Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____*Комп'ютерних наук*_____
(повна назва)
Кафедра_____*Системотехніки*_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти_____другий (магістерський)_____

Дослідження мурашиного алгоритму для побудови маршрутів на картах рогейну
(тема)

Виконав: студент 2 курсу, групи *ІТПм-22-1*
Спеціальності *122 Інформаційні технології проектуванняі*
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма *Комп'ютерні науки*
(повна назва освітньої програми)
Ісаєнко А.Ю.
(прізвище, ініціали)
Керівник *проф. Іванов В.Г.*
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри системотехніки _____ *проф. Гребеннік І.В.*
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет *Комп'ютерних наук*

Кафедра *Системотехніки*

Рівень вищої освіти другий (магістерський)

Спеціальність *122 Інформаційні технології проектування*
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма *Комп'ютерні науки*
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____20___р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові *Ісаєнко Андрію Юрійовичу*
(прізвище, ім'я, по батькові)

1. Тема роботи: *Дослідження мурашиного алгоритму для побудови маршрутів на картах рогейну*
затверджена наказом по університету від *20.11* 2023 р. №*1373Ст*
2. Термін подання студентом роботи до екзаменаційної комісії: 18.01.2024 р
3.Вихідні дані до роботи: *регламент змагань, способи побудови маршрутів на мапі, , оптимізаційні алгоритми, модифікація мурашиного алгоритму координати КП, вага КП, довжина дистанції, тривалість змагань, швидкість, масштаб мапи.*

4. Перелік питань, що потрібно опрацювати в роботі: *Вступ. Аналіз предметної області пов'язаною з побудовою маршрутів на мапах спортивного орієнтування. Огляд оптимізаційних алгоритмів Розробка вимог до алгоритму. Вибір алгоритму. Розробка модифікацій. Модифікація та аналіз алгоритму. Висновки*

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій: *мапа, граф, інтерфейс застосунку, блок схеми алгоритму, приклади маршрутів, графіки роботи алгоритму на кожній мапі, графіки роботи ефективності алгоритмів.*

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін и виконання етапів роботи	Примітка
1	*Отримання завдання на виконання роботи*	*16.10.2023*	
2	*Огляд критеріїв для розробки методу порівняння*	*17-22.11.2023*	
3	*Огляд критеріїв порівняльного аналізу*	*23-31.11.2023*	
4	*Огляд літератури*	*01-17.11.2023*	
5	*Опис математичної моделі порівняльного аналізуі*	*17-30.11.2023*	
6	*Розробка експериментальних систем*	*01-11.12.2023*	
7	*Проведення експерименту*	*11-14.12.2023*	
8	*Збор та аналіз даних експерименту*	*14-20.12.2023*	
9	*Оформлення пояснювальної записки*	*20-29.12.2023*	
10	*Представлення на рецензування*	*03.01.2024*	

Дата видачі завдання 16.10.2023 р.

Студент _____ *Ісаєнко А.Ю.*
(підпис)

Керівник роботи _____ *проф. Іванов В.Г.*
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи магістра: 79 с., 22 рис.

РОГЕЙН, МУРАШИНИЙ АЛГОРИТМ, МАРШРУТ, МОДИФІКАЦІЯ, МЕТАВЕРИСТИКА, ОПТИМІЗАЦІЙНОГО АЛГОРИТМУ.

Об’єктом розробки є функція для побудови маршруту на картах з рогейну

Предметом розробки є дослідження та модифікація мурашиного алгоритму, для дотримання умов побудови маршруту

Метою роботи є дослідження мурашиного алгоритму та його модифікація для дотримання обмежень які накладає специфіка змагань з рогейну.

Система дозволить користувачам застосунку отримувати маршрут на карті згідно тим даним що вони ввели.

Методом розробки є функціональний підхід проектування програмного забезпечення.

Результатом кваліфікаційної роботи магістра є дослідження алгоритму та знаходження способу його модифікації.

Область застосування – розроблений алгоритм може використовуватися в застосунку для побудови маршрутів для особистого користування спортсменами.

ABSTRACT

Explanatory note to the qualification work of the masters contains: 79 pages, 14 figures.

ROGAINE, ANT ALGORITHM, ROUTE, MODIFICATION, METAVERISTICS, OPTIMIZATION ALGORITHM.

The object of study is functions for building routes on rogaine maps.

The subject of study is the research and modification of the ant algorithm, to comply with the conditions for constructing the route

The purpose of the is to куыуфкср the ant algorithm and its modification to comply with the limitations imposed by the specifics of rogaine competitions.

The system will allow users of the application to receive a route on the map according to the data they entered.

The method of development is a functional approach to software design

The result of the master's certification work s the study of the algorithm and finding a way to modify it.

Scope - the developed algorithm can be used in the application to build routes for personal use by athletes.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

КП – контрольний пункт

ACO – ant colony optimization

ІС – інформаційна система;

WF – Windows Forms

БД – база даних;

СУБД – система управління базами даних

ВСТУП

Рогейн - це командний вид орієнтування на місцевості[1], де спортсмени мають скласти маршрут між контрольними пунктами (КП), визначивши дистанцію, яку вони планують пробігти, залежно від тривалості змагань. Основною метою є збір якнайбільше балів, які присуджуються за відвідування КП. Побудова маршруту є складним і обговорюваним етапом, оскільки він не має чіткого алгоритму. Проте успішність команди на змаганнях може повністю залежати від цього етапу. Система розробляється з метою того, щоб після змагань спортсмен мав можливість проаналізувати свій побудований маршрут, порівняти його з оптимальним і виявити можливі помилки. Крім того, цю систему можна використовувати для навчання спортсменів побудові оптимальних маршрутів під час тренувань та проведення досліджень.

Метою роботи є дослідження мурашиного та його модифікація що б використовувати його для побудови маршрутів на мапах з рогейну.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОПИС ОБ’ЄКТА ДОСЛІДЖЕННЯ

1.1 Аналіз предметної області

Рогейн є доволі молодим видом спортивного орієнтування, і не має великої розповсюдженості через те що має великі вимоги до фізичної підготовки учасників, складнішу організацію і специфічні самі правила змагань. Але водночас є і його візитною карткою для прихильників цього виду спорту. Спортсменам необхідно створити свій власний маршрут через рознесені по мапі (рис. 1.1) КП, зібравши якомога більше балів, і в кластися в контрольний час.

