середнього геометричного

$$a_n = \sqrt[n]{\text{добуток елементів } n-\text{го рядка}} \qquad (2.1)$$

- суми значень середнього геометричного всіх рядків

$$\sum a_i = a_1 + a_2 + \ldots + a_n \tag{2.2}$$

- компонентів нормалізованого вектора пріорітетів (НВП)

$$n - \text{й компонент} = \frac{a_n}{\sum a_i} \tag{2.3}$$

Для перевірки узгодженості необідно провести обчислення трьох характеристик, а саме:

- власного значення матриці, шляхом додавання суми кожного стовпця матриці помноженого на компоненти нормалізованого вектора пріорітетів

$$\lambda_{max} = \text{сума елементів } 1 - \text{го стовпця} \cdot 1 - \text{й КНВП} + \cdots + \text{сума елементів } n - \text{го стовпця} \cdot n - \text{й КНВП} \tag{2.4}$$

- індексу узгодженості, кількісного показника суперечливості результатів порівняння. Вплив суб'єктивних помилок, внесених експертами, сприяє розбіжностям та підвищенню значення цього індексу. Для ілюстрації суперечливості наведемо наступний приклад: експерт вказує, що альтернатива *A*1 поступається альтернативі *A*2, яка в свою чергу поступається *A*3, водночас з цим вказує, що *A*1 краще за *A*3.

Індекс узгодженності

$$\text{ІУ} = \frac{\lambda_{max} - n}{n - 1}, \tag{2.5}$$

де n – розмірність матриці.

Для того, щоб судження можна було вважати задовільними, значення індексів повинне бути в діапазоні:

- від 0 до 0.1 для матриці розмірністю 5 на 5 та вище;
- від 0 до 0.08 для матриць – 4 на 4
- від 0 до 0.05 для матриці – 3 на 3.

- обчислення відносної узгодженості передбачає визначення співвідношення між індексом узгодженості (ІУ) та показником випадкової узгодженості (ПВУ). Якщо отримане значення менше 0.1, дані можна вважати досить узгодженими. Загальна відносна узгодженість відповідає за зважене середнє значення ВУ по всім матрицям попарних порівнянь.

Відносна узгодженість

$$\text{ВУ} = \frac{\text{ІУ}}{\text{ПВУ}}, \tag{2.6}$$

де ПВУ – показник випадкової узгодженості або ж випадовий індекс, який залежить від розміру матриці.

Значення ПВУ представлені у таблиці 2.1.

Таблиця 2.1

Значення показнику випадкової узгодженості

Розмірність матриці	1	2	3	4	5	6	7	8	9	10
Випадковий індекс	0	0	0.58	0.9	1.12	1.24	1.32	1.41	1.45	1.49

Після обчислення відношення узгодженості необхідно провести перегляд оцінок у випадку, коли отримане значення більше 10-15% [20].

Аналогічно до попарного порівняння критеріїв, проводиться і порівняння варіантів по кожному з них. Після цього знову обчислюються три характеристики: власне значення матриці, індекс узгодженості та відносна узгодженість.

Наступним кроком виконується обчислення підсумкового значення пріорітетів для кожного варіанта

$$\text{К}(\text{В}_1) = \text{оцінка В}_1\text{по першому критерію} \cdot 1 - \text{й компонент НВП} + \\ + \ldots + \text{оцінка В}_1\text{ по } n - \text{му критерію} \cdot n - \text{й компонент НВП} \tag{2.7}$$

Аналогічно необхідно провести обчислення K для $\text{В}_2 \ldots \text{В}_n$.

Після цього проводиться вибір найкращого рішення проблеми, для якого значення *K* максимальне.

Перейдемо до практичної частини.

Спочатку оберемо ціль, критерії та альтернативи:

- ціль – обрати базову модель для дослідження;

- критерії: витрати на розгортання (*deployment costs*), захищеність (*security*) та ступінь зв’язності;

- альтернативи: повний граф, Ейлерів граф, Гамільтонів граф та дерево.

Тепер побудуємо ієрархію проблеми вибору базової моделі (рис. 2.5).

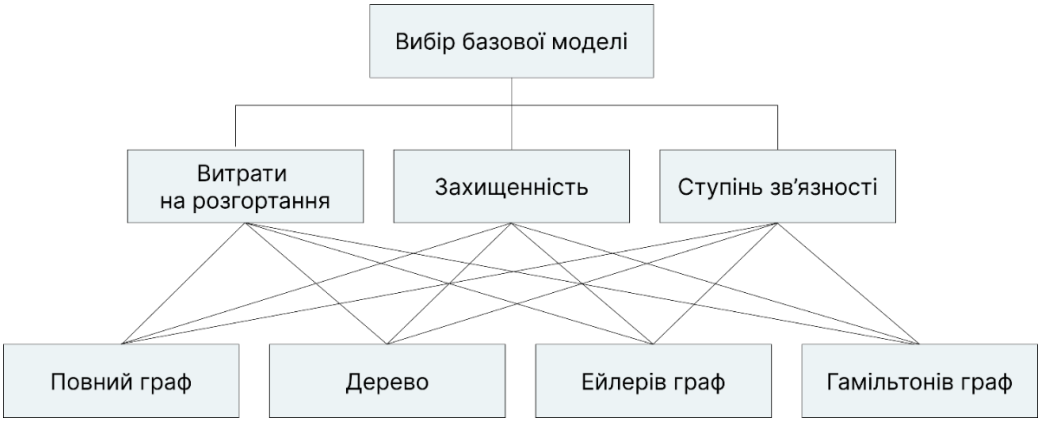


Рис. 2.5. Ієрархія проблеми

Для автоматизації проведення вибору за методом аналізу ієрархій створимо програму на мові *Python*.

В неї включимо основну функцію *calculate_characteristics*(*matrix*), яка буде обраховувати необхідні нам характеристики:

- *geometric_means* – середнє геометричне для кожного рядка матриці;

- *sum_geometric_means* – сума значень середнього геометричного всіх рядків;

- *normalized_vector* – компоненти нормалізованого вектора пріоритетів (НВП);

- *column_sums* – суми стовпців матриці;

- *eigenvalue* – значення матриці (λ_{max});

- *consistency_index* – індекс узгодженості (ІУ);

- *relative_consistency* – відносна узгодженість (ВП).

Для цього в неї будуть передаватися:

- матриця попарного порівняння по критеріям *matrix*_3*x*3;

- матриця попарного порівняння альтернатив по критерію «Витрати на розгортання» *matrix*_4*x*4_1;

- матриця попарного порівняння альтернатив по критерію «Захищенність» *matrix*_4*x*4_2;

- матриця попарного порівняння альтернатив по критерію «Ступінь зв'язності» *matrix*_4*x*4_3;

Після цього формується таблиця *total_table*, в яку входять НВП як критеріїв, так і варіантів по критеріям.

Використовуючи цикл *for*, код проходить за значеннями від 1 до довжини таблиці *total_table*. Для кожного значення *i* від 1 до довжини *total_table* він обчислює глобальний пріоритет, перемножуючи елементи першого рядка таблиці на відповідні елементи рядка з номером *i* (поточного рядка).

Наступним кроком побудуємо таблицю попарного порівняння критеріїв (табл. 2.2), де:

$К_1$ – ступінь зв'язності = 2;

$К_2$ – захищеність;

$К_3$ – витрати на розгортання;

СГ – середнє геометричне;

НВП – нормалізований вектор пріоритетів;

λ_{max} – власне значення матриці;

ІУ – індекс узгодженності;

ВУ – відношення узгодженності.

Обчислимо середнє геометричне як корінь третього степеню із добутку оцінок першого рядка таблиці 2.2 за формулою (2.1):

$$a_1 = \sqrt[3]{1 \cdot 5 \cdot 3} = 2.466212$$

$$a_2 = \sqrt[3]{1/5 \cdot 1 \cdot 1/3} = 0.405480$$

$$a_3 = \sqrt[3]{1/3 \cdot 3 \cdot 1} = 1$$

Після цього обчислимо суму значень середнього геометричного всіх рядків за формулою (2.2)

$$\sum a_i = 2.466212 + 0.405480 + 1 = 3.871692$$

Далі розрахуємо значення компонентів нормалізованого вектора пріоритетів (НВП) шляхом ділення середнього геометричного кожного з трьох рядків на їх суму за формулою (2.3)

$$1 - \text{й компонент} = \frac{2.466212}{3.871692} = 0.636985$$
$$2 - \text{й компонент} = \frac{0.405480}{3.871692} = 0.104729$$
$$3 - \text{й компонент} = \frac{1}{3.871692} = 0.258284$$

Перейдемо до обчислення власного значення матриці. Для цього виконаємо додавання кожного із стовпців таблиці 2.2 та кожне отримане значення помножимо на НВП за формулою (2.4)

$$\lambda_{max} = 1.533333 \cdot 0.636985 + 9 \cdot 0.104729 + 4.333333 \cdot 0.258284 = 3.038511$$

Після цього необхідно обчислити індекс узгодженності за формулою (2.5)

$$\text{ІУ} = \frac{3.038511 - 3}{3 - 1} = 0.019255$$

Обчислимо відносну узгодженість за формулою (2.6), поділивши отримане значення ІУ на показник випадкової узгодженості із таблиці 2.2, оскільки у нас матриця три на три, то необхідне значення рівне 0.58

$$\text{ВУ} = \frac{0.019255}{0.58} = 0.033199$$

Порівняємо отримане значення відносної узгодженості із 0.1. Оскільки воно менше, можемо сказати про те, що судження, які були виказані експертом,

приймаються. Тому в повторному заповненні матриці попарного порівняння не виникає необхідності.

Таблиця 2.2

Попарне порівняння критеріїв

	$К_1$	$К_2$	$К_3$	СГ	НВП
$К_1$	1	5	3	2.466212	0.636985
$К_2$	1/5	1	1/3	0.405480	0.104729
$К_3$	1/3	3	1	1	0.258284
Всього	1.533333	9	4.333333	3.871692	-
λ_{max}	3.038511	-	-	-	-
ІУ	0.019255	-	-	-	-
ВУ	0.033199	-	-	-	-

Після цього нам необхідно сформувати ще три таблиці, але вже попарного порівняння альтернатив по кожному із критерів (витрат на розгортання, захищенності, степеня зв'язності), де

- $В_1$ – дерево;
- $В_2$ – Ейлерів граф;
- $В_3$ – Гамільтонів граф;
- $В_4$ – повний граф.

Почнемо з матриці порівняння альтернатив по критерію «Витрати на розгортання» (табл. 2.3).

Обчислимо середнє геометричне як корінь четвертого ступеню із добутку оцінок першого рядка таблиці 2.3 за формулою (2.1)

$$a_1 = \sqrt[4]{1 \cdot 3 \cdot 3 \cdot 5} = 2.590020$$

$$a_2 = \sqrt[4]{1/3 \cdot 1 \cdot 1 \cdot 3} = 1$$

$$a_3 = \sqrt[4]{1/3 \cdot 1 \cdot 1 \cdot 3} = 1$$

$$a_4 = \sqrt[4]{1/5 \cdot 1/3 \cdot 1/3 \cdot 1} = 0.386097$$

Після цього обчислимо суму значень середнього геометричного всіх рядків за формулою (2.2)

$$\sum a_i = 2.590020 + 1 + 1 + 0.386097 = 4.976117$$

Далі розрахуємо значення компонентів нормалізованого вектора пріорітетів (НВП) шляхом ділення середнього геометричного кожного з чотирьох рядків на їх суму за формулою (2.3)

$$1-\text{й компонент} = \frac{2.590020}{4.976117} = 0.5204901$$

$$2-\text{й компонент} = \frac{1}{4.976117} = 0.200959$$

$$3-\text{й компонент} = \frac{1}{4.976117} = 0.200959$$

$$4-\text{й компонент} = \frac{0.386097}{4.976117} = 0.077590$$

Перейдемо до обчислення власного значення матриці. Для цього виконаємо додавання кожного із стовпців таблиці 2.2 та кожне отримане значення помножимо на НВП за формулою (2.4)

$$\lambda_{max} = 1.866666 \cdot 0.5204901 + 5.333333 \cdot 0.200959 + 5.333333 \cdot 0.200959 + \\ +12 \cdot 0.077590 = 4.046235$$

Після цього необхідно обчислимо індекс узгодженості за формулою (2.5)

$$\text{ІУ} = \frac{4.046235 - 4}{4 - 1} = 0.015411$$

Обчислимо відносну узгодженість за формулою (2.6), поділивши отримане значення ІУ на показник випадкової узгодженості із таблиці 2.2, оскільки у нас матриця чотири на чотири, то необхідне значення рівне 0.9

$$\text{ВУ} = \frac{0.0154115}{0.9} = 0.017123$$

Порівняємо отримане значення відносної узгодженості із 0.1. Оскільки воно менше, можемо сказати про те, що судження, які були виказані експертом, приймаються. Тому в повторному заповненні матриці попарного порівняння не виникає необхідності.

Таблиця 2.3

Попарне порівняння варіантів по критерію «Витрати на розгортання»

$К_3$	$В_1$	$В_2$	$В_3$	$В_4$	СГ	НВП
$В_1$	1	3	3	5	2.590020	0.5204901
$В_2$	1/3	1	1	3	1	0.200959
$В_3$	1/3	1	1	3	1	0.200959
$В_4$	1/5	1/3	1/3	1	0.386097	0.077590
Всього	1.866666	5.333333	5.333333	12	4.976117	-
λ_{max}	4.046235	-	-	-	-	-
ІУ	0.015411	-	-	-	-	-
ВУ	0.017123	-	-	-	-	-

Перейдемо до формування матриці порівняння альтернатив по критерію «Захищеність» (табл. 2.4).

Обчислимо середнє геометричне як корінь четвертого ступеню із добутку оцінок першого рядка таблиці 2.4 за формулою (2.1)

$$a_1 = \sqrt[4]{1 \cdot 3 \cdot 3 \cdot 5} = 2.590020$$

$$a_2 = \sqrt[4]{1/3 \cdot 1 \cdot 1/3 \cdot 3} = 0.759835$$

$$a_3 = \sqrt[4]{1/3 \cdot 3 \cdot 1 \cdot 5} = 1.495349$$

$$a_4 = \sqrt[4]{1/5 \cdot 1/3 \cdot 1/5 \cdot 1} = 0.339809$$

Після цього обчислимо суму значень середнього геометричного всіх рядків за формуло. (2.2)

$$\sum a_i = 2.590020 + 0.759835 + 1.495349 + 0.339809 = 5.185013$$

Далі розрахуємо значення компонентів нормалізованого вектора пріорітетів (НВП) шляхом ділення середнього геометричного кожного з чотирьох рядків на їх суму за формулою (2.3)

$$1-\text{й компонент} = \frac{2.590020}{5.1850133} = 0.499520$$

$$2-\text{й компонент} = \frac{0.759835}{5.1850133} = 0.146544$$

$$3-\text{й компонент} = \frac{1.495349}{5.1850133} = 0.288398$$

$$4-\text{й компонент} = \frac{0.339809}{5.1850133} = 0.0655367$$

Перейдемо до обчислення власного значення матриці. Для цього виконаємо додавання кожного із стовпців таблиці 2.3 та кожне отримане значення помножимо на НВП за формулою (2.4)

$$\lambda_{max} = 1.866666 \cdot 0.499520 + 7.333333 \cdot 0.146544 + 4.533333 \cdot 0.288398 + \\ + 14 \cdot 0.0655367 = 4.232018$$

Після цього обчислимо індекс узгодженості за формулою (2.5)

$$\text{ІУ} = \frac{4.232018 - 4}{4 - 1} = 0.077339$$

Обчислимо відносну узгодженість за формулою (2.6), поділивши отримане значення ІУ на показник випадкової узгодженості із таблиці 2.2, оскільки у нас матриця чотири на чотири, то необхідне значення рівне 0.9

$$\text{ВУ} = \frac{0.077339}{0.9} = 0.085932$$

Порівняємо отримане значення відносної узгодженості із 0.1. Оскільки воно менше, можемо сказати про те, що судження, які були виказані експертом, приймаються. Тому для повторного заповнення матриці попарного порівняння не виникає необхідності.