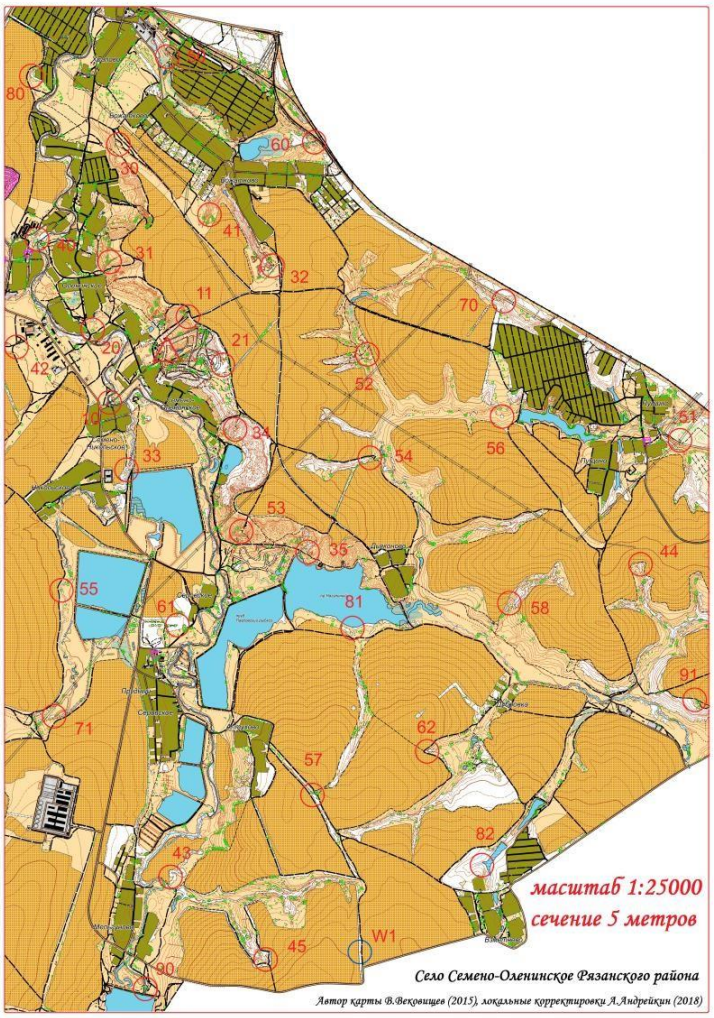

Рисунок 1.1 – Карта для рогейну

Традиційна тривалість змагань з рогейну – 24 години. Але існують й інші

варіанти на 12, 6 та 3 години. Кількість балів за відвідування КП визначається першою цифрою його номеру. Кількість КП зазвичай визначається організаторами змагань і зазвичай варіюється в районі 30-70, залежно від місцевості та тривалості змагань. Наразі не існує жодного визначеного алгоритму для побудови маршрутів для змагань з рогейну, і спортсмени зазвичай покладаються на свій досвід, покладаючись на евристичний метод. По-перше спортсменам необхідно визначити дистанцію яку вони можуть подолати за відведнний час. Далі для побудови маршруту вони можуть використати курвіметр, що б знати яку довжину по мапі вони планують подолати. Але більшою популярністю користуються побудова маршруту методом голок та нитки. Вони відміряють нитку запланованої довжини в масштабі до мапи, роблять з неї петлю, бо фініш і старт зазвичай знаходяться в одному місці, і це означає круговий маршрут. Далі в КП втикаються канцелярські голки. Петля накидується на них, і методом перетягування спортсмени намагаються знайти маршрут який принесе їм найбільше балів. Але доволі важко швидко на око визначити, буде цей район з КП вигідніший, чи інший.

Таким чином, без відсутності точної послідовності дій алгоритмізувати знаходження оптимального маршруту є доволі складною задачею. Тому було обрано варіант разглянути існуючі задачі[2], наближені до цієї.

1.2 Аналіз задач комбінаторної оптимізації

Задачі комбінаторної оптимізації - це клас задач оптимізації, в яких метою є знаходження оптимального рішення зі скінченного набору можливих комбінацій, щоб максимізувати або мінімізувати певну цільову функцію. У цих задачах рішення представляється набором об'єктів або елементів, які піддаються вибору або перестановці, і існує обмеження на комбінації, які можуть бути вибрані чи розглянуті.

Задача побудови маршруту на картах з рогейну нагадує такі комбінаторні

оптимізаційні задачі як задача комівояжера або задача про рюкзак. Але з певними відмінностями. Це виглядає як комбінація задачі комівояжера та задачі про рюкзак. Оскільки вона включає в себе пошук оптимального маршруту[3] як у задачі комівояжера і мінімізації довжини маршруту. А оскільки необов'язково відвідувати всі вершини, і головна мета - максимізація вартості шляху за обмеження на його довжину, це відповідає характеристикам задачі про рюкзак.

Отже можна звернутися до розв'язань та алгоритмів, що використовуються для задачі комівояжера або задачі про рюкзак для розв'язання даної задачі, зробивши певні модифікації. Але для початку треба більш детально розглянути відмінності між цими задачами.

Схожість із задачею комівояжера. Як у задачі комівояжера, шукається оптимальний маршрут, пройшовши через певні точки (КП) і маючи обмеження на максимальну довжину маршруту.

Схожість із задачею про рюкзак. Є обмеження на сумарний обсяг (довжину) предметів (КП) у "рюкзаку" (маршруті).

Вартість (бали) КП є вагою предмету в задачі про рюкзак.

Особливості. У задачі не потрібно відвідувати всі КП. Це відрізняється від класичної задачі комівояжера, де всі точки повинні бути відвідані. Основна мета - максимізувати сумарну вартість балів за КП на обмеженій довжині маршруту.

Існують різні сервіси та калькулятори для знаходження рішення подібних задач, але за невеликої кількості точок. У такому випадку можна точно знайти найкращий варіант звичаним методом перебору. Але для великої кількості точок такий мето вже буде дуже неефективним, через факторіальну складність алгориму - O(n!). експоненційний зріст часу виконання. Як сказано в прикладі, вже при відносно невеликій кількості міст (>66) вона не може бути вирішена методом перебору варіантів ніякими теоретично мислимими комп'ютерами за час, менше кількох мільярдів років[4]. Тому необхідно розглянути існуючі алгоритми для розв’язання цих задач, для обирання алгоритму який можна буде модифікувати.

2 ПОСТАНОВКА ЗАДАЧІ НА ДОСЛІДЖЕННЯ

2.1 Опис задачі

Найкращим варінтом представлення мапи з КП є граф (рис 1.2). Вершини є об’єктами, які означають КП, та мають параметр номеру вершини, шо буде визначати також і бали за неї. Ребра графа будуть об’єктами які показують можливі переходи між вершинами, а також мати параметр довжини переходу. Повний граф представлятиме с себе два списки з об’єктами цих класів. Його можна конвертувати на матрицю інцидентності для простішого оперування даними ти списком з номерів вершин. На виході ми маємо отримати як раз послідовність вершин маршруту.

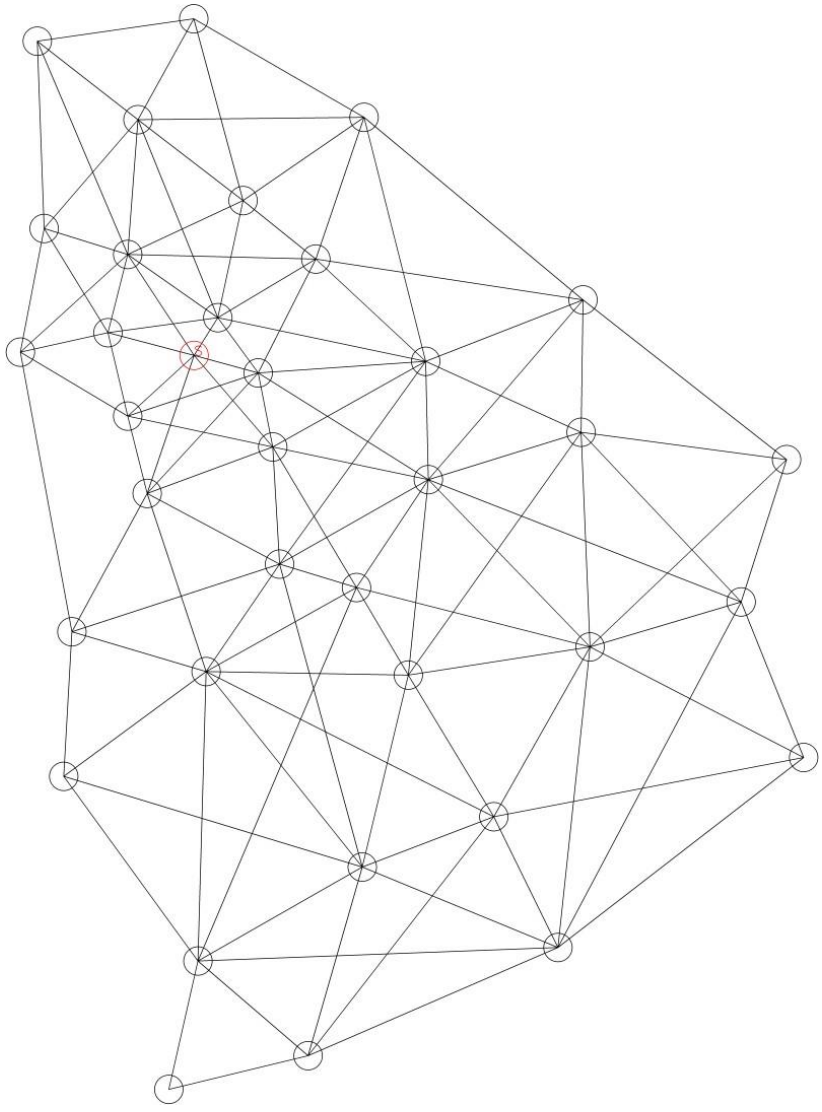

Рисунок 1.2 – Графічне відображення побудованого графа

Основною метою є розробка оптимального маршруту на графі, що відображає мапу з КП. За правилами рогейну мапа створюється таким чином, що неможливо відвідати всі КП за виділений час на проходження дистанції[5]. Це означає, що для пошуку оптимального маршруту не потрібно проводити повне обстеження графа. Таким чином, перше обмеження визначається довжиною маршруту, яку не можна перевищувати. Це змінна в алгоритмі, яку спортсмени визначають самостійно перед змаганням, встановлюючи кількість кілометрів або приблизно визначаючи швидкість, з якою планують пройти весь шлях. На рисунку 1.3 показаний приклад маршруту довжиною 20 км яким спортсмени можуть пробігти на змаганнях

Рисунок 1.3 – Маршрут на змаганнях з рогейну

Довжина маршруту буде визначатися сумуванням довжин ребер між вершинами які будуть відвідуватися. Крім відстаней між точками, існує параметр вартості контрольних пунктів, тобто вартість вершин графа. Вартість маршруту залежить від цього параметра, оскільки відвідуючи контрольний пункт, спортсмен отримує за нього бали, а потім бали всіх відвіданих контрольних пунктів сумуються. Максимізація кількості балів є головною метою. Отже, задача пошуку оптимального маршруту полягає в знаходженні найдорожчого маршруту за заданої його максимальної довжини[6]. Це буде критерій оптимальності маршруту.

$$\max_{l \le L} \sum_{i \in V} v_i \quad (2.1)$$

де l – довжина отриманого маршруту, L – задана користувачем максимальна довжина маршруту, V – множина вершин, v – кількість балів за вершину.

Задача має добре працювати при великій кількості змінних. Будемо вважати, що час роботи має бути прийнятним для користувача.