Таблиця 2.4

Попарне порівняння варіантів по критерію «Захищеність»

$К_2$	$В_1$	$В_2$	$В_3$	$В_1$	СГ	НВП
$В_1$	1	3	3	5	2.590020	0.499520
$В_2$	1/3	1	1/3	3	0.759835	0.146544
$В_3$	1/5	3	1	5	1.495348	0.288398
$В_4$	1/5	1/5	1/5	1	0.339808	0.065537
Всього	1.866666	7.333333	4.533333	14	5.185013	-
λ_{max}	4.232018	-	-	-	-	-
ІУ	0.077339	-	-	-	-	-
ВУ	0.085932	-	-	-	-	-

Перейдемо до формування матриці порівняння альтернатив по критерію «Ступінь зв’язності» (табл. 2.5).

Таблиця 2.5

Попарне порівняння варіантів по критерію «Ступінь зв’язності»

$К_1$	$В_1$	$В_2$	$В_3$	$В_4$	СГ	НВП
$В_1$	1	1/7	1/7	1/3	0.287190	0.0484711
$В_2$	7	1	1/3	5	1.848147	0.311924
$В_3$	7	3	1	5	3.201085	0.540268
$В_4$	3	1/5	1/5	1	0.588566	0.099336
Всього	18	4.342857	1.6761904	11.333333	5.924990	-
λ_{max}	4.258525	-	-	-	-	-
ІУ	0.086175	-	-	-	-	-
ВУ	0.095750	-	-	-	-	-

Обчислимо середнє геометричне як корінь четвертого ступеню із добутку оцінок першого рядка таблиці 2.5 за формулою (2.1)

$$a_1 = \sqrt[4]{1 \cdot 1/7 \cdot 1/7 \cdot 1/3} = 0.287190$$

$$a_2 = \sqrt[4]{7 \cdot 1 \cdot 1/3 \cdot 5} = 1.848147$$

$$a_3 = \sqrt[4]{7 \cdot 3 \cdot 1 \cdot 5} = 3.201085$$

$$a_4 = \sqrt[4]{3 \cdot 1/5 \cdot 1/5 \cdot 1} = \ 0.588566$$

Після цього обчислимо суму значень середнього геометричного всіх рядків за формулою (2.2)

$$\sum a_i = 0.287190 + \ 1.848147 + 3.201085 \ + \ 0.588566 = 5.924991$$

Далі розрахуємо значення компонентів нормалізованого вектора пріорітетів (НВП) шляхом ділення середнього геометричного кожного з чотирьох рядків на їх суму за формулою (2.3)

$$1 - \text{й компонент} \ = \ \frac{0.287190}{5.924990} = 0.0484711$$

$$2 - \text{й компонент} \ = \ \frac{1.848147}{5.924990} = 0.311924$$

$$3 - \text{й компонент} \ = \ \frac{3.201085}{5.924990} = 0.540268$$

$$4 - \text{й компонент} \ = \ \frac{0.588566}{5.924990} = \ 0.099336$$

Перейдемо до обчислення власного значення матриці. Для цього виконаємо додавання кожного із стовпців таблиці 2.5 та кожне отримане значення помножимо на НВП за формулою (2.4)

$$\lambda_{max} = 18 \cdot \ 0.048471 + 4.342857 \cdot 0.311924 + 1.6761904 \cdot \ 0.540268 \ + 11.33333 \cdot$$
$$\cdot 0.099336 = \ 4.258525$$

Після цього обчислимо індекс узгодженості за формулою (2.5)

$$\text{ІУ} \ = \ \frac{4.258525 - 4}{4 - 1} \ = \ \ 0.086175$$

Обчислимо відносну узгодженість за формулою (2.6), поділивши отримане значення ІУ на показник випадкової узгодженості із таблиці 2.2, оскільки у нас матриця чотири на чотири, то необхідне значення рівне 0.9

$$\text{ВУ} = \frac{0.086175}{0.9} = 0.095750$$

Порівняємо отримане значення відносної узгодженості із 0.1. Оскільки воно менше, можемо сказати про те, що судження, які були виказані експертом, приймаються. Тому в повторному заповненні матриці попарного порівняння не виникає необхідності.

Сформуємо таблицю для підсумкових значень пріорітетів (табл. 2.6). В перший рядок внесемо значення компонентів нормалізованого вектора пріорітетів, представлених в таблиці попарного порівняння критеріїв (див. табл. 2.2). Починаючи з другого рядка, заповнюємо перший стовпець значеннями НВП з матриці попарного порівняння альтернатив по критерію $К_1$ (див. табл. 2.5), аналогічним чином заповнюємо решту комірок.

Таблиця 2.6

Обчислення підсумкових значень пріорітетів

	$К_1$	$К_2$	$К_3$	ПЗП
	0.636985	0.104729	0.258284	
$В_1$	0.048471	0.520490	0.499520	0.214404
$В_2$	0.311924	0.200959	0.146544	0.257587
$В_3$	0.540268	0.200959	0.288398	0.439678
$В_4$	0.099336	0.077590	0.065536	0.088328

Тепер обчислимо глобальний пріоритет К(В1) шляхом множення НВП $К_1$ на НВП B_1 по кожному із критеріїв за формулою (2.7)

$$К(В_1) = 0.636985 \cdot 0.048471 + 0.104729 \cdot 0.520490 + 0.258284 \cdot 0.499520 = \\ = 0.214404$$

Анагічним чином обчислюємо решту глобальних пріорітетів:

$$К(В_2) = 0.636985 \cdot 0.311924 + 0.104729 \cdot 0.200959 + 0.258284 \cdot 0.146544 = \\ = 0.257587$$

$$К(В_3) = 0.636985 \cdot 0.540268 + 0.104729 \cdot 0.200959 + 0.258284 \cdot 0.288398 = \\ = 0.439678$$

$$К(В_4) = 0.636985 \cdot 0.099336 + 0.104729 \cdot 0.077590 + 0.258284 \cdot 0.065536 = \\ = 0.214404$$

Виконаємо сортування отриманих значень глобальних пріоритетів в порядку зменшення та побачимо, що найбільше значення відповідає альтернативі «Гамільтонів граф», а саме 0.439678. Враховуючи це, можна зробити висновок про те, що в якості базової моделі для дослідження обирається саме він.

Висновки за розділом

В другому розділі було висунуто два критерії: ступінь зв’язності повинен бути рівний двом (згідно до технології *Token Ring*); зроблено припущення про те, що збільшення сумарної довжини кабелю призводить до збільшення випромінювання, що в свою чергу призводить до зменшення захищенності мережі. Після цього було сформовано набір моделей, серед яких: повний граф, дерево, Гамільтонів граф та Ейлерів граф. Для вибору базової моделі було використано метод аналізу ієрархій, тобто побудовано ієрархію проблеми, таблиці попарного порівняння як критеріїв, так і альтернатив по кожному з критеріїв, а також обчислено підсумкові значення пріорітетів. В результаті було обрано Гамільтонів граф як базову модель для дослідження, оскільки значення глобального пріоритету саме у цієї альтернативи в два рази вище, ніж у решти, а саме 0.439678 проти 0.214404 (дерево), 0.257587 (Ейлерів граф) та 0.214404 (повний граф).

РОЗДІЛ 3
ВИБІР І РОЗРОБКА ІНСТРУМЕНТАРІЮ ДОСЛІДЖЕННЯ РОЗРОБЛЕНОЇ МОДЕЛІ ПРОЄКТУВАННЯ ТОПОЛОГІЇ ЛКМ СТАНДАРТУ *IEEE* 802.5

3.1. Систематизація програмно-алгоритмічних способів дослідження розробленої моделі проєктування топології ЛКМ стандарту *IEEE* 802.5

Оскільки задача комівояжера є *NP*-повною, що свідчить про відсутність найефективнішого методу її вирішення для всіх можливих вхідних даних за поліноміальний час, необхідно провести попереднє дослідження методів, на основі якого потім обрати найбільш оптимальний.

Для вирішення задачі комівояжера можна використати:

- точні алгоритми: метод повного перебору, метод розгалуджень та меж, метод Літтла, алгоритм Хелда-Карпа;

- евристичні алгоритми: метод найближчого сусіда, метод найбільш дешевого включення, дерев'яний алгоритм та інші;

- метаевристичні алгоритми: генетичний алгоритм, мурашиний алгоритм, алгоритм імітації відпалу.

При використанні методу повного перебору відбувається перегляд всіх можливих маршрутів та пошук найбільш оптимального. Використання цього методу має суттєвий недолік – при збільшенні компонентів мережі зростають складність обчислення та часові витрати [21]. Наприклад, якщо нам потрібно знайти найбільш оптимальний шлях для кількості компонентів рівній 6, буде існувати 720 варіантів маршруту.

Метод гілок та меж вважається одним із найбільш ефективних методів, в якому відбувається відсікання непридатних маршрутів, які мають довжину більшу, ніж найбільш оптимальний шлях на даний момент. Для цього виконується побудова дерева, в якому на кожному етапі відбувається розділення на гілки та оцінка нижньої

межі. В тому випадку, коли нижня межа гілки більша, ніж поточна оптимальна відстань, вона відсікається, в іншому випадку гілка вважається новим кандидатом на оптимальне вирішення задачі. Цей метод має рекурсивну структуру та продовжує розбиття на гілки до тих пір, поки не виконається умова зупинки [22].

Метод Літтла полягає в тому, що спочатку знаходяться мінімальні елементи в кожному рядку та їх значення віднімається від решти елементів. Таким же чином виконується проходження по стовпцям з відсутними нулями. Після цього обчислюються коефіцієнти кожного нульового елемента шляхом додавання найменших елементів в рядку та стовпці. Обирається елемент з найбільшим значенням коефіцієнту, та якщо їх декілька, обирається будь-який. Ця дуга вноситься в гамільтонів контур та виконується видалення елементів рядка та стовпця, в якому знаходився цей компоненту. Цей цикл повторюється доти, поки порядок матриці не буде рівним двом. В завершення алгоритму ці дві дуги вносяться в гамільтонів контур. Слід прийняти до уваги те, що після кожного циклу виконується підрахунок значення нижньої межі, яке рівне сумі всіх видалених елементів [23].

Алгоритм Хелда-Карпа використовує динамічне програмування для ефективного обчислення оптимальних шляхів. Більш швидкий, ніж метод повного перебору, але вимагає більших витрат памяті. Полягає в розбитті задачі на підзадачі, тобто спочатку визначаються короткі шляхи від початкової вершини до сусідніх, після цього – до інших вершин через одну, дві та більше проміжних вершин. В завершенні виконується вибір найкоротшого шляху із всіх отриманих варіантів [24].

Ідея жадібного алгоритму, до яких можна віднести метод найближчого сусіда та метод найбільш дешевого включення, для поставленої задачі полягає у виборі більш близько розташованого компонента, який ще не був пройдений, на кожному етапі його проходження. Цей метод вважається простим та швидким, але він не гарантує знаходження оптимального шляху [25].

Мурашиний метод відноситься до ймовірносних алгоритмів пошуку оптимального шляху. Цей метод полягає в знаходженні найкоротшого шляху на основі концентрації феромонів, виділених мурахою на шляху. Чим коротше шлях, тим більше феромонів, та навпаки. Зі збільшенням кількості феромонів збільшується

і ймовірність вибору мурахою певного маршруту. Також слід взяти до уваги процес випаровування, оскільки на більш довгих шляхах зменшується активність мурах, яка призводить до втрати щільності сліду феромонів.

Порядок виконання алгоритму передбачає попереднє задання кількості мурах та феромону, а також як він випаровується. Після цього кожна з мурах прокладає незалежний маршрут, по завершенню обходу якого відбувається обчислення його відстані. Наступним кроком слід провести оновлення феромонів на шляхах найкращого рішення та перевірити умову зупинки [26].

Метод імітації відпалу також відноситься до ймовірносних алгоритмів та використовується для вирішення задач оптимізації. На початку виконання алгоритму випадковим чином генерується початкове рішення, тобто шлях між компонентами ЛКМ. Після цього здійснюється його невелика зміна, наприклад, зміна порядку декількох компонентів [27].

Наступним кроком виконується обчисленням сумарної відстані від початкового та кінцевого компонента кожного з доступних на даному етапі рішень та отримані значення модифікованих рішень порівнюються з поточним. В тому випадку, коли сумарна відстань одного з модифікованих шляхів менша за значення на поточному, виконується оновлення рішення. При використанні цього методу гірший шлях також може прийнятися з певною ймовірністю [27].

Цей алгоритм передбачає повернення на попередній крок з подальшою заміною поточного кращого рішення на гірше. Якість отриманого результату обумовлюється вибором початкового шляху, механізмами генерації нових рішень та умовою завершення, яка може бути максимальною кількістю кроків алгоритму з фіксованим результатом або ж загальна кількість ітерацій.

В результаті дослідження можна зробити наступні висновки:

- використання точних алгоритмів є прийнятним у випадку використання невеликих вхідних даних та коли точність є критичним параметром. Проте збільшення розміру даних призводить до значних витрат часу та обчислювальних ресурсів, що робить їх застосування непрактичним для великих задач. Точні

алгоритми надають гарантію знаходження найкоротшого маршруту, що відрізняє їх від інших методів.

- евристичні методи, хоча не гарантують оптимального рішення, є більш ефективними та швидкими у порівнянні з точними методами. Вони можуть знайти дуже наближені рішення за прийнятний час.

- метаевристичні методи зазвичай надають більш оптимальні рішення, ніж евристичні, але їх результати залежать від налаштування та адаптації під різні вхідні дані індивідуальних гіперпараметрів.

Тобто вибір методу залежить від наступних параметрів: розміру набору вхідних даних, часу на виконання, точності результату та доступної продуктивності обчислень. Перед нами стоїть задача побудови топології ЛКМ для наземного сегменту БАК, отже в цьому випадку нам доцільніше використати метаевристичні методи, а саме генетичний алгоритм, оскільки нам важлива точність на великій кількості елементів у вхідному наборі при достатньому часі та великій продуктивності.

3.2. Розробка налаштування генетичного алгоритму для дослідження розробленої моделі

Компоненти ЛКМ представимо у вигляді системи трьох координат *x*, *y*, *z*, які безпосередньо відповідають за широту, довготу та висоту над рівнем моря. Відстань між двома компонентами в тримірному евклідовому просторі можна обчислити як квадратний корінь з суми квадратів різниць кожної з координат

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

Для розв'язання задачі комівояжера за допомогою генетичних алгоритмів спочатку потрібно сформувати початкову популяцію. Щоб це зробити, представимо хромосому як рядок з (n-1) генів, де кожен ген представляє номер компоненту ЛКМ. Послідовність генів відображає порядок проходження компонентів. Оскільки компонент 1 є одночасно початковим і кінцевим, не виникає необхідності у записі

його в першому та останньому гені хромосоми. Таке представлення хромосоми забезпечує коретність рішення, так як початковий та кінцевий компонент будуть завжди однакові.