2.2 Постановка задачі

Об’єкт розробки - функція для побудови маршруту на картах з рогейну

Предмет розробки – дослідження та модифікація мурашиного алгоритму, для дотримання умов побудови маршруту

Елементи модифікації класичного мурашиного алгоритму:

— Додавання параметра балів за вершину.

— Додавання параметру довжини маршруту, яку не можна перевищувати.

— Вибір маршруту обмеженої довжини з максимальною кількістю балів за відвідані вершини.

— Зміна критерія вибору наступного кроку з урахуванням балів за вершину і відстані до неї

— Максимізаця суми балів за відвідані вершини, замість мінімізації довжини маршруту, з урахуванням обмеження цієї довжини.

Вхідні даними даними є мапа змагань, яка завантажується в попередньо розроблений застосунок. В цьому застосунку користувач мишкою позначає розташування КП та вводить їх номери. Застосунок автоматично конвертує їх в об'єкт графа з вершинами та ребрами. Об'єкт графа і максимальна дистанція є вхідними даними для модифікованого алгоритму.

Вихідними даними алгоритму є масив з послідовністю вершин, яка відображає оптимальный маршруту. Для зручності користувача, він буде відображатися на мапі у застосунку.

Сфера застосування: за правилами ронейну, спортсмени не можуть користуватися сторонніми засобами побудови маршруту, тому алгоритм не має виконуватися в режимі реального часу. Він буде використовуватися лише для навчання та аналізу.

Задача має добре працювати при великій кількості змінних. Орієнтована кількість точок у графі, відповідно до кількості КП на мапі буде 30-70

Час роботи має бути прийнятним для користувача, відповідно до сфери застосування. Для такої умови встановимо обмеження у 10 хвилин для максимальної кількості точок.

3 ДОСЛІДЖЕННЯ МЕТОДІВ ВИРІШЕННЯ ЗАДАЧІ

3.1 Вибір алгоритму для модифікації:

3.1.1 Опис задач комівояжера та задачі про рюкзак

Задача комівояжера і задача про рюкзак - це дві різні оптимізаційні задачі, які часто виникають в області оптимізації та комбінаторної оптимізації.

Задача комівояжера. В даній задачі є граф, в якому кожен вузол представляє місто, а ребра представляють можливі шляхи між містами. Мета полягає в тому, щоб знайти найкоротший шлях, який проходить через кожне місто рівно один раз і повертається в початкове місто. Ця задача належить до класу NP, що означає, що немає ефективного алгоритму для знаходження точного рішення за прийнятний час.

Задача про рюкзак. У цій задачі є набір предметів, кожен з яких має свою вагу і цінність. Також є рюкзак з обмеженою вагою. Мета полягає в тому, щоб вибрати підмножину предметів так, щоб їх вага не перевищувала обмеження рюкзака, а сума цінностей була максимальною. Ця задача також є класу NP.

Отже, основна відмінність полягає в характері задач: задача комівояжера пов'язана з пошуком оптимального шляху в графі, тоді як задача про рюкзак стосується вибору оптимального підмножини предметів з обмеженим об'ємом (вагою).

За характером цих задач можна зрозуміти, що поставлена задача також відноситься до класу NP. Знання класу складності вашої є корисним з декількох причин. Знання класу складності допомагає розуміти теоретичні обмеження та важкість задачі в контексті обчислювальної теорії. Також класу задачі може вплинути на вибір алгоритмів для її розв'язання. Це може допомогти визначити, чи існують відомі ефективні стратегії оптимізації для конкретної задачі. Знання

класу складності допомагає визначити границі того, наскільки ефективно можна вирішити вашу задачу відносно розміру вхідних даних.

Те, що задача належить до класу NP, має декілька важливих наслідків та значень. Наявність невідомого ефективного алгоритму перевірки. Якщо задача належить до класу NP, то для неї існує невідомий ефективний алгоритм для перевірки правильності потенційного розв'язку. Це означає, що якщо мається потенційне рішення, можна перевірити його правильність за поліноміальний час. Якщо задача є NP-повною, це свідчить про те, що пошук оптимального розв'язку цієї задачі найімовірніше є важким, і немає відомого ефективного алгоритму для його знаходження за поліноміальний час. Приналежність задачі до класу NP, дає можливість зведення або перетворення задачі до інших задач з класу NP. Це дозволяє визначити взаємозв'язки та аналогії між різними задачами, як це зроблено у випадку пошуку оптимального маршруту. Знання, що задача належить до класу NP, дозволяє розглядати загальні методи і підходи для розв'язання задач цього класу, такі як евристичні методи, алгоритми з приблизним розв'язком, метаевристичні методи, генетичні алгоритми тощо.

3.1.2 Огляд методів вирішення задач комівояжера та задачі про рюкзак

Методи вирішення задачі комівояжера:

— Повний перебір.
— Метод гілок і меж.
— Динамічне програмування.
— Метаевристичні методи.

Методи вирішення задачі про рюкзак:

— Жадібний алгоритм.
— Динамічне програмування.
— Метаевристичні методи.

Задача про рюкзак.

Повний перебір включає в себе розгляд усіх можливих комбінацій міст для знаходження оптимального шляху. Алгоритм вибирає початковий міст, перебирає всі можливі шляхи, і обирає той, який має найменшу сумарну відстань. Цей метод дозволяє знайти точне оптимальне рішення, але його ефективність сильно знижується зі збільшенням кількості міст.

Метод гілок і меж використовує дерево вирішення, де кожен вузол представляє можливий шлях. Він вирішує підзадачі та обрізає гілки, які не можуть призвести до оптимального рішення. Це робиться за допомогою оцінювання верхньої границі для кожної гілки. Метод гарантує знаходження оптимального рішення, але його ефективність залежить від якості обрізання гілок.

Динамічне програмування вирішує задачу шляхом розбиття її на менші підзадачі. Спочатку створюється матриця для збереження оптимальних відстаней між парами міст. Далі, для кожного міста обчислюється оптимальний шлях, який включає це місто та інші міста. Метод гарантує оптимальність рішення, але його обмеження включає велику використання пам'яті та високий час обчислень.

Метаевристичні методи, такі як генетичні алгоритми, віджимання та мурашині алгоритми, застосовують еволюційні чи навчальні підходи до пошуку оптимального шляху. Генетичні алгоритми використовують концепції еволюції для покращення популяції шляхів. Віджимання вдосконалює поточний шлях, використовуючи локальні операції. Мурашині алгоритми моделюють поведінку мурах для знаходження оптимального шляху. Вони надають швидкі наближені рішення, але не гарантують знаходження оптимального рішення.

Задача про рюкзак.

Жадібний алгоритм вирішує задачу шляхом вибору предметів з найвищим відношенням цінності до ваги. Починаючи з порожнього рюкзака, алгоритм по черзі додає предмети, починаючи з тих, які мають найкраще

відношення цінності до ваги. Жадібний підхід простий, але може не завжди гарантувати оптимальне рішення.

Динамічне програмування для задачі рюкзака застосовує той самий принцип розбиття задачі на менші підзадачі. Таблиця заповнюється для збереження оптимальних рішень підзадач. Алгоритм розглядає кожен предмет і обчислює, чи його включення до рюкзака може покращити цінність. Метод гарантує оптимальність рішення, але може бути вимогливим до пам'яті для великих задач.

Метаевристичні методи для задачі рюкзака включають генетичні алгоритми та віджимання. Генетичні алгоритми використовують еволюційний підхід, емулюючи процеси вибору, кросовера та мутації для вдосконалення комбінації предметів. Віджимання спрощається на локальній оптимізації, вибираючи оптимальний піднабір предметів для покращення поточного рюкзака.

Так як для нашої задачі важливе швидке знаходження рішення при великій кількості вершин графа, а також допускається знаходження не точного, а наближеного рішення було обрано розглянути метаеврестичні методи її вирішення.

3.1.3 Метаеврестичні методи вирішення задач

Метаевристичні методи, такі як генетичні алгоритми, віджимання та мурашині алгоритми, застосовують еволюційні чи навчальні підходи до пошуку оптимального маршруту. Вибір конкретного оптимізаційного алгоритму залежить від конкретних характеристик та вимог самої задачі. Щоб краще визначити, який алгоритм може бути більш відповідним, ватро врахувати певні фактори, які можуть мати вплив. Для роботи в великим графом, може бути доцільно вибрати алгоритми, які мають низьку обчислювальну складність. Треба врахувати специфічні обмеження, наприклад, обмеження на час або інші параметри, що б вибрати алгоритм, який може ефективно працювати з цими

обмеженнями. Для точного рішення, необхідно вибрати алгоритми, які гарантують знаходження глобального оптимуму. Якщо задача має динамічний характер, то будуть корисними алгоритми, які можуть адаптуватися до змін у середовищі. Так само важливо, щоб алгоритм був легко налаштовуваним та підтримуваним,

До алгоритмів, які б можливо були ефективними для вирішення задачі, відносяться алгоритм рою частинок (PSO), метод вовчої зграї або мурашиний алгоритм. Для того що б обрати між цими алгоритмами необхідно провести порівняльний провести порівняльний огляд алгоритмів.

3.1.3.1 Алгоритм рою частинок (PSO):

Алгоритм рою частинок моделює поведінку рою частинок, які рухаються в просторі пошуку з метою знаходження оптимального рішення. Кожна частинка представляється точкою у просторі, а її рух визначається швидкістю та напрямком. Кожна частинка пам'ятає найкраще рішення, яке вона вже знайшла, і найкраще рішення свого сусіда. Алгоритм працює на принципі кооперації, де частинки спільно оптимізують своє рухове просторове положення. Він включає кроки оновлення швидкості та переміщення для кожної частинки, а також враховує вплив глобального та локального найкращих рішень. PSO ефективно використовує концепції координації та колективного інтелекту для знаходження оптимального рішення в просторі пошуку.

Якщо важливо врахувати динаміку зміни маршруту та ефективність в умовах складного ландшафту, PSO може бути корисним.

Переваги: PSO може виявити високу швидкість знаходження рішення, особливо в просторах пошуку великого обсягу. Ефективний для оптимізації у многовимірних просторах параметрів. Глобальна кооперація між частинками може допомогти уникнути застрягання в локальних мінімумах. Простота реалізації

Обмеження: Залежність від параметрів може зробити алгоритм чутливим до їхнього вибору. Якщо простір пошуку має складну структуру, може виникнути проблема виявлення оптимального маршруту в області з численними мінімумами.

3.1.3.2 Алгоритм вовчої зграї:

Алгоритм вовчої зграї натхненний поведінкою вовків у природі та розв'язанням проблем координації та співпраці в групі. У вовчій зграї розв'язки представляють індивідуальні вовки, а їхні рухи та взаємодії моделюють різні стратегії полювання. Алгоритм включає кроки оновлення позицій та швидкостей вовків, де кожен вовк враховує позиції інших вовків для прийняття оптимального рішення. Вовча зграя використовує механізми пошуку, подібні до генетичних алгоритмів, для постійного вдосконалення розв'язків. Цей алгоритм демонструє здатність до швидкого знаходження оптимальних рішень завдяки використанню стратегій індивідуальної та групової оптимізації.

Якщо ваша задача вимагає співпраці між агентами та враховує соціальну структуру, вовча зграя може бути варіантом.

Переваги:

— Вовча зграя використовує стратегії вовків, що може бути ефективним для пошуку в областях з різними оптимальними маршрутами.

— Має елементи глобального та локального пошуку.

— Здатний до глобальної оптимізації.

Обмеження:

— Потребує установки параметрів, і їхні зміни можуть впливати на результати.

— В частинних випадках може виявити меншу швидкість збіжності порівняно із швидшими алгоритмами.

— Чутливий до великої кількості агентів.