Перейдемо до формування початкової популяції. В рамках окремої хромосоми номери компонентів не повинні повторюватися, оскільки кожен з них повинен бути пройдений лише один раз.

Наступним кроком є схрещування особин, в результаті якого створюються нові нащадки, тобто нові можливі рішення задачі коміявожера. Для цього спочатку формуються батьківські пари та на основі одного з методів схрещування формуються хромосоми нащадків шляхом комбінації генів хромосом батьківських особин. Відбір батьків можна виконати наступними методами: метод за правилом рулетки, стохастична універсальна вибірка, ранговий метод, масштабування придатності, турнірний відбір та випадковий відбір.

Відбір за правилом рулетки

Відбір за правилом рулетки або ж відбір пропорційної придатності (англ. *fitness proportionate selection* (*FPS*)) передбачає побудову колеса рулетки, в якій за кожен сектор відповідає окрема особа [29]. Розмір сектора залежить від значення відношення придатності окремої особини до середньої придатності.

Принцип відбору за правилом рулетки представлено на рис. 3.1.

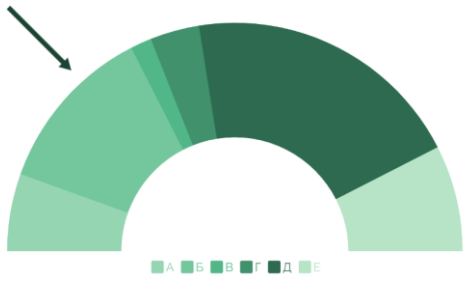


Рис. 3.1. Відбір за правилом рулетки

Відбір проводиться наступним чином: обирається фіксована точка та запускається колесо, особина, на чий сектор вказує стрілка після зупинки колеса, обирається батьком. Кількість запусків колеса залежить від кількості особин в

популяції. Чим більше значення придатності, тим вища ймовірність відбору саме цієї особини в якості батьківської.

Для заданої популяції середнє значення буде рівне сумі придатностей особин, поділене на їх кількість, тобто 113. Щоб отримати долю, поділимо значення придатності окремої особини, наприклад А, на середнє значення 113, та отримаємо: 0.10. Представимо це значення у вигляді відсотків, помноживши його на 100. Відношення придатності окремої особини до середньої придатності представлено в табл. 3.1.

Таблиця 3.1

Відношення придатності окремої особини до середньої придатності

Особина	Придатність	Доля, %
А	12	11
Б	27	24
В	4	3
Г	8	7
Д	45	40
Е	17	15

В основному в цьому методі присутні дві проблеми:

- В тому випадку, коли в вибірці присутні особини з досить великою функцією придатності на фоні інших, зазвичай будуть обиратися саме вони. Таким чином, виникає втрата різноманітності популяції;

- У випадку, коли у функції приналежності мала дисперсія, тобто значення досить схожі між собою, вибір виконується майже випадково.

Стохастична універсальна вибірка

Стохастична універсальна вибірка (англ. *stochastic universal sampling – SUS*) модифікаваний варіант попереднього методу. Модифікація полягає в тому, що використовується не одна фіксована точка, а декілька, що зумовлює більш рівномірний відбір батьківских особин за один оберт рулетки [28]. Принцип роботи стохастичної універсальної вибірки представлено на рис. 3.2.

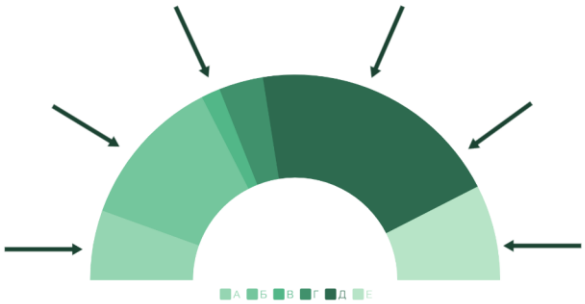


Рис. 3.2. Стохастична універсальна вибірка

Масштабування придатності

Цей метод реалізує масштабування значень придатності до певного інтервалу.

Представимо використання цього методу на прикладі. Задамо значення придатності для шести особин в табл. 3.2.

Таблиця 3.2

Придатності особин

Особина	Придатність
А	12
Б	27
В	4
Г	8
Д	45
Е	17

Тобто значення придатності представлені в діапазоні від 4 до 45 і їх потрібно перевести в новий діапазон, наприклад, від 30 до 80. Для цього необхідно використати математичну формулу

$$f^* = a \cdot f + b, \tag{3.1}$$

де f – придатність; a та b – коефіцієнти перетворення.

Для знаходження коефіцієнтів a та b сформуємо систему рівняння за формулою (3.1):

$$\begin{cases} a \cdot f_{min} + b = d_{min} \\ a \cdot f_{max} + b = d_{max} \end{cases}, \tag{3.2}$$

де f_{min} та f_{max} – значення мінімальної та максимальної придатності, в цьому випадку 4 та 45; d_{min} та d_{max} – початкове та кінцеве значення діапазону, тобто 30 та 80.

З цієї системи (3.2) отримаємо наступні формули для знаходження коефіцієнтів a та b

$$a = \frac{d_{min} - d_{max}}{f_{min} - f_{max}}, \tag{3.3}$$

$$b = d_{min} - a \cdot f_{min} \tag{3.4}$$

Підставимо значення в формулу (3.3)

$$a = \frac{30 - 80}{4 - 45} = \frac{-50}{-41} = 1.22$$

Та формулу (3.4)

$$b = 30 - 1.22 \cdot 4 = 25.12$$

Тепер підставимо отримані значення коефіцієнтів та придатність окремих особин в формулу (3.1) та обчислимо нові значення придатності.

Наприклад, для особини Г нове значення придатності буде наступним:

$$f^* = a \cdot f + b = 1.22 \cdot 8 + 25.12 = 34.88$$

Для додаткової наглядності представимо відповідні зміни на кругових діаграмах (рис. 3.3 – 3.4).

Також, отримані результати представимо у вигляді графіку, де синіми точками представлені початкові значення придатності f, а червоними хрестиками – нові значення (рис. 3.5).

Рис. 3.3. До масштабування

Рис. 3.4. Після масштабування

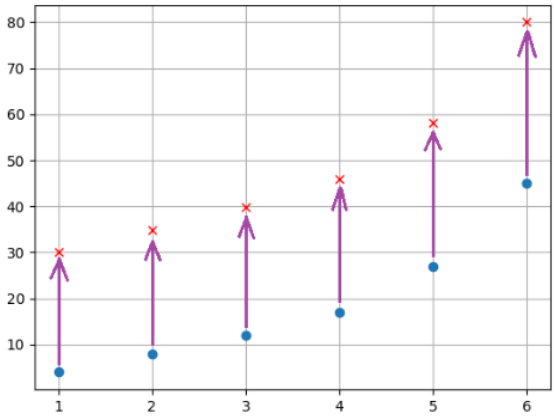


Рис. 3.5. Результат масштабування придатності

Цей метод надає рівномірний відбір батьківських особин, оскільки навіть у особин з найгіршим значення придатності існує можливість перейти у нову вибірку та забезпечити різноманість хромосом.

Турнірний метод

При використанні турнірного методу виконується випадковий вибір декількох претендентів, після цього на основі функції приналежності обирається найбільш придатний (рис. 3.6) [29]. Процес повторюється, поки кількість батьків не буде рівна розміру популяції. Цей метод вважається одним із найпопулярніших, оскільки є ефективним при мінімальних часових витратах. До цього ж, у нього є ще одна

перевага, оскільки він може працювати не тільки з додатними значеннями функції приналежності, а і з від’ємними.

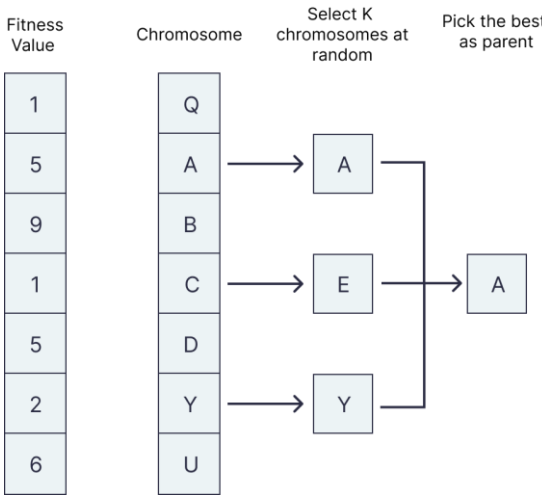
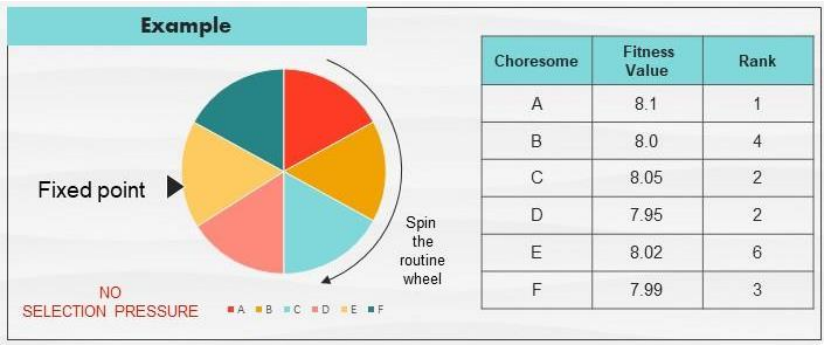


Рис. 3.6. Турнірний метод

Ранговий метод

Ранговий метод полягає у сортуванні особин в порядку зростання функції приналежності, кожній з особин призначається ранг та на основі обчисленої частки відносно рангу виконується відбір пар батьків [28]. Цей метод ефективний у наступних випадках: при високій придатності окремих особин та при приблизно однаковій придатності.

На рис. 3.7 представлено приклад рангованого методу при максимально однакових значеннях функціїї приналежності.



Choresome	Fitness Value	Rank
A	8.1	1
B	8.0	4
C	8.05	2
D	7.95	2
E	8.02	6
F	7.99	3

Рис. 3.7. Рангований метод

Візьмемо популяцію задану в табл. 3.3 та проведемо сортування значень придатності в порядку зростання, задавши кожній особини відповідний ранг від 1 до

6 в даному випадку. Після цього обчислимо середнє значення рангів та отримаємо 21. Для обчислення долі рангу ділимо ранг на середнє значення.

Таблиця 3.3

Обчислення долі рангу

Особина	Придатність	Ранг	Доля рангу, %
А	12	3	14
Б	27	5	23
В	4	1	4
Г	8	2	9
Д	45	6	28
Е	17	4	19

Ключовою перевагою вважається те, що при використанні рангованого відбору у кожної особини існує шанс бути обраною, тобто цей метод веде себе більш надійно, оскільки призначає придатність на основі рангу а не пропорційно.

Існує два типи рангового відбору: лінійний та нелінійний. Різниця між ними полягає в тому, що при лінійному відборі кожній з особин призначається ймовірність вибору, пропорційно її рангу, а в нелінійному – непропопорційно.

Існують наступні методи схрещування: одноточке, двухточкове та *k*-точкове, рівномірне, впорядковане, схрещування зміщенням та імітація двійкового зміщування.

Одноточкове схрещування

Суть одноточкового схрещування полягає в наступному (рис. 3.8):

1. Випадковим чином обирається точка розриву всередині хромосоми;
2. Створюються два нащадки, перший з яких отримує гени першого з батьків до вибраної точки, решта генів отримується від іншого батька, які відсутні в першій половині хромосоми першого батька. Аналогічним чином формуються хромосоми іншого нащадка [31].

Після завершення цієї процедури популяція збільшується в два рази.

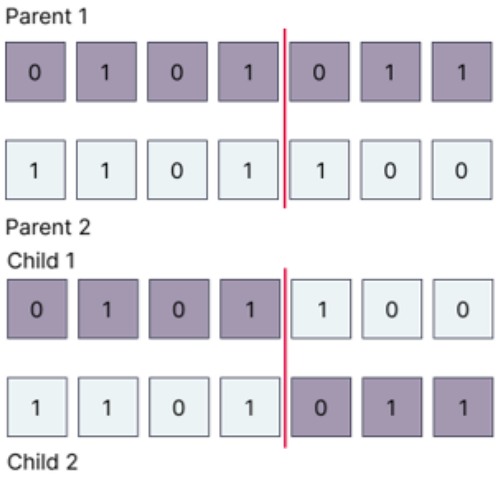


Рис. 3.8. Одноточкове схрещування

Рівномірне схрещування

При використанні цього методу схрещування виконується шляхом випадково відбору окремих пар генів із хромосом. В найпростішому варіанті виконується обмін генами з заданою ймовірністю, але частіше обираються випадкові гени першого та другого батьків та проводиться їх обмін (рис. 3.9) [30].

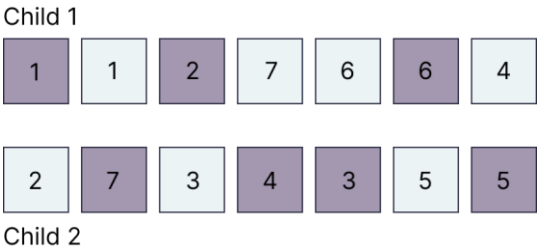


Рис. 3.9. Рівномірне схрещування

Впорядковане схрещування

Існують випадки, коли хромосоми батьків складаються з чисел, які не повторюються, але після схрещування це змінюється. В деяких задачах це є недопустимим, наприклад, в задачі комівояжера. Для вирішення даної проблеми можна використати метод впорядкованого схрещування, який полягає в наступному: виконуємо, наприклад, двухточкове схрещування та проходимо по всім генам першої батьківської особини, починаючи з другої точки, та записуємо значення, які відстутні в хромосомі нащадка (рис. 3.10) [30][31]. Аналогічним чином проводимо схрещування генів другого батька та другого нащадка.

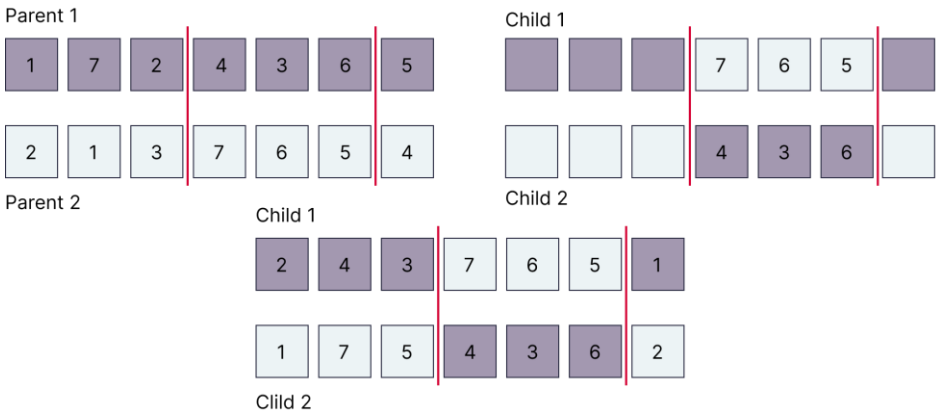


Рис. 3.10. Впорядковане схрещування

Двухточкове та *k*-точкове схрещування

Більш ефективний метод в порівнянні з одноточковим схрещуванням. В даному методі обирається випадковим чином дві і більше точки з урахуванням двух умов: вони не є межами хромосоми і не співпадають одна з одною.

Реалізувати метод з двома точками розриву можна з використанням двох операторів одноточкового схрещування, тобто спочатку виконати обмін між першими двома частинами хромосом, які розділяє перша точка, після чого аналогічну операцію необхідно виконати між кінцевими двома частинами, розділених другою точкою (рис. 3.11) [30].