3.1.3.3 Мурашиний алгоритм:

Мурашиний алгоритм базується на моделюванні поведінки мурах при пошуку шляху в природі. У віртуальному середовищі мурашки залишають феромони на шляхах, які вони пройшли, і інші мурахи використовують ці феромони для прийняття рішення про вибір шляху. Алгоритм включає процеси відкладання феромонів, оновлення концентрацій феромонів, та вибір шляхів з урахуванням феромонів та інших факторів. Мурашиний алгоритм здатний знаходити оптимальні рішення в просторі пошуку завдяки комбінації глобального та локального пошуку [7]. Його ефективність підтримується ідеєю взаємодії та співпраці мурашок для вирішення складних задач оптимізації.

Якщо задача включає велику кількість вершин і обмеження на максимальну довжину маршруту, мурашиний алгоритм може бути гарним вибором.

Переваги:

— Мурашиний алгоритм добре підходить для задач комбінаторної оптимізації, таких як задача комівояжера комівояжера та інших графових задач.

— Добре враховує обмеження на довжину маршруту та може оптимізувати вартість шляху.

— Здатний розв'язувати задачі з великою кількістю можливих маршрутів.

Обмеження:

— Велика кількість параметрів може потребувати налаштування.

— Швидкість збіжності може бути вплинута великим обсягом можливих маршрутів та великими обчислювальними витратами.

3.1.4 Вибір між алгоритмами:

Вибір між цими алгоритмами повинен бути зроблений на підставі конкретних характеристик задачі, вимог до швидкості, точності та можливостей

влаштування алгоритму, які були описані в постановці задачі.

Для задачі знайдення найдорожчого маршруту за заданої його максимальної довжини, мурашиний алгоритм може бути відмінним вибором, оскільки він спеціалізується на задачах комбінаторної оптимізації на графах. Він може ефективно враховувати обмеження довжини маршруту та максимізувати вартість шляху, що є ключовим для даної задачі, спираючись на принципи взаємодії мурашиних колоній та феромонів.

У порівнянні з алгоритмом рою частинок (PSO), мурашиний алгоритм вигідний своєю спроможністю ефективно вирішувати комбінаторні оптимізаційні задачі, такі як пошук оптимального маршруту на графі. Його здатність враховувати феромони та інші фактори прийняття рішень може допомогти зберігати та розвивати інформацію про найдорожчі маршрути, що може бути важливо для оптимізації шляху.

У порівнянні з алгоритмом вовчої зграї, мурашиний алгоритм демонструє високу ефективність у комбінаторних задачах і може забезпечити більш точні та оптимальні рішення в умовах обмежень. Водночас, враховуючи специфіку задачі максимізації вартості при заданій довжині маршруту, мурашиний алгоритм зберігає свою актуальність.

Загалом, мурашиний алгоритм є потенційно найбільш підходящим для даної задачі, оскільки його особливості та переваги добре відповідають умовам пошуку оптимального маршруту у графі

Крім оптимізаційних алгоритмів , мурашиний було порівняно з іншими перспективними алгоритмами для цієї задачі. Розглянемо недоліки кожного з цих алгоритмів в порівнянні з мурашиним алгоритмом для вирішення конкретної задачі.

У задачі про рюкзак, жадібний алгоритм може обирати предмети з великою вагою та низькою вартістю, не гарантуючи оптимального рішення. Динамічне програмування може стати витратним при великій кількості предметів через високу обчислювальну складність. Метод гілок та границь може вимагати обчислення верхньої межі для багатьох підзадач, що також

може бути витратним.

У задачі комівояжера, метод грубої сили має експоненційну складність при збільшенні кількості міст, що робить його неефективним для великих обсягів даних. Жадібний алгоритм може застрягнути в локальних мінімумах та не гарантувати знаходження оптимального рішення. Динамічне програмування також може виявитися витратним для великої кількості міст через високу обчислювальну складність.

Мурашиний алгоритм, хоча і має свої обмеження, включає в себе можливість застрягання в локальних оптимумах та вимагає налаштування параметрів, проте він відзначається розділеним пошуком та адаптивністю до змін, що може бути корисним для вирішення задачі.

Мурашині алгоритми мають декілька важливих переваг, які можуть зробити їх привабливими для вирішення цієї задачі оптимізації маршруту. Мурашиний алгоритм використовує розділений пошук, де кожна мураха будує свій власний маршрут. Це може бути особливо корисним, коли необхідно знайти глобально оптимальний маршрут у складних умовах задачі. Мурахи розподіляються по різних частинах простору шляхів, що дозволяє ефективно досліджувати його. Іншою важливою перевагою є адаптивність мурашиного алгоритму до змін у вихідних умовах. Він може швидко враховувати зміни в обмеженнях чи вартості вершин, що робить його добре придатним для динамічних задач. Мурашиний алгоритм також може уникати локальних мінімумів завдяки стохастичності в обранні шляху. Його елемент випадковості дозволяє уникати застрягання в локальних оптимумах та сприяє пошуку глобально оптимального рішення. Ще однією важливою характеристикою є імітація соціальної взаємодії мурах в природі. Це може бути корисним, коли взаємодія між різними частинами системи, такими як КП на маршруті, може впливати на оптимальний вибір шляху. Крім того, мурашині алгоритми легко розширюються для врахування додаткових умов чи обмежень, що може бути важливим для налаштування під конкретні вимоги вашої задачі.

3.2 Мурашиний алгоритм

3.2.1 Опис мурашиного алгоритму

Мурашиний алгоритм (ant colony optimization, ACO) є ефективним поліноміальним методом для наближеного розв'язання проблеми комівояжера та подібних задач пошуку маршрутів у графах[8]. Основна ідея полягає у вивченні та використанні моделі поведінки мурах, які прокладають шляхи від колонії до джерела їжі (рис. 3.1), що є метаевристичною стратегією оптимізації.