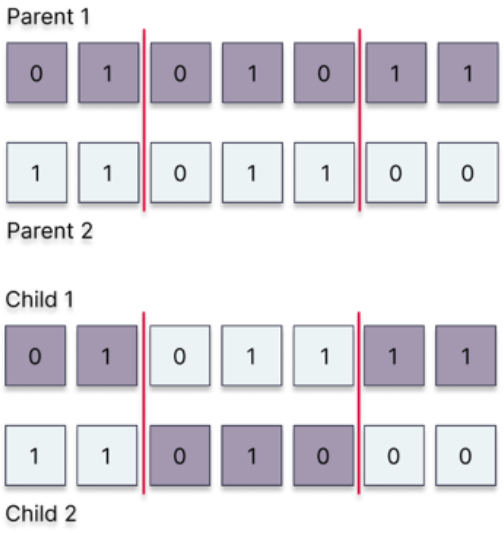


Рис. 3.11. Двухточкове схрещування

Схрещування зміщенням

Цей метод застосовується при використанні дійсних значень в генах. Спочатку обираються дві батьківські особини $p1$ та $p2$. Виконується проходження пар генів кожного з батьків та обчислюється інтервал

$$[p1_i - \alpha(p2_i - p1_i); p2_i + \alpha(p2_i - p1_i)]$$

Значення коефіцієнта α може змінюватися від 0 до 1. На практичні частіше обирають 0.5. При 0 інтервал співпадає зі значенням генів батьківських особин, при 0.5 – збільшується вдвічі, при 1 – більше, ніж в 4 рази.

Представимо залежність інтервалу від значення коефіцієнта (рис. 3.12)

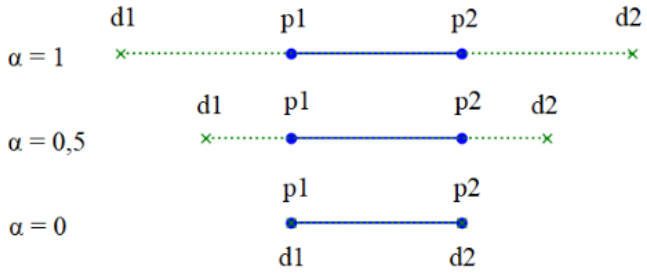


Рис. 3.12. Залежність інтервалу від значення коефіцієнта

На заданому інтервалі випадково обирається два значення для нащадків. Аналогічним чином призначаються інші гени.

Імітація двійкового схрещування

Цей метод імітує властивості одноточкового схрещування при використанні двійкових хромосом. Однією з особливостей є те, що середні значення хромосом батьків та нащадків рівні.

Для обчислення генів нащадків використовуються

$$o1 = \frac{1}{2} \cdot [(1 + \beta) \cdot p_1 + (1 - \beta) \cdot p_2],$$

$$o2 = \frac{1}{2} \cdot [(1 - \beta) \cdot p_1 + (1 + \beta) \cdot p_2]$$

Значення коефієнту розподілення обирається випадково для кожної пари нащадків. При $\beta = 1$, нащадки ідентичні батькам, при $\beta < 1$ або $\beta > 1$ середні значення нащадків будуть, відповідно, віддалятися або зближатися.

Для того, щоб забезпечити схожість нащадків з батьками, можна скористатися системою для обчислення коефіцієнту розподілення:

$$\beta = \begin{cases} 2u \cdot \frac{1}{\eta + 1}, u \leq 0.5 \\ (\frac{1}{2}(1-u))^{\frac{1}{\eta+1}}, u > 0.5 \end{cases}$$

Значення u генерується випадковим чином в діапазоні від 0 до 1, а параметр η визначає схожість батьків та нащадків та приймає значення 10 або 20 в більшості випадків.

Наступним кроком є виконання мутації особин, в результаті якої відбуваються зміни в генетичному матеріалі особини та забезпечується різноманість в популяції. Кількість особин, які будуть піддаватися мутації, визначається налаштуваннями генетичного алгоритму. Для вибору ймовірності мутації можна обрати варіанти $1/PS$ або $1/CL$, де PS – розмір популяції, а CL – довжина хромосоми.

Розглянемо більш детально варіанти мутації, а саме: інвертування бітів, мутація обміном, мутація оберненням, мутація перетасуванням, мутація для дійсних чисел.

Інвертування бітів

Цей метод передбачає декілька варіантів застосування. В одній з реалізацій обирається з певною ймовірністю хромосома, яка буде піддаватися мутації, та виконується зміна значення випадкового гену на інверсне (рис. 3.12). В іншій реалізації, можливо виставити ймовірність для вибору як хромосоми, так і гену [32].

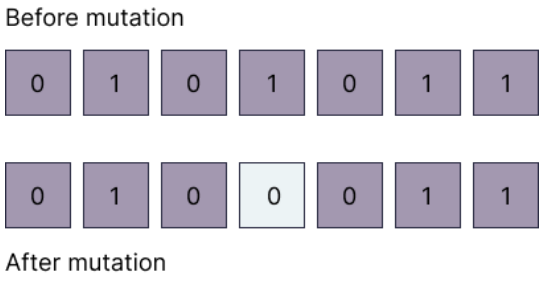


Рис. 3.13. Інвертування бітів

Мутація обміном

Випадковим чином обираються декілька генів та виконується обмін місцями їх значень (рис. 3.14) [32]. Цей варіант мутації можна використати для впорядкованих списків.

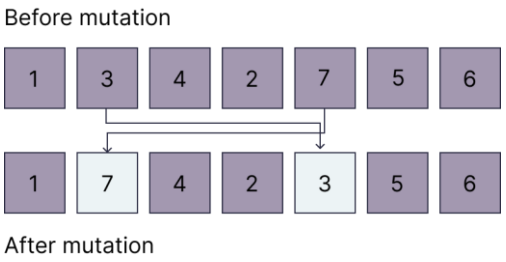


Рис. 3.14. Мутація обміном

Мутація оберненням

Модифікація мутації обміном, але з відмінністю у виборі генів. В цьому випадку обирається саме непреривна послідовність, яка згодом змінює своє розташування на обернене (рис. 3.15) [32].

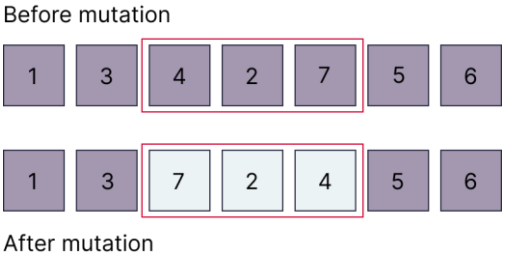


Рис. 3.15. Мутація оберненням

Мутація для дійсних чисел

Цей метод передбачає зміну значення випадково обраного гену на випадкове число, яке повинне знаходитися в межах допустимого діапазону значень та бути близьком до початкового (рис. 3.16).

Нове значення гену можна змоделювати, використавши нормальний закон розподілення. В якості математичного очікування обирається початкове значення гену, яке обмежується діапазоном допустимих значень (рис. 3.17) [32].

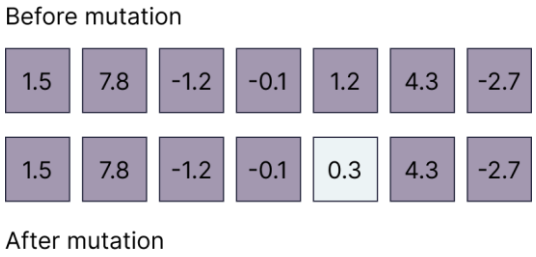


Рис. 3.16. Мутація для дійсних чисел

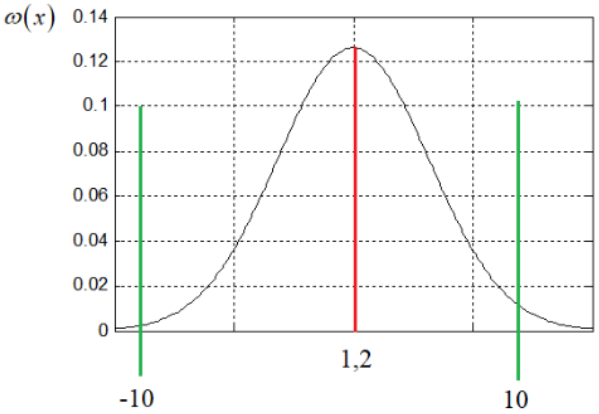


Рис. 3.17. Математичне очікування

Мутація перетасуванням

Таким же чином, як і попередньому варіанті, випадковим чином обирається непреривна послідовність. Але розташування змінюється не в оберненому напрямку, а випадковому (рис. 3.18) [32].

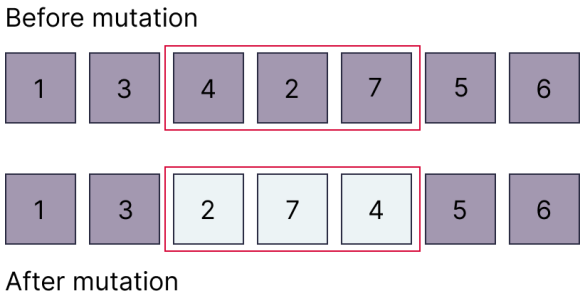


Рис. 3.18. Мутація перетасуванням

Перейдемо до селекції, в якій відбувається відбір найбільш придатних особин для створення наступного покоління на основі функції придатності.

Для цього існує два варіанта: відбирати особин за їх віком або ж за значенням функції приналежності.

В першому варіанті відбір грунтується на тому, що особина може допускатися в наступну популяцію лише протягом кінцевої кількості поколінь. Після цього навіть

незважаючи на те, що вона може мати найбільше значення ФП, вона видається із популяції. Представимо це на прикладі, зліва у нас представлений вік кожної з особин. В цьому випадку, найстарші особини, а саме *P*4 та *P*7 виключаються із популяцію, а для решти – вік збільшується на одиницю (рис. 3.19) [29].

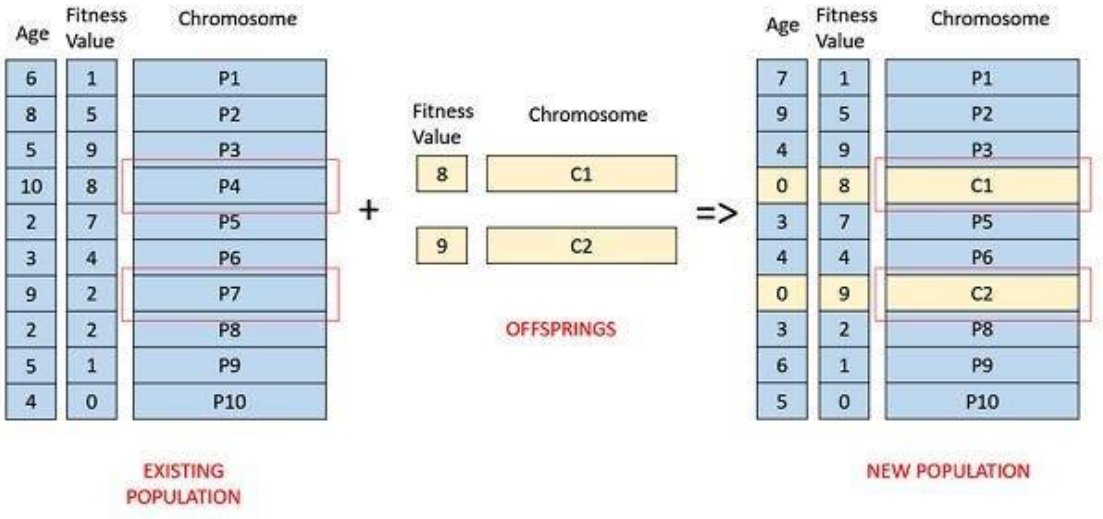


Рис. 3.19. Відбір за віком

Для реалізації другого методу існує два варіанти.

Перший із яких - елітізм. Після обчислення ФП виконаємо сортування особин в порядку її зростання; особина, яка має найменше значення сумарної довжини шляху, є найбільш придатною. Обираємо кількість особин для наступного покоління та видаляємо решту з них [29].

У випадку його використання збільшується продуктивність генетичного алгоритму, оскільки в кожне наступне покоління завжди переходять найкращі з минулого. Але все-таки існує недолік, оскільки гени елітних особин можуть розповсюджуватися в популяції і це може призвести до втрати різноманітності.

Другий варіант – відбір за ФП. В цьому випадку можлива заміна особин з найменшими ФП більш придатними. Приведемо приклад, в якому найбільш придатні особини, а саме *C*1 та *C*2, замінюють найменш придатні *P*1 та *P*10 (рис. 3.20). У випадку, коли у найменш придатних особин однакова ФП, вибір для заміни виконується довільно [29].

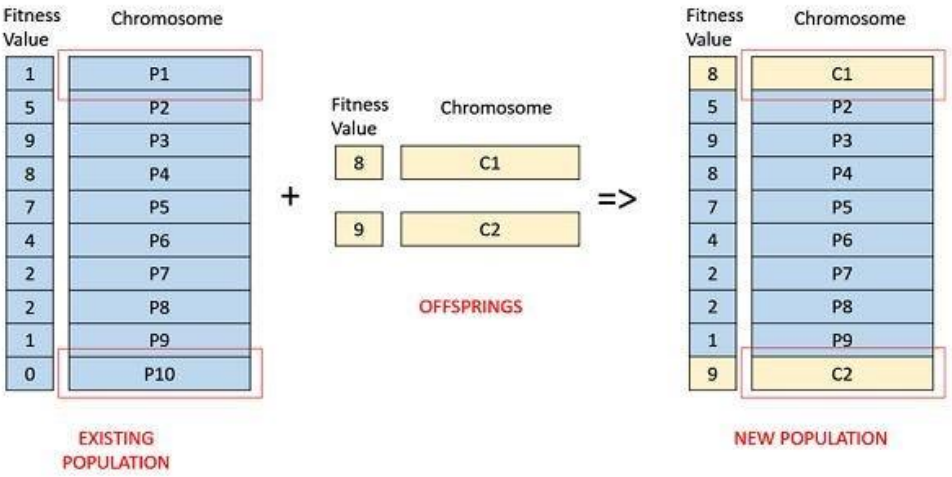


Рис. 3.20. Відбір на основі ФП

На останньому кроці алгоритму відбувається перевірка умови завершення алгоритму. На початку виконання алгоритму можна задати кількість поколінь або ж значення функції приналежності, яка вважається задовільною для вирішення поставленої задачі, при яких буде відбуватися завершення роботи генетичного алгоритму.

Також, як критерій умови завершення, можна обрати повторюваність значення функції принадлежності в *n*-й кількості останніх поколінь.

Висновки за розділом

В третьому розділі було розглянуто основі інструменти для вирішення поставленої задачі, серед яких: точні методи, евристичні та метаевристичні. Серед них на основі висунутих критеріїв, а саме розміру набору вхідних даних, часу на виконання, точності результату та доступної продуктивності, було обрано генетичний алгоритм. Після чого було проведено дослідження основних варіантів проведення відбору, схрещування та мутації. Серед них для подальшого використання було обрано: турнірний відбір, впорядковане схрещування та мутацію обміном.

РОЗДІЛ 4
ДОСЛІДЖЕННЯ РОЗРОБЛЕНОЇ МОДЕЛІ НА УМОВНОМУ ПРИКЛАДІ НА ПРЕДМЕТ ВИЗНАЧЕННЯ ЕФЕКТИВНОЇ ТОПОЛОГІЇ ЛКМ ДЛЯ НАЗЕМНОГО СЕГМЕНТУ СИСТЕМИ БЕЗПІЛОТНИХ АВІАЦІЙНИХ АПАРАТІВ

4.1. Розробка структурної схеми генетичного алгоритму

На рис. 4.1 представлена блок-схема генетичного алгоритму. Для його виконання необхідно пройти наступні етапи: створення початкової популяції особин; схрещування; мутація; селекція; перевірка умови припинення алгоритму. В тому випадку, коли умова не виконується, переходимо до кроку 2, в іншому - визначаємо кращу особину, тобто найбільш оптимальний маршрут.

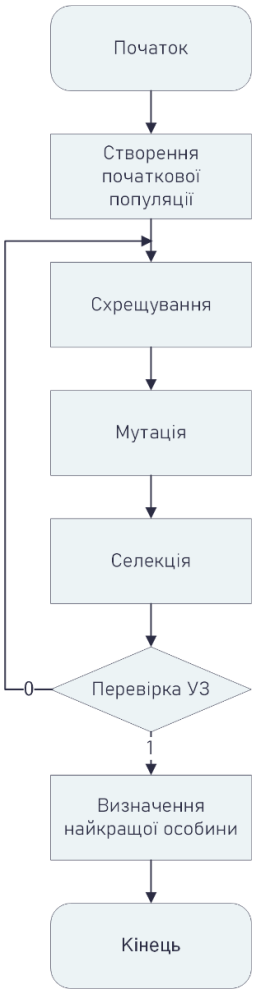


Рис. 4.1. Блок-схема генетичного алгоритму

4.2. Розробка комп'ютерної програми реалізації структурної схеми генетичного алгоритму

В коді використано два класи:

- *Vector* призначений для зберігання координат довготи, ширини та висоти на рівнем моря елементів ЛОМ наземного сегменту БАК (рис. 4.2);

```
class Vector:
    def __init__(self, x, y, z):
        self.x = x
        self.y = y
        self.z = z
```

Рис. 4.2. Клас *Vector*

- *Individual* – для представлення послідовності елементів та функції приналежності особини (рис. 4.3).

```
class Individual:
    def __init__(self, path):
        self.path = path
        self.fitness = None
```

Рис. 4.3. Клас *Individual*

та наступні функції:

- *calculate_distance* – для обчислення евклідової відстані між точками;
- *generate_random_path* – для генерації випадкового шляху;
- *calculate_fitness* – для обчислення функції приналежності особини;
- *generate_initial_population* – для генерації початкової популяції;
- *tournament_selection* – для відбору батьків за турнірним методом;
- *crossbreeding* – для двохточкового схрещування хромосом двух батьків;
- *mutation* – для мутації особин з певною заданою ймовірністю;
- *select_next_generation* – для відбору особин для наступного покоління.

Розглянемо детальніше ці функції.

Функція *calculate_distance* приймає в себе два параметри *point*1 та *point*2, які представляють собою два об'єкти класу *Vector*, тобто точки в трьохвимірному

просторі. У функції проводиться обчислення квадратного кореню суми квадратів різниці між відповідними координатами точок (рис. 4.4).

```
def calculate_distance(point1, point2):
    dx = point2.x - point1.x
    dy = point2.y - point1.y
    dz = point2.z - point1.z
    return math.sqrt(dx**2 + dy**2 + dz**2)
```

Рис. 4.4. Функція *calculate_distance*

Функція *generate_random_path* приймає в себе параметр *num_point*, який відповідає за кількість точок. В ній створюється список *path*, який містить послідовність чисел від 1 до *num_point*-1. З використанням функції *random.shuffle*() виконується перемішування елементів цього списку. Функція повертає новий список, в якому початкова та кінцева точки завжди одинакові, тобто рівні 0 (рис. 4.5).

```
def generate_random_path(num_points):
    path = list(range(1, num_points))
    random.shuffle(path)
    return [0] + path + [0]
```

Рис. 4.5. Функція *generate_random_path*

Функція *calculate_fitness* приймає два параметри, *individual* та *distance_matrix*. Спочатку створюється змінна *total_distance*, яка буде зберігати сумарну відстань шляху, та присвоюється їй значення 0. Після цього виконується цикл, що перебирає всі точки послідовності. Всередині циклу виконується обчислення сумарної відстані шляхом додавання відстаней між поточною *individual.path*[*i*] та наступною *individual.path*[*i*+1] точками (рис. 4.6). Ці значення беруться з матриці відстаней *distance_matrix*, в якій елемент *i*,*j* представляє відстань між точками *i* та *j*.

```
def calculate_fitness(individual, distance_matrix):
    total_distance = 0
    for i in range(len(individual.path) - 1):
        total_distance += distance_matrix[individual.path[i]][individual.path[i+1]]
    return total_distance
```

Рис. 4.6. Функція *calculate_fitness*

Функція *generate_initial_population* приймає два параметри, *num_individuals* та *num_points*. В ній задається початкова популяція шляхом виклику функції *generate_random_path* та створення екземпляру об'єкту *individual*, який додається в список *population* з використанням методу *append num_individuals* разів (рис. 4.7).

```
def generate_initial_population(num_individuals, num_points):
    population = []
    for _ in range(num_individuals):
        path = generate_random_path(num_points)
        individual = Individual(path)
        population.append(individual)
    return population
```

Рис. 4.7. Функція *generate_initial_population*

Функція *tournament_selection* приймає два параметри, *population* та *tournament_size*. В ній виконується цикл, в якому випадковим чином обираються особини в кількості *tournament_size* з використанням функції *random.sample*(). Таким чином формується група учасників турніру. Після цього обирається переможець з найменшим значення сумарної відстані *fitness*, який додається до списку *selected_parents* (рис. 4.8).

```
def tournament_selection(population, tournament_size):
    selected_parents = []
    for _ in range(len(population)):
        tournament_contestants = random.sample(population, tournament_size)
        winner = min(tournament_contestants, key=lambda x: x.fitness)
        selected_parents.append(winner)
    return selected_parents
```

Рис. 4.8. Функція *tournament_selection*

Функція *crossbreeding* приймає два параметри, *parent*1 та *parent*2. Спочатку прибираємо початковий та кінцевий елементи *parent*1 та *parent*2, оскільки немає необхідності їх задіювати у схрещуванні.Створюються списки: *child*1 та *child*2, які будуть зберігати хромосоми нащадків; *child*1*P*1, *child*1*P*2, *child*2*P*1, *child*2*P*2 – для зберіганні унаслідованих генів кожного з батьків.

Випадковим чином обираються дві точки розриву, які записуються в *pointA* та *pointB*, та виконується перевірка рівності обраних точок. У випаду їх ідентичності проводиться повторна генерація однієї з них. Для вибору початкової та кінцевої точки

використовуються функції *min* та *max* відповідно, значення записуються в змінні *startPoint* та *endPoint* (рис. 4.9).

```
def crossbreeding(parent1, parent2):
    parent1 = parent1[1:-1]
    parent2 = parent2[1:-1]
    child1 = []
    child2 = []
    child1P1 = []
    child1P2 = []
    child2P1 = []
    child2P2 = []

    pointA = int(random.random() * (len(parent1)-1))
    pointB = int(random.random() * (len(parent1)-1))
```

Рис. 4.9. Функція *crossbreeding*

Після цього батьківські гени між *startPoint* та *endPoint* дублюються в дві змінні*: child1P1* та *child2P1*. Батьківські гени, які не включені в *child1P1* та *child2P1*, записуються відповідно в змінні *child1P2* та *child2P2*.

Потім, в залежності від значення *startPoint*, проводиться формування хромосом нащадків (рис. 4.10 – 4.11).

Якщо ця змінна:

- рівна 0, тоді відбувається контатенація двох *змінних child1P1 + child1P2* та *child2P1 + child2P2*;

- рівна 1 – в перший ген записується останній елемент *child*1P2 для першого нащадка та *child1P2* – для другого, а в гени після кінцевої точки розриву заповнюються елементами *child1P2* та *child2P2*. в діапазоні індексів від 0 до довжини *child1P2* – 1. Між ними відповідно розміщуються змінні *child1P1* та *child2P1;*

- в іншому випадку – після кінцевої точки розриву розміщуються елементи *child1P2* та *child2P2* в діапазоні від 0 до *endInter*, решта елементів цих векторів записуються в гени до початкової точки розриву. Між ними відповідно розміщуються змінні *child1P1* та *child2P1*.

Тепер додаємо до сформованих векторів хромосом нащадків нулі в початок та кінець з допомогою функції *insert* та *append*.

```
    while pointA == pointB:
        pointB = int(random.random() * (len(parent1)-1))

    startPoint = min(pointA, pointB)
    endPoint = max(pointA, pointB)
    print(startPoint, endPoint)

    for i in range(startPoint, endPoint+1):
        child1P1.append(parent1[i])
        child2P1.append(parent2[i])

    child1P2 = [item for item in parent2 if item not in child1P1]
    child2P2 = [item for item in parent1 if item not in child2P1]

    endInter = len(parent1)-endPoint
```

Рис. 4.10. Функція *crossbreeding*

```
if startPoint == 0:
    child1 = child1P1 + child1P2
    child2 = child2P1 + child2P2
elif startPoint == 1:
    child1 = [child1P2[len(child1P2)-1]] + child1P1 + child1P2[:(len(child1P2)-1)]
    child2 = [child2P2[len(child1P2)-1]] + child2P1 + child2P2[:(len(child1P2)-1)]
else:
    child1 = child1P2[endInter:len(child1P2)] + child1P1 + child1P2[:endInter]
    child2 = child2P2[endInter:len(child1P2)] + child2P1 + child2P2[:endInter]

child1.insert(0, 0)
child1.append(0)
child2.insert(0, 0)
child2.append(0)

return Individual(child1), Individual(child2)
```

Рис. 4.11. Функція *crossbreeding*

Переходимо до функції *mutation*, яка приймає два параметри *individual* та *mutation_rate*. Виконується генерація випадкового числа в діапазоні від 0 до 1, якщо це значення менше заданої ймовірності, то мутація проводиться. Тепер обираються дві точки, які будуть представляти початок і кінець ділянки генів, що будуть змінюватися. З генів *individual.path* особини обирається ділянка між *startGene* і *endGene*, і ця ділянка інвертується за допомогою [::-1] (рис. 4.12). Інвертовані значення *values_to_reverse* замінюють відповідну ділянку в генах *individual.path*.

```
def mutation(individual, mutation_rate):
    if random.random() < mutation_rate:
        geneA = random.randint(1, len(individual.path) - 1)
        geneB = random.randint(1, len(individual.path) - 1)

        startGene = min(geneA, geneB)
        endGene = max(geneA, geneB)

        values_to_reverse = individual.path[startGene:endGene][::-1]
        individual.path[startGene:endGene] = values_to_reverse
```

Рис. 4.12. Функція *mutation*

Функція *select_next_generation* приймає два аргументи: *population*, що представляє поточну популяцію особин, і *num_individuals*, який визначає, скільки особин потрібно вибрати в наступне покоління. Виконується сортування значення функції приналежності у порядку зростання (значення *reverse=False*). Після цього обирається перша половина відсортованої популяції з найкращими значеннями ФП, яка формує наступне покоління (рис. 4.13).

```
def select_next_generation(population, num_individuals):
    population.sort(key=lambda x: x.fitness, reverse=False)
    next_generation = population[:num_individuals//2]
    return next_generation
```

Рис. 4.13. Функція *select_next_generation*

Останньою буде функція *main*.

Запис початкових даних. Користувачу пропонується ввести кількість точок *n* та їх координати *x*, *y* та *z*, які записуються в вектор *points* (рис. 4.14).

```
n = int(input("Введіть кількість точок: "))
points = []

if n % 2 == 0:
    num_individuals = n
else:
    num_individuals = n + 1

for i in range(n):
    x = float(input(f"Введіть x-координату для точки {i+1}: "))
    y = float(input(f"Введіть y-координату для точки {i+1}: "))
    z = float(input(f"Введіть z-координату для точки {i+1}: "))
    points.append(Vector(x,y,z))
```

Рис. 4.14. Запис вхідних даних

Формування матриці відстаней. Діагональ матриці *distance_matrix* заповнюється значеннями нескінченності. Якщо *i* і *j* не рівні, то викликається функція *calculate_distance*, яка обчислює відстань між двома точками *points*[*i*] і *points*[*j*]. Результат цього обчислення присвоюється елементу *distance_matrix*[*i*, *j*] (рис. 4.15).

```
distance_matrix = np.full((n, n), float('inf'))

for i in range(n):
    for j in range(n):
        if i != j:
            distance_matrix[i, j] = calculate_distance(points[i], points[j])

print("\nМатриця відстаней:")
for i in range(n):
    for j in range(n):
        print(f"{distance_matrix[i, j]:.3f}\t", end="")
    print()
```

Рис. 4.15. Формування матриці відстаней

Генерація початкової популяції. Відбувається виклик функції *generate_initial_population*, в яку передаються: змінна *num_individuals*, яка позначає кількість особин у популяції, *n* - кількість точок. У результаті виконання цього рядка коду, *initial_population* стає списком особин з їхніми генами.

Для кожної особини викликається функція *calculate_fitness*, яка обчислює функції приналежності на основі її генів і матриці відстаней *distance_matrix* (рис. 4.16). Отримані значення присвоюється *individual* у вигляді атрибута *fitness*.

```
initial_population = generate_initial_population(num_individuals, n)

for individual in initial_population:
    individual.fitness = calculate_fitness(individual, distance_matrix)

print("\nПочаткова популяція:")
for i, individual in enumerate(initial_population):
    print(f"Особина {i+1}: {individual.path}, ФП: {individual.fitness}")
```

Рис. 4.16. Генерація початкової популяції

Після цього створюються три змінних:

- *num_generations* – кількість поколінь;

- *min_fitness_values* – мінімальне значення функції приналежності;

- *current_generation* – поточне покоління, якому перед виконанням генетичного алгоритму присвоюємо початкову популяцію *initial_population*.

Переходимо до циклу, який буде виконуватися, доки не буде виконана умова завершення. На кожному проході циклу збільшується значення лічильника поколінь *num_generations* на 1.