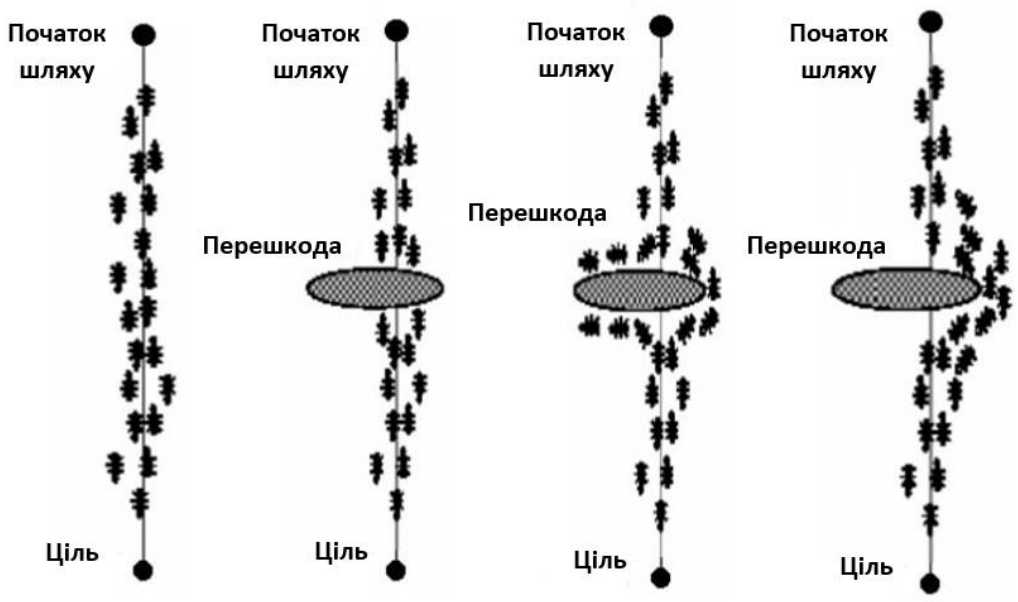

Рисунок 3.1 – Пошук мурахами оптимального шляху до джерела їжі

У природі, мурахи спочатку рухаються хаотично, наприклад, у пошуках їжі. Коли вони її знаходять, повертаються до своєї колонії, залишаючи за собою феромонні сліди. Якщо інші мурахи натрапляють на цей шлях, вони не продовжують блукати хаотично, а навпаки, йдуть за цим шляхом, підсилюючи його, якщо знаходять їжу.

Але з часом феромон повинен випаровуватися, зменшуючи його привабливість. Чим більше часу потрібно мурахам для проходження цього шляху та повернення, тим більше часу феромонам потрібно для випаровування.

Коротший шлях використовується частіше, тому кількість феромонів стає вищою на коротших шляхах, ніж на довших. Випаровування феромонів також має перевагу у тому, що воно допомагає уникнути конвергенції до локально оптимальних рішень. Якби не було випаровування взагалі, шляхи, вибрані першими мурахами, часто ставали б привабливими для інших. У такому випадку обмеження області пошуку рішень було б значною. Вплив випаровування феромонів у реальних мурашиних системах ще не повністю зрозумілий, але він виявляється ключовим у штучних системах[9].

Загальна суть полягає в тому, що коли одна мураха знаходить оптимальний (короткий) шлях від колонії до джерела їжі, інші мурахи виявляють схильність використовувати цей самий маршрут. Позитивний зворотний зв'язок призводить до того, що багато мурах обирають той самий шлях. Основна ідея алгоритму мурашиної колонії полягає в імітації цієї природної поведінки за допомогою "симульованих мурах", що рухаються по графу, який представляє проблему, яку необхідно вирішити.

У визначеному алгоритмі оптимізації мурашиним методом, симульовані мурахи функціонують як обчислювальні агенти, які шукають ефективні рішення для задачі оптимізації. Для використання алгоритму мурашиної колонії, проблему оптимізації необхідно трансформувати в пошук найкоротшого шляху у зваженому графі. На першому етапі кожної ітерації кожна мураха створює рішення, стохастично визначаючи порядок проходження ребер графу. На другому етапі порівнюються шляхи, знайдені різними мурахами. Останній етап включає оновлення рівнів феромонів на кожному ребрі графу.

Кожна мураха має скласти рішення для переміщення по графу. При виборі наступного ребра на шляху, мураха оцінює довжину кожного ребра, доступного з її поточного положення, а також відповідний рівень феромонів. На кожному кроці алгоритму мураха переходить зі стану x у стан y, що відповідає більш повному проміжному рішенню. Таким чином, кожна мураха k обчислює $A_k(x)$ набір набір можливих розширень до поточного стану в кожній

окремій ітерації та переходить в одне з них.

Для мурахи k імовірність переходу p_{xy}^{k} зі стану x у стан y є поєднанням двох значень – привабливості η_{xy} що обчислюється певними евристичними ознаками, що вказують на апріорне бажання цього руху та рівень сліду τ_{xy} шляху, вказуючи на те, наскільки доречним було в минулому зробити цей конкретний крок. Рівень стежки являє собою апостеріорну вказівку на бажаність цього ходу.

Загалом ймовірність того, що мураха k перейде зі стану x в стан y, обчислюється за формулою, показаною на рисунку 3.2.

$$p_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum_{z \in allowed_x} (\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}.$$

Рисунок 3.2 – Вірогідність переходу мурахи

Перший створений алгоритм отримав назву "мурашина система" і був розроблений для вирішення проблеми комівояжера[10] (рис. 3.3). Основна мета цієї проблеми полягає в тому, щоб знайти найкоротший шлях для з'єднання послідовності міст[11].

Загальний алгоритм є досить простим і базується на групі мурах, кожна з яких виконує один із можливих маршрутів між містами. На кожному етапі мураха обирає місто для переходу згідно з наступними правилами:

— Кожне місто повинно бути відвідане лише один раз.

— Віддаленіші міста мають менші шанси бути обраними (менша видимість).

— Шляхи з більш сильним феромоновим слідом між двома містами мають більшу ймовірність вибору.

— Після завершення подорожі мураха відкладає більше феромонів на всіх ребрах, якщо шлях був коротший.

— Після кожної ітерації феромоновий слід випаровується з певною мірою.