Наступним кроком встановлюємо розмір турнірному відбору батьків *tournament_size* та записуємо результат виконання функції *tournament_selection* в змінну *selected_parents* (рис. 4.17).

```
tournament_size = 2
selected_parents = tournament_selection(current_generation, tournament_size)

print("\nОбрані батьки з використанням турнірного відбору:")
for i, parent in enumerate(selected_parents):
    print(f"Батько {i+1}: {parent.path}, ФП: {parent.fitness}")
```

Рис. 4.17. Запис результату турнірного відбору у змінну *selected_parents*

Тепер створюємо список для зберігання нащадків *children* та переходимо до виконання циклу, який ітерується від 0 до довжини змінної *selected_parents* з кроком *2*. В ньому виконується вибір двох батьків з цієї змінної та виклик функції *crossbreeding*, результати якої записуються в змінні *child*1 та *child*2 (рис. 4.18). Отримані нащадки записуються в список *children з* використанням функції *append.*

```
children = []
for i in range(0, len(selected_parents), 2):
    parent1 = selected_parents[i]
    parent2 = selected_parents[i+1]
    child1, child2 = crossbreeding(parent1.path, parent2.path)
    children.append(child1)
    children.append(child2)

print("\nЗгенеровані нащадки:")
for i, child in enumerate(children):
    print(f"Нащадок {i+1}: {child.path}, ФП: {calculate_fitness(child, distance_matrix)}")
```

Рис. 4.18. Схрещування батьківських хромосом

Наступним кроком задаємо ймовірність мутації для кожної особини в змінну *mutation_rate*. Після цього виконується цикл, який перебирає всіх особин з поточного покоління, в яку входять як батьки, так і нащадки (рис. 4.19), та проводить мутацію окремих особин.

```
mutation_rate = 1.0 / len(selected_parents)
current_generation.extend(children)
for individual in current_generation:
    mutated_before = list(individual.path)
    mutation(individual, mutation_rate)
    if individual.path != mutated_before:
        print(f"\nМутація особини: {individual.path}")
```

Рис. 4.19. Мутація особин

Тепер виконується відбір особин для наступного покоління з використанням функції *select_next_generation* (рис. 4.20).

```
next_generation = select_next_generation(current_generation, len(current_generation))
print("\nНаступне покоління:")
for i, individual in enumerate(next_generation):
    print(f"Особина {i+1}: {individual.path}, ФП: {individual.fitness}")
```

Рис. 4.20. Відбір особин для наступного покоління

Для перевірки умови завершення знаходимо мінімальне значення ФП *min_fitness* серед усіх особин сформованого покоління *next_generation*, яке записуємо в список *min_fitness_values*. Умова завершення – останні 5 елементів списку *min_fitness_values* мають однакове значення. Якщо умова виконується, то цикл *while* завершується (рис. 4.21). Значення *next_generation* присвоюється *current_generation*.

```
min_fitness = min([ind.fitness for ind in next_generation])
min_fitness_values.append(min_fitness)
if len(min_fitness_values) >= 5 and all(value == min_fitness
                                        for value in min_fitness_values[-5:]):
    print(f"\nЗавершення алгоритму після {num_generations} поколінь.")
    break
current_generation = next_generation
```

Рис. 4.21. Перевірка умови завершення ГА

Після завершення циклу виводиться перший елемент *current_generation*, який має найменше значення ФП та є найкращим рішенням поставленої задачі (рис. 4.22).

```
print("\nФінальне рішення:")
first_individual = current_generation[0]
print(f"Шлях: {individual.path}, довжина шляху: {individual.fitness}")
```

Рис. 4.22. Виведення першого елемент*у current_generation*

4.3. Розрахунок топології ЛКМ стандарту *IEEE* 802.5 для наземного сегменту системи безпілотних авіаційних апаратів

Компоненти ЛКМ представимо у вигляді системи трьох координат *x* (широти), *y* (довготи), *z* (висоти над рівнем моря) (табл. 4.1). Для обчислення висоти було використано сайт *https://www.advancedconverter.com/map-tools/find-altitude-by-coordinates*, в якому для отримання значення висоти необхідно вказати широту та довготу, після чого натиснути на кнопку «*Search*».

Наступним кроком вносимо значення координат в програму та запускаємо її. Перед виконанням основної частини, тобто генетичного алгоритму, виконується переведення широти та довготи з радіан в метри (табл. 4.2).

Таблиця 4.1

Координати точок

№ точки	Широта, радіан	Довгота, радіан	Висота, м
1	50.54331	30.23587	116
2	50.54431	30.23925	116
3	50.54507	30.23254	117
4	50.54849	30.23998	122
5	50.54621	30.23755	117
6	50.54906	30.23564	125
7	50.54806	30.24134	126
8	50.55198	30.23844	129

Таблиця 4.2

Координати точок

№ точки	Широта, м	Довгота, м	Висота, м
1	5601280.19	2128294.09	116
2	5601392.98	2128482.42	116
3	5601477.17	2127982.75	117
4	5601858.08	2128343.55	122
5	5601602.77	2128279.24	117
6	5601921.53	2128015.92	125
7	5601810.13	2128457.22	126
8	5602246.19	2128077.89	129

Отримаємо матрицю відстаней представлену в табл. 4.3.

Таблиця 4.3

Матриця відстаней

	1	2	3	4	5	6	7	8
1	*inf*	219.526	368.422	580.033	322.927	699.126	554.574	989.989
2	219.526	*inf*	506.713	485.420	292.056	705.025	418.029	944.341

Продовження таблиці 4.3

	1	2	3	4	5	6	7	8
3	368.422	506.713	*inf*	524.680	321.989	445.666	579.709	774.977
4	580.033	485.420	524.680	*inf*	263.330	333.728	123.431	470.381
5	322.927	292.056	321.989	263.330	*inf*	413.525	273.419	674.296
6	699.126	705.025	445.666	333.728	413.525	*inf*	455.139	330.549
7	554.574	418.029	579.709	123.431	273.419	455.139	*inf*	577.969
8	989.989	944.341	774.977	470.381	674.296	330.549	577.969	*inf*

Для наглядності побудуємо граф, де вершини графу – компоненти ЛКМ наземного сегменту БАК, а дуги – канали зв’язку між ними (рис. 4.23).

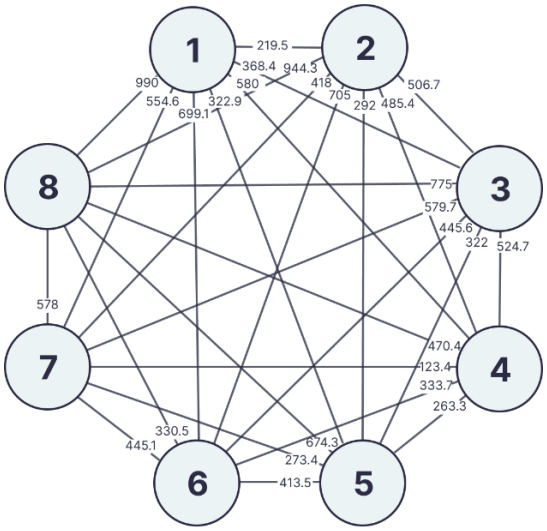


Рис. 4.23. Граф мережі

Відобразимо точки, створивши інтерактивну карту з використанням *Google Maps*:

- Для цього переходимо на сайт *https://www.google.com/maps/d/* та натискаємо на кнопку «Створити нову карту»;

- У лівому верхньому куті натиснемо на кнопку «Імпорт»;

- Обираємо *Excel*-файл з координатами;

- Обираємо стовпці з координатами широти та довготи. Натискаємо на «Продовжити»;

- Обираємо стовпець з заголовками маркерів та натискаємо «Готово».

Створена інтерактивна карта представлена на рис. 4.24.

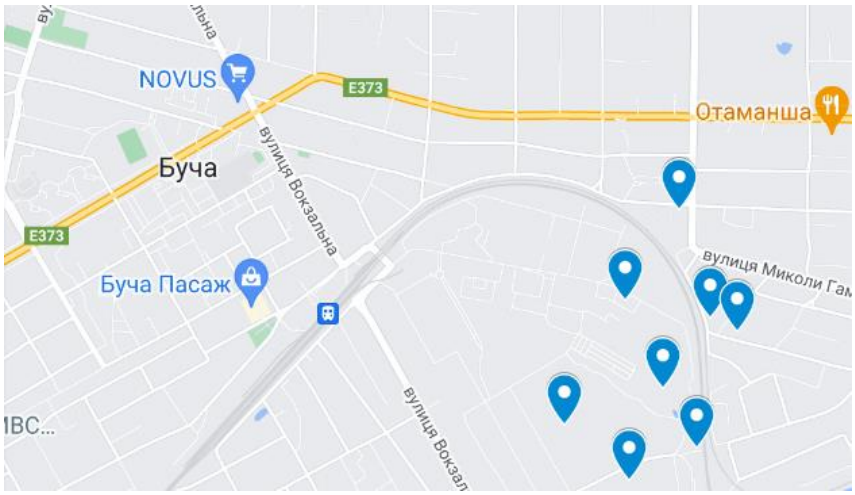


Рис. 4.24. Інтерактивна карта

Початкова популяція включає в себе вісім особин з наступними функціями приналежності (табл. 4.4).

Таблиця 4.4

Початкова популяція

№ особини	Шлях	Функція приналежності
1	0, 3, 1, 5, 4, 7, 2, 6, 0	4767.55941
2	0, 1, 3, 7, 6, 5, 2, 4, 0	3299.01615
3	0, 2, 4, 6, 3, 5, 1, 7, 0	4060.34438
4	0, 3, 7, 6, 1, 5, 2, 4, 0	3842.019034
5	0, 6, 1, 2, 4, 5, 3, 7, 0	4008.92754
6	0, 7, 5, 2, 3, 1, 4, 6, 0	3896.35228
7	0, 3, 6, 7, 4, 5, 2, 1, 0	3541.15767
8	0, 4, 3, 5, 1, 6, 7, 2, 0	3764.40694

Для першого покоління за турнірним відбором були обрані наступні батьки (табл. 4.5).

Таблиця 4.5

Вибрані батьки

№ батьківської особини	Шлях	Функція приналежності
1	0, 3, 6, 7, 4, 5, 2, 1, 0	3541.15767
2	0, 6, 1, 2, 4, 5, 3, 7, 0	4008.92754
3	0, 6, 1, 2, 4, 5, 3, 7, 0	4008.92754
4	0, 7, 5, 2, 3, 1, 4, 6, 0	3896.35228
5	0, 3, 7, 6, 1, 5, 2, 4, 0	3842.01903
6	0, 3, 7, 6, 1, 5, 2, 4, 0	3842.01903
7	0, 1, 3, 7, 6, 5, 2, 4, 0	3299.01615
8	0, 2, 4, 6, 3, 5, 1, 7, 0	4060.34438

Точки розрізу для схрещування:

- для першої пари – 1 та 3;

- для другої пари – 0 та 1;

- для третьої пари – 0 та 3;

- для четвертої пари – 0 та 5.

Після чого були сформовані нащадки шляхом впорядкованого схрещування (табл. 4.6).

Таблиця 4.6

Сформовані нащадки

№ нащадка	Шлях	Функція приналежності
1	2	3
1	0, 3, 6, 7, 4, 1, 2, 5, 0	3899.28799
2	0, 5, 1, 2, 4, 3, 6, 7, 0	4187.57053
3	0, 6, 1, 7, 5, 2, 3, 4, 0	3804.09531
4	0, 7, 5, 6, 1, 2, 4, 3, 0	3865.76964
5	0, 3, 7, 6, 1, 5, 2, 4, 0	3842.01903
6	0, 3, 7, 6, 1, 5, 2, 4, 0	3842.01903

Продовження таблиці 4.6

1	2	3
7	0, 1, 3, 7, 6, 5, 2, 4, 0	3299.01615
8	0, 2, 4, 6, 3, 5, 1, 7, 0	4060.34438

В результаті отримаємо наступну популяцію (табл. 4.7).

Таблиця 4.7

Отримана популяція

№ особини	Шлях	Функція приналежності
1	0, 3, 1, 5, 4, 7, 2, 6, 0	4767.55941
2	0, 1, 3, 7, 6, 5, 2, 4, 0	3299.01615
3	0, 2, 4, 6, 3, 5, 1, 7, 0	4060.34438
4	0, 3, 7, 6, 1, 5, 2, 4, 0	3842.01903
5	0, 6, 1, 2, 4, 5, 3, 7, 0	4008.92754
6	0, 7, 5, 2, 3, 1, 4, 6, 0	3896.35228
7	0, 3, 6, 7, 4, 5, 2, 1, 0	3541.15767
8	0, 4, 3, 5, 1, 6, 7, 2, 0	3764.40694
9	0, 3, 6, 7, 4, 1, 2, 5, 0	3899.28799
10	0, 5, 1, 2, 4, 3, 6, 7, 0	4187.57053
11	0, 6, 1, 7, 5, 2, 3, 4, 0	3804.09531
12	0, 7, 5, 6, 1, 2, 4, 3, 0	3865.76964
13	0, 3, 7, 6, 1, 5, 2, 4, 0	3842.01903
14	0, 3, 7, 6, 1, 5, 2, 4, 0	3842.01903
15	0, 1, 3, 7, 6, 5, 2, 4, 0	3299.01615
16	0, 2, 4, 6, 3, 5, 1, 7, 0	4060.34438

Проведемо сортування особин в порядку збільшення функції приналежності, в результаті чого до наступного покоління перейдуть вісім особин з найкращими значеннями функції приналежності (табл. 4.8).

Таблиця 4.8

Обрані в наступне покоління особини

№ особини	Шлях	Функція приналежності
1	0, 1, 3, 7, 6, 5, 2, 4, 0	3299.01615
2	0, 1, 3, 7, 6, 5, 2, 4, 0	3299.01615
3	0, 3, 6, 7, 4, 5, 2, 1, 0	3541.157675
4	0, 4, 3, 5, 1, 6, 7, 2, 0	3764.40694
5	0, 6, 1, 7, 5, 2, 3, 4, 0	3804.09531
6	0, 3, 7, 6, 1, 5, 2, 4, 0	3842.01903
7	0, 3, 7, 6, 1, 5, 2, 4, 0	3842.01903
8	0, 3, 7, 6, 1, 5, 2, 4, 0	3842.01903

Розглянемо наступне з поколінь, в якому з’явився новий кандидат на найкраще вирішення задачі.

В ньому були обрані наступні батьки (табл. 4.9)

Таблиця 4.9

Обрані батьки

№ особини	Шлях	Функція приналежності
1	0, 4, 3, 5, 1, 6, 7, 2, 0	3764.40694
2	0, 1, 3, 7, 6, 5, 2, 4, 0	3299.01615
3	0, 4, 3, 5, 1, 6, 7, 2, 0	3764.40694
4	0, 1, 3, 7, 6, 5, 2, 4, 0	3299.01615
5	0, 1, 3, 7, 6, 5, 2, 4, 0	3299.01615
6	0, 1, 3, 7, 6, 5, 2, 4, 0	3299.01615
7	0, 1, 3, 7, 6, 5, 2, 4, 0	3299.01615
8	0, 1, 3, 7, 6, 5, 2, 4, 0	3299.01615

Точки розрізу для схрещування:

- для першої пари – 0 та 5;

- для другої пари – 1 та 2;

- для третьої пари – 3 та 4;

- для четвертої пари – 1 та 3.

Після чого сформовані нащадки шляхом впорядкованого схрещування (табл. 4.10).

Таблиця 4.10

Сформовані нащадки

№ особини	Шлях	Функція приналежності
1	0, 4, 3, 5, 1, 6, 7, 2, 0	3764.40694
2	0, 1, 3, 7, 6, 5, 2, 4, 0	3299.01615
3	0, 4, 3, 5, 1, 7, 6, 2, 0	4095.45098
4	0, 2, 3, 7, 4, 5, 1, 6, 0	4128.93301
5	0, 7, 2, 4, 6, 5, 1, 3, 0	4585.99118
6	0, 7, 2, 4, 6, 5, 1, 3, 0	4585.99118
7	0, 4, 3, 7, 6, 1, 5, 2, 0	3571.74891
8	0, 4, 3, 7, 6, 1, 5, 2, 0	3571.74891

Проведено мутацію:

Мутація особини: [0, 2, 5, 7, 1, 6, 3, 4, 0]

Мутація особини: [0, 4, 3, 2, 6, 7, 1, 5, 0]

Сформована популяція має наступний вигляд (табл. 4.11).

Таблиця 4.11

Отримана популяція

№ особини	Шлях	Функція приналежності
1	2	3
1	0, 1, 3, 7, 6, 5, 2, 4, 0	3299.01615
2	0, 1, 3, 7, 6, 5, 2, 4, 0	3299.01615
3	0, 3, 6, 7, 4, 5, 2, 1, 0	3541.15767
4	0, 4, 3, 5, 1, 6, 7, 2, 0	3764.40694

Продовження таблиці 4.11

1	2	3
5	0, 2, 5, 7, 1, 6, 3, 4, 0	3216.69414
6	0, 3, 7, 6, 1, 5, 2, 4, 0	3842.01903
7	0, 3, 7, 6, 1, 5, 2, 4, 0	3842.01903
8	0, 3, 7, 6, 1, 5, 2, 4, 0	3842.01903
9	0, 4, 3, 5, 1, 6, 7, 2, 0	3764.40694
10	0, 1, 3, 7, 6, 5, 2, 4, 0	3299.01615
11	0, 4, 3, 2, 6, 7, 1, 5, 0	4617.10633
12	0, 2, 3, 7, 4, 5, 1, 6, 0	4128.93301
13	0, 7, 2, 4, 6, 5, 1, 3, 0	4585.99118
14	0, 7, 2, 4, 6, 5, 1, 3, 0	4585.99118
15	0, 4, 3, 7, 6, 1, 5, 2, 0	3571.74891
16	0, 4, 3, 7, 6, 1, 5, 2, 0	3571.74891

Виконується сортування особин в порядку зростання їх функції приналежності і в результаті до наступного покоління переходять наступні (табл. 4.12)

Таблиця 4.12

Особини, які переходять до наступного покоління

№ особини	Шлях	Функція приналежності
1	0, 2, 5, 7, 1, 6, 3, 4, 0	3216.69414
2	0, 1, 3, 7, 6, 5, 2, 4, 0	3299.01615
3	0, 1, 3, 7, 6, 5, 2, 4, 0	3299.01615
4	0, 1, 3, 7, 6, 5, 2, 4, 0	3299.01615
5	0, 3, 6, 7, 4, 5, 2, 1, 0	3541.15767
6	0, 4, 3, 7, 6, 1, 5, 2, 0	3571.74891
7	0, 4, 3, 7, 6, 1, 5, 2, 0	3571.74891
8	0, 4, 3, 5, 1, 6, 7, 2, 0	3764.40694

В наступному поколінні з’явився ще більш кращий кандидат.

В цьому поколінні були обрані наступні батьки (табл. 4.13).

Таблиця 4.13

Обрані батьки

№ особини	Шлях	Функція приналежності
1	0, 1, 3, 7, 6, 5, 2, 4, 0	3299.01615
2	0, 1, 3, 7, 6, 5, 2, 4, 0	3299.01615
3	0, 2, 5, 7, 1, 6, 3, 4, 0	3216.69414
4	0, 4, 3, 7, 6, 1, 5, 2, 0	3571.74891
5	0, 4, 3, 7, 6, 1, 5, 2, 0	3571.74891
6	0, 1, 3, 7, 6, 5, 2, 4, 0	3299.01615
7	0, 2, 5, 7, 1, 6, 3, 4, 0	3216.69414
8	0, 3, 6, 7, 4, 5, 2, 1, 0	3541.15767

Точки розрізу для схрещування:

- для першої пари – 2 та 3;

- для другої пари – 1 та 5;

- для третьої пари – 1 та 3;

- для четвертої пари – 0 та 2.

Проведомо формування нащадків шляхом впорядкованого схрещування (табл. 4.14).

Таблиця 4.14

Сформовані нащадки

№ особини	Шлях	Функція приналежності
1	2	3
1	0, 2, 4, 7, 6, 1, 3, 5, 0	3878.97858
2	0, 2, 4, 7, 6, 1, 3, 5, 0	3878.97858
3	0, 2, 5, 7, 1, 6, 3, 4, 0	3216.69414
4	0, 4, 3, 7, 6, 1, 5, 2, 0	3571.74891

Продовження таблиці 4.14

1	2	3
5	0, 4, 3, 7, 6, 1, 5, 2, 0	3571.74891
6	0, 2, 3, 7, 6, 4, 1, 5, 0	3911.07795
7	0, 2, 5, 7, 3, 6, 4, 1, 0	2523.44895
8	0, 3, 6, 7, 2, 5, 1, 4, 0	3822.08324

А також мутацію:

Мутація особини: [0, 2, 5, 7, 1, 3, 6, 4, 0]

Мутація особини: [0, 1, 3, 7, 5, 6, 2, 4, 0]

Мутація особини: [0, 4, 6, 7, 3, 1, 5, 2, 0]

В результаті сформувалася наступна популяція (табл. 4.15).

Таблиця 4.15

Отримана популяція

№ особини	Шлях	Функція приналежності
1	2	3
1	0, 2, 5, 7, 1, 3, 6, 4, 0	3294.17468
2	0, 1, 3, 7, 6, 5, 2, 4, 0	3299.01615
3	0, 1, 3, 7, 6, 5, 2, 4, 0	3299.01615
4	0, 1, 3, 7, 5, 6, 2, 4, 0	3185.63909
5	0, 3, 6, 7, 4, 5, 2, 1, 0	3541.15767
6	0, 4, 3, 7, 6, 1, 5, 2, 0	3571.74891
7	0, 4, 3, 7, 6, 1, 5, 2, 0	3571.74891
8	0, 4, 3, 5, 1, 6, 7, 2, 0	3764.40694
9	0, 2, 4, 7, 6, 1, 3, 5, 0	3878.97858
10	0, 2, 4, 7, 6, 1, 3, 5, 0	3878.97858
11	0, 2, 5, 7, 1, 6, 3, 4, 0	3216.69414
12	0, 4, 6, 7, 3, 1, 5, 2, 0	3649.22944
13	0, 4, 3, 7, 6, 1, 5, 2, 0	3571.74891
14	0, 2, 3, 7, 6, 4, 1, 5, 0	3911.07795

Продовження таблиці 4.15

1	2	3
15	0, 2, 5, 7, 3, 6, 4, 1, 0	2523.44895
16	0, 3, 6, 7, 2, 5, 1, 4, 0	3822.08324

Серед якої до наступного покоління переходять такі особини (табл. 4.16)

Таблиця 4.16

Особини, які переходять до наступного покоління

№ особини	Шлях	Функція приналежності
1	0, 2, 5, 7, 3, 6, 4, 1, 0	2523.44895
2	0, 1, 3, 7, 5, 6, 2, 4, 0	3185.63909
3	0, 2, 5, 7, 1, 6, 3, 4, 0	3216.69414
4	0, 2, 5, 7, 1, 3, 6, 4, 0	3294.17468
5	0, 1, 3, 7, 6, 5, 2, 4, 0	3299.01615
6	0, 1, 3, 7, 6, 5, 2, 4, 0	3299.01615
7	0, 3, 6, 7, 4, 5, 2, 1, 0	3541.15767
8	0, 4, 3, 7, 6, 1, 5, 2, 0	3571.74891

Умовою зупинки роботи генетичного алгоритму була повторюваність значення функції приналежності протягом п'яти поколінь. Відповідно до того, що ця умова виконалась, можемо зробити висновок про те, що при заданих вхідних значеннях кожної з точок існує лише один варіант вирішення поставленої задачі, а саме послідовність елементів локальної комп’ютерної мережі стандарту *IEEE* 802.5, яка починається з першої точки, після цього з'єднує точки 3, 6, 8, 4, 7, 5, 2, та повертається в початкову. Загальна довжина кабелю при такому з'єднанні елементів рівна 2523.44895 м.

Для візуалізації отриманого результату роботи генетичного алгоритму представимо це з'єднання елементів ЛКМ на інтерактивній карті (рис. 4.25).

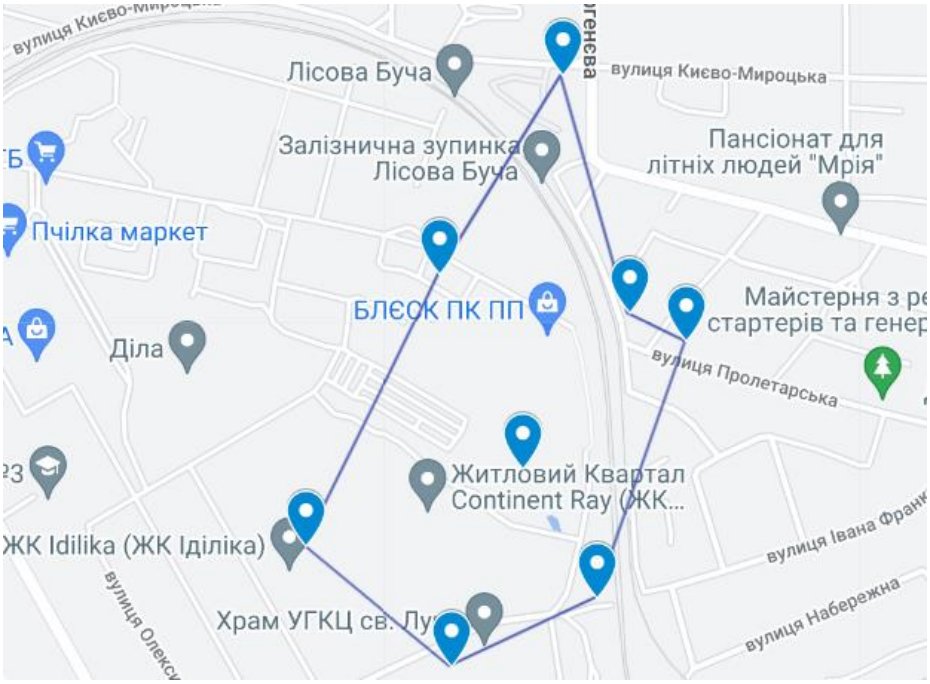


Рис. 4.25. Результат роботи генетичного алгоритму

Висновки за розділом

Четвертий розділ присвячений розробці програми на мові *Python* генетичного алгоритму для проєктування топології ЛКМ стандарту *IEEE* 802.5. Було попередньо створено блок-схему алгоритму, на основі якої було виконано написання програми. Для написання програми було використано чотири бібліотеки, а саме: *numpy*, *math*, *geopy* та *random*. Перед виконанням тестування програми було створено набір вхідних значень широти, довготи та висоти над рівнем моря для восьми точок. На їх основі було побудовано інтерактивну карту для наглядності отриманих від програми результатів.

ВИСНОВКИ

Перший розділ кваліфікаційної роботи присвячений системному аналізу ЛКМ стандарту *IEEE* 802.5, в якому розглядаються особливості, такі як використовувані топології, кількість станцій в сегменті, швидкості передачі даних, найбільш використувані специфікації фізичного рівня, принцип роботи *Token Ring*, механізми виявлення та компенсації помилок, а також типи кадрів. Додаково визначено критерії ефективності архітектурних рішень ЛКМ, а саме: продуктивність (час відгуку, пропускна здатність, максимальна пропускна здатність та затримка), надійність, безпека, відмовостійкість, розширюваність, масштабованість, керованість, прозорість, сумісність та вартість.

Другий розділ присвячений розробці набору моделей задачі проектування топологій ЛКМ стандарту *IEEE* 802.5. Комп'ютерна мережа була представлена у вигляді графу, де вершини графу – компоненти ЛКМ наземного сегменту БАК, а дуги – канали зв'язку між ними.

Перед розробкою набору було визначено критерії, такі як:

- кожна пара вершин повинна бути зв'язана одна з одною, тобто ступінь зв'язності повинен бути рівний двом;

- зроблено припущення про те, що збільшення сумарної довжини кабелю призводить до збільшення вартості капітальних витрат та випромінювання, що призводить до зменшення захищенності мережі.

На основі розробленого набору моделей (повний граф, дерево, Ейлерів граф та Гамільтонів граф) було проведено вибір базової моделі з використанням методу аналізу ієрархій. В результаті було обрано Гамільтонів граф як базову модель для дослідження, оскільки значення глобального пріоритету, а саме 0.439678, у цієї альтернативи на порядок вище, ніж у решти.

Третій розділ присвячено вибору інструментарію для дослідження розробленої моделі. Попередньо було розглянуто точні алгоритми (метод повного перебору, метод гілок та меж, метод Літтла, алгоритм Хелда-Карпа); евристичні алгоритми (метод найближчого сусіда, метод найбільш дешевого включення, дерев'яний

алгоритм); метаевристичні алгоритми (генетичний алгоритм, мурашиний алгоритм, алгоритм імітації відпалу).

На основі обраних параметрів, таких як розмір набору вхідних даних, час на виконання, точність результату та доступна продуктивність, серед усіх інструментів було обрано саме генетичний алгоритм, оскільки виникає необхідність у точності на великій кількості елементів у вхідному наборі при достаточньому часі та великій продуктивності.

Після цього було проведено більш детальний розгляд етапів генетичного алгоритму, а також способів їх виконання:

- відбору (відбір метод за правилом рулетки, стохастична універсальна вибірка, ранговий метод, масштабування придатності, турнірний відбір та випадковий відбір);

- схрещування (одноточкове, двохточкове та *k*-точкове, рівномірне, впорядковане, схрещування зміщенням та імітація двійкового зміщування);

- мутація (інвертування бітів, мутація обміном, мутація оберненням, мутація перетасуванням, мутація для дійсних чисел).

Серед доступних варіантів для подальшого використання було обрано: турнірний відбір, впорядковане схрещування та мутація оберненням.

Вибір кожного з методу обумовлений наступним:

- турнірний відбір легко реалізувати і він досить простий, оскільки не вимагає проводити додаткові обчислювання та/або проводити ражування по збільшенню ФП; зберігає різноманіття в популяції, оскільки навіть менш придатні особини можуть використовуватися в наступних кроках алгоритму;

- впорядковане схрещування дозволяє зберігати порядок, уникаючи недопустимих комбінацій, тобто рішень, де в одній послідовності повторюються одні і ті ж самі елементи ЛКМ;

- мутація оберненням зберігає загальну структуру шляху, що обходить через всі елементи ЛКМ, але змінює порядок їх відвідування, що може призвести до виявлення більш ефективних рішень.

Четвертий розділ присвячено розробці програмного коду на мові *Python* для реалізації генетичного алгоритму.

В коді використано:

-чотири бібліотеки:

- *math*:

- *math.sqrt* – для обчислення квадратного корення при розрахунку евклідової відстані;

- *numpy*:

- *np.full* - для створення матриці відстаней розміру *n* на *n* з початковими значеннями;

- *geopy:*

- *geopy.geodesic* - для обчислення геодезичної відстані між двома точками на земній поверхні;

- *random:*

- *random.shuffle* - для генерації випадкових шляхів для початкової популяції;

- *random.sample* - для випадкового вибору особин з популяції у методі турнірного відбору;

- *random.randint* - для генерації випадкових чисел для вибору двох випадкових позиції при схрещуванні та мутації;

- *random.random* - для генерації випадкових чисел, які вирішують, чи слід проводити мутацію.

- два класи:

- *Vector* призначений для зберігання координат довготи, ширини та висоти на рівнем моря елементів ЛКМ наземного сегменту БАК;

- *Individual* – для представлення послідовності елементів та функції приналежності особини

- наступні функції:

- *calculate_distance* – для обчислення евклідової відстані між точками;

- *generate_random_path* – для генерації випадкового шляху;

- *calculate_fitness* – для обчислення функції приналежності особини;

- *generate_initial_population* – для генерації початкової популяції;

- *tournament_selection* – для відбору батьків за турнірним методом;

- *crossbreeding* – для двохточкового схрещування хромосом двух батьків;

- *mutation* – для мутації особин з певною заданою ймовірністю;

- *select_next_generation* – для відбору особин для наступного покоління.

Для тестування створенної програми було створено вхідні дані восьми елементів ЛКМ стандарту *IEEE* 802.5 у вигляді системи трьох координат: широти, довготи та висоти над рівнем моря. При заданих вхідних значення існує один варіант вирішення задачі з наступною послідовністю з'єднання елементів ЛКМ стандарту *IEEE* 802.5: 1, 3, 4, 7, 8, 6, 5, 2, 1, та з загальною довжиною кабелю рівною 2523.45 м.

Для наглядності отриманих результатів було попередньо створено електронну таблицю *Excel*, яка була імпортована в *Google Maps* та на основі якої була створена інтерактивна карта.

В наступних версіях цієї програми можна додати інтерактивний інтерфейс, в якому користувач зможе комбінувати різноманітні методи відбору, схрещування та мутації з детальною інструкцією до кожного з них з метою покращення отриманих результатів, вбудувати в нього інтерактивні карти та графік з відображенням найкращий кандидатів для вирішення поставленої задачі в кожному поколінні.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. *IEEE* 802.5 *Token Ring* [*Electronic resource*]. – *Access mode*: *https*://*www.cs.wmich.edu*/*~yang*/*tlt*/*cs*555/*tokenring.htm* (*lastaccess*: 11.12.2023). – Title from the screen.

2. *Token Ring*/*IEEE* 802.5 [*Electronic resource*]. – *Access mode*: *http*://*www.upage.blogfa.com*/*post*/217 (*lastaccess*: 11.12.2023). – Title from the screen.

3. *Understanding IBM Token-Ring* [*Electronic resource*]. – Режим доступу: *https*://*flylib.com*/*books*/*en*/1.82.1.34/1/ (*lastaccess*: 11.12.2023). – Title from the screen.

4. Комп’ютерні мережі. Частина 3 : навч. посіб. / І. Р. Арсенюк, А. А. Яровий. – Вінниця : ВНТУ, 2017. – 85 с

5. *Understanding Token Ring* [*Electronic resource*]. – *Access mode*: *https*://*www.ardent-tool.com*/*network*/*Understanding_Token_Ring.html* (*lastaccess*: 11.12.2023). – Title from the screen.

6. *Token Ring Frame Formats* [*Electronic resource*]. – *Access mode*: *https*://*www.ccexpert.us*/*gigabit-ethernet*/*token-ring-frame-formats.html* (*lastaccess*: 11.12.2023). – Title from the screen.

7. *Troubleshooting Token Ring* [*Electronic resource*]. – *Access mode*: *https*://*www.cisco.com*/*en*/*US*/*docs*/*internetworking*/*troubleshooting*/*guide*/*tr*1906.*html* (*lastaccess*: 11.12.2023). – Title from the screen.

8. Технология *Token Ring* (802.5) [*Electronic resource*]. – *Access mode*: *http*://*math.gsu.by*/*wp-content*/*uploads*/*courses*/*networks*/*r*3.4.*html* (*lastaccess*: 11.12.2023). – Title from the screen.

9. Вимоги, пропоновані до сучасних обчислювальних мереж [Електронний ресурс]. – Режим доступу: *https://comp-net.at.ua*/*index*/*vimogi_do_komp_juternikh_merezh*/*0-7* (дата звернення 11.12.2023 р). – Назва з екрана.

10. *Horizontal vs*. *Vertical Scaling*: *Everything You Need to Know* [*Electronic resource*]. – *Access mode*: *https*://*phoenixnap.com*/*blog*/*horizontal-vs-vertical-scaling* (*lastaccess*: 11.12.2023). – Title from the screen.

11. *What is a Computer Network and Communication System*? [*Electronic resource*]. – *Access mode*: *https*://*er.yuvayana.org*/*what-is-a-computer-network-and-communication-system*/ (*lastaccess*: 11.12.2023). – Title from the screen.

12. *What is the difference between an undirected and a directed Graph*? [*Electronic resource*]. – *Access mode*: *https*://*www.geeksforgeeks.org*/*what-is-the-difference-between-an-undirected-and-a-directed-graph*/ (*lastaccess*: 11.12.2023). – Title from the screen.

13. *Types of Graphs with Examples* [*Electronic resource*]. – *Access mode*: *https*://*www.geeksforgeeks.org*/*graph-types-and-applications*/ (*lastaccess*: 11.12.2023). – Title from the screen.

14. *Types of Trees in Graph Theory* [*Electronic resource*]. – *Access mode*: *https*://*medium.com*/*@rohan*10*dalvi*/*types-of-trees-in-graph-theory*-62*ca*97*d*94089 (*lastaccess*: 11.12.2023). – Title from the screen.

15. *Euler Graph in Discrete Mathematics* [*Electronic resource*]. – *Access mode*: *https*://*www.javatpoint.com*/*euler-graph-in-discrete-mathematics* (*lastaccess*: 11.12.2023). – Title from the screen.

16. *Hamiltonian Graph in Discrete mathematics* [*Electronic resource*]. – *Access mode*: *https*://*www.javatpoint.com*/*hamiltonian-graph-in-discrete-mathematics* (*lastaccess*: 11.12.2023). – Title from the screen.

17. *The AHP Method*: *Definition and Example* [*Electronic resource*]. – *Access mode*: *https*://*www.indeed.com*/*career-advice*/*career-development*/*ahp-method* (*lastaccess*: 11.12.2023). – Title from the screen.

18. Метод аналізу ієрархій як інструмент для прийняття рішень при стратегічному плануванні [Електронний ресурс] –Режим доступу: *https://pidru4niki.com/15660721/menedzhment/metod_analizu_iyerarhiy_instrument_dlya_priynyattya_rishen_pri_strategichnomu_planuvanni* (дата звернення 11.12.2023 р). – Назва з екрана.

19. Метод аналізу ієрархій [Електронний ресурс] – Режим доступу: *https*://*dss.tg.ck.ua*/*ahp-help* (дата звернення 11.12.2023 р). – Назва з екрана.

20. Метод аналізу ієрархій [Електронний ресурс] – Режим доступу: *https://stud.com.ua/53023/ekonomika/metod_analizu_iyerarhiy* (дата звернення 11.12.2023 р). – Назва з екрана.

21. *Brute force approach* [*Electronic resource*]. – *Access mode*: *https://www.javatpoint.com/brute-force-approach* (*lastaccess*: 11.12.2023). – Title from the screen.

22. *Branch and Bound Algorithm* [*Electronic resource*]. – *Access mode*: *https://www.baeldung.com/cs/branch-and-bound* (*lastaccess*: 11.12.2023). – Title from the screen.

23. Алгоритм Літтла [Електронний ресурс] – Режим доступу: *http://ni.biz.ua/12/12_15/12_150704_entropiya.html* (дата звернення 11.12.2023 р). – Назва з екрана.

24. *Unveiling the Held-Karp Algorithm*: *Revolutionizing the Traveling Salesman Problem* [*Electronic resource*]. – *Access mode*: *https://medium.com/@data-overload/unveiling-the-held-karp-algorithm-revolutionizing-the-traveling-salesman-problem-9fb45b4cf58d* (*lastaccess*: 11.12.2023). – Title from the screen.

25. *Greedy Algorithm* [*Electronic resource*]. – *Access mode*: *https://www.programiz.com/dsa/greedy-algorithm#google_vignette* (*lastaccess*: 11.12.2023). – Title from the screen.

26. *Introduction to Ant Colony Optimization* [*Electronic resource*]. – *Access mode*: *https://www.geeksforgeeks.org/introduction-to-ant-colony-optimization/* (*lastaccess*: 11.12.2023). – Title from the screen.

27. *Optimization Techniques — Simulated Annealing* [*Electronic resource*]. – *Access mode*: *https://towardsdatascience.com/optimization-techniques-simulated-annealing-d6a4785a1de7* (*lastaccess*: 11.12.2023). – Title from the screen.

28. *Selections in genetic algorithms* [*Electronic resource*]. – *Access mode*: *https://www.linkedin.com/pulse/selections-genetic-algorithms-ali-karazmoodeh-g9yyf?trk=articles_directory* (*lastaccess*: 11.12.2023). – Title from the screen.

29. *Genetic Algorithms - Selection* [*Electronic resource*]. – *Access mode*: *https*://*medium*.*datadriveninvestor*.*com*/*genetic-algorithms-selection*-5634*cfc*45*d*78 (*lastaccess*: 11.12.2023). – Title from the screen.

30. *Crossover in Genetic Algorithm* [*Electronic resource*]. – *Access mode*: *https*://*www*.*geeksforgeeks*.*org*/*crossover-in-genetic-algorithm*/ (*lastaccess*:)

31. *Crossover Operators in Genetic Algorithm* [*Electronic resource*]. – *Access mode*: *https*://*medium*.*com*/*geekculture*/*crossover-operators-in-ga-cffa*77*cdd*0*c*8 (*lastaccess*: 11.12.2023). – Title from the screen.

32. *Genetic Algorithms – Mutation* [*Electronic resource*]. – *Access mode*: *https*://*www*.*tutorialspoint*.*com*/*genetic_algorithms*/*genetic_algorithms_mutation*.*htm* (*lastaccess*: 11.12.2023). – Title from the screen.

33. Слободян О. Положення про дипломні роботи (проєкти) випускників Національного авіаційного університету. – Київ: СМЯ НАУ, 2017. – 63 с.

ДОДАТКИ

Додаток А
Лістинг програми

```
import math

# Функція для обчислення характеристик матриці попарних порівнянь
def calculate_characteristics(matrix):
    # Розмірність матриці
    n = len(matrix)

    # Середнє геометричне для кожного рядка
    geometric_means = [math.pow(math.prod(row), 1/n) for row in matrix]

    # Сума значень середнього геометричного всіх рядків
    sum_geometric_means = sum(geometric_means)

    # Компоненти нормалізованого вектора пріорітетів (НВП)
    normalized_vector = [mean / sum_geometric_means for mean in geometric_means]

    column_sums = [sum(row[i] for row in matrix) for i in range(n)]

    eigenvalue = sum(column_sums[i] * normalized_vector[i] for i in range(n))

    # Індекс узгодженості (ІУ)
    consistency_index = (eigenvalue - n) / (n - 1)

    # Показник випадкової узгодженості (ПВУ) для матриці 3x3 дорівнює 0.58, для 4x4 - 0.9
    random_index = 0.58 if n == 3 else 0.9

    # Відносна узгодженість (ВУ)
    relative_consistency = consistency_index / random_index

    return geometric_means, sum_geometric_means, eigenvalue, consistency_index, normalized_vector,
relative_consistency, column_sums

# Значення матриць для розрахунків
matrix_3x3 = [
    [1, 5, 3],
    [1/5, 1, 1/3],
    [1/3, 3, 1]
]

matrix_4x4_1 = [
    [1, 3, 3, 5],    # Дерево
    [1/3, 1, 1, 3],  # Ейлерів граф
    [1/3, 1, 1, 3],  # Гамільтонів граф
```

```
    [1/5, 1/3, 1/3, 1]  # Повний граф
]

matrix_4x4_2 = [
    [1, 3, 3, 5],    # Дерево
    [1/3, 1, 1/3, 3],  # Ейлерів граф
    [1/3, 3, 1, 5],  # Гамільтонів граф
    [1/5, 1/3, 1/5, 1]  # Повний граф
]

matrix_4x4_3 = [
    [1, 1/7, 1/7, 1/3],    # Дерево
    [7, 1, 1/3, 5],  # Ейлерів граф
    [7, 3, 1, 5],  # Гамільтонів граф
    [3, 1/5, 1/5, 1]  # Повний граф
]

# Розрахунок характеристик для матриці 3x3
geometric_means_3x3, sum_geometric_means_3x3, eigenvalue_3x3, consistency_index_3x3,
normalized_vector_3x3, relative_consistency_3x3, column_sums_3x3 =
calculate_characteristics(matrix_3x3)
print("Значення для таблиці попарного порівняння критеріїв")
print("Середнє геометричне для кожного рядка:", geometric_means_3x3)
print("Сума значень середнього геометричного всіх рядків:", sum_geometric_means_3x3)
print("Компоненти нормалізованого вектора пріорітетів (НВП):", normalized_vector_3x3)
print("Власне значення матриці (λmax):", eigenvalue_3x3)
print("Індекс узгодженості (ІУ):", consistency_index_3x3)
print("Відносна узгодженість (ВУ):", relative_consistency_3x3)

# Розрахунок характеристик для матриці 4x4 за різними критеріями
geometric_means_4x4_1, sum_geometric_means_4x4_1, eigenvalue_4x4_1, consistency_index_4x4_1,
normalized_vector_4x4_1, relative_consistency_4x4_1, column_sums_4x4_1 =
calculate_characteristics(matrix_4x4_1)
geometric_means_4x4_2, sum_geometric_means_4x4_2, eigenvalue_4x4_2, consistency_index_4x4_2,
normalized_vector_4x4_2, relative_consistency_4x4_2, column_sums_4x4_2 =
calculate_characteristics(matrix_4x4_2)
geometric_means_4x4_3, sum_geometric_means_4x4_3, eigenvalue_4x4_3, consistency_index_4x4_3,
normalized_vector_4x4_3, relative_consistency_4x4_3, column_sums_4x4_3 =
calculate_characteristics(matrix_4x4_3)
print("\nЗначення для таблиці попарного порівняння альтернатив по критерію витрати на розгортання")
print("Середнє геометричне для кожного рядка:", geometric_means_4x4_1)
print("Сума значень середнього геометричного всіх рядків:", sum_geometric_means_4x4_1)
print("Компоненти нормалізованого вектора пріорітетів (НВП):", normalized_vector_4x4_1)
print("Власне значення матриці (λmax):", eigenvalue_4x4_1)
print("Індекс узгодженості (ІУ):", consistency_index_4x4_1)
print("Відносна узгодженість (ВУ):", relative_consistency_4x4_1)

print("\nЗначення для таблиці попарного порівняння альтернатив по критерію захищенність")
print("Середнє геометричне для кожного рядка:", geometric_means_4x4_2)
```

```
print("Сума значень середнього геометричного всіх рядків:", sum_geometric_means_4x4_2)
print("Компоненти нормалізованого вектора пріорітетів (НВП):", normalized_vector_4x4_2)
print("Власне значення матриці (λmax):", eigenvalue_4x4_2)
print("Індекс узгодженості (ІУ):", consistency_index_4x4_2)
print("Відносна узгодженість (ВУ):", relative_consistency_4x4_2)

print("\nЗначення для таблиці попарного порівняння альтернатив по критерію ступінь зв'язності 2")
print("Середнє геометричне для кожного рядка:", geometric_means_4x4_3)
print("Сума значень середнього геометричного всіх рядків:", sum_geometric_means_4x4_3)
print("Компоненти нормалізованого вектора пріорітетів (НВП):", normalized_vector_4x4_3)
print("Власне значення матриці (λmax):", eigenvalue_4x4_3)
print("Індекс узгодженості (ІУ):", consistency_index_4x4_3)
print("Відносна узгодженість (ВУ):", relative_consistency_4x4_3)

table = [
    normalized_vector_3x3,
    [normalized_vector_4x4_3[0], normalized_vector_4x4_1[0], normalized_vector_4x4_2[0]],
    [normalized_vector_4x4_3[1], normalized_vector_4x4_1[1], normalized_vector_4x4_2[1]],
    [normalized_vector_4x4_3[2], normalized_vector_4x4_1[2], normalized_vector_4x4_2[2]],
    [normalized_vector_4x4_3[3], normalized_vector_4x4_1[3], normalized_vector_4x4_2[3]]
]

print("\nПідсумкова таблиця")
for row in table:
    print('\t'.join(map(str, row)))

print(" ")
for i in range(1, len(table)):
    global_priority = sum(table[0][j] * table[i][j] for j in range(len(normalized_vector_3x3)))
    print(f"Глобальний пріоритет_{i}: {global_priority}")
```