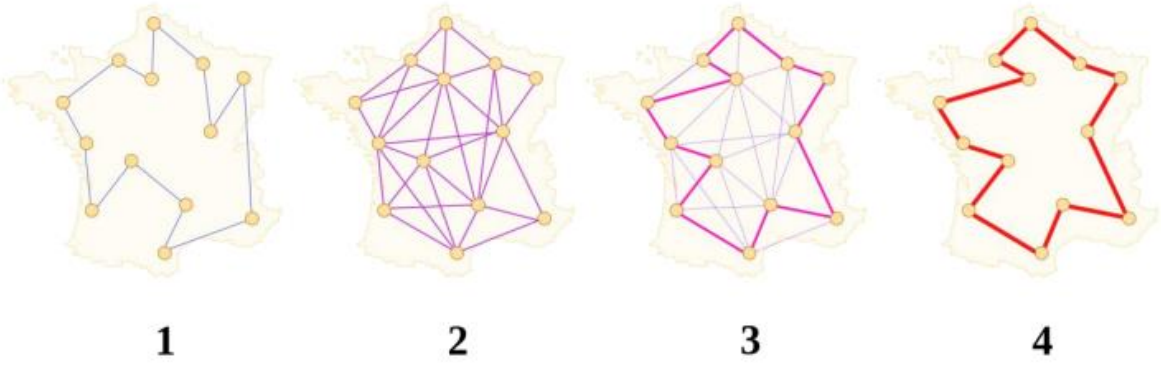

Рисунок 3.3 – Рішення задачі комівояжера мурашиним алгоритмом

3.2.2 Варіанти мурашиних алгоритмів

Мурашиний алгоритм є класом оптимізаційних алгоритмів, а варіацій у нього може бути багато. Основні варіанти та модифікації мурашиного алгоритму включають:

— Мурашиний систематичний пошук (Ant System) представляє собою фундаментальний варіант мурашиного алгоритму. У цьому підході мурахи визначають свій шлях, обираючи крок на основі рівня феромонів та інших важливих факторів. Цей метод дозволяє мурахам взаємодіяти з оточуючим середовищем, при цьому керуючись важливими даними, такими як концентрація феромонів на різних шляхах та інші параметри.

— Мурашиний колоніальний оптимізатор (Ant Colony Optimization - ACO). Розширена версія мурашиного алгоритму, яка використовує принципи організації роботи мурашиних колоній у природі[12].

— Мурашиний систематичний пошук з локальним оновленням феромонів (Ant System with Local Pheromone Update) представляє собою ефективну модифікацію мурашиного алгоритму, де кожна мурашина колонія взаємодіє з феромонами на локальному рівні, враховуючи навколишнє середовище. Основний принцип полягає в тому, що кожна мураха, крім

врахування глобальної інформації про феромони, також взаємодіє з феромонами, які оточують її поточне положення.

— Мурашиний імператорський алгоритм (Elitist Ant System) представляє собою вдосконалену версію базового мурашиного алгоритму, де вводиться новий елемент – "еліти" або мурашині імператори[13]. Ці елітні мурахи мають важливу роль у координації зусиль інших мур для ефективного вирішення завдань оптимізації. Вони визначають напрямок руху та впливають на вибір шляхів, граючи ключову роль у забезпеченні збільшеної збіжності та вдосконалення результатів алгоритму.

— Мурашиний ансамбль (Ant Ensemble) є модифікацією мурашиного алгоритму, яка використовує концепцію паралельної роботи декількох мурахиних агентів. Кожен мурашиний агент працює над вирішенням задачі оптимізації незалежно один від одного. Після завершення своєї роботи всі агенти об'єднують свої результати, щоб сформувати загальне рішення. Цей підхід сприяє більш швидкому та ефективному пошуку оптимального рішення шляхом використання колективної експертизи мурашиного ансамблю.

— Мурашиний алгоритм з динамічним оновленням феромонів (Dynamic Ant Colony System) представляє собою розвинену модифікацію базового мурашиного алгоритму, яка активно реагує на динамічні зміни в середовищі. Цей підхід автоматично адаптує параметри алгоритму та враховує змінюючіся умови навколишнього середовища. Відмітним аспектом є можливість динамічної зміни значень феромонів з плином часу, що дозволяє мурахам адекватно реагувати на зміни та забезпечує стійкість алгоритму в різноманітних умовах. Такий підхід покращує ефективність та надійність алгоритму, роблячи його більш гнучким та придатним до застосування в різних сценаріях оптимізації.

— Мурашиний алгоритм з мультиферомонами (Multi-Phenomena Ant Colony Optimization) є інноваційною модифікацією базового мурашиного алгоритму, яка впроваджує концепцію різних видів феромонів для різних аспектів вирішуваної проблеми. Цей підхід дозволяє мурахам використовувати

різні типи феромонів для різних аспектів задачі оптимізації, що призводить до покращення ефективності та точності отриманих рішень. Застосування мультиферомонів дозволяє алгоритму краще враховувати різноманітні вимоги та обмеження, які можуть виникати у вирішуванні реальних завдань оптимізації. Такий підхід є важливим кроком у напрямку підвищення універсальності та адаптивності мурашиного алгоритму.

— Мурашиний алгоритм з врахуванням динаміки (Ant Colony Optimization with Dynamic Environment) представляє собою розширений варіант мурашиного алгоритму, який демонструє високу адаптивність до змін у вирішуваній задачі. Цей підхід враховує динамічні зміни в середовищі або графі задачі та автоматично адаптує параметри алгоритму для оптимального вирішення нових умов. Завдяки цій властивості, мурашиний алгоритм може ефективно пристосовуватися до змінюючихся обставин, що може бути особливо важливим у реальних сценаріях, де умови оптимізації можуть змінюватися з часом. Такий підхід підсилює гнучкість та універсальність мурашиного алгоритму в різних областях застосування.

— — Ранговий мурашиний алгоритм (Rank-based Ant Algorithm). Цей варіант мурашиного алгоритму визначає ранг для кожної мурасі в залежності від якості знайденого маршруту. Мурахи з вищим рангом мають більший вплив на вибір маршрутів, сприяючи ефективнішому використанню інформації від найкращих рішень. Це сприяє покращенню збіжності алгоритму та забезпечує більш ефективний пошук оптимальних рішень.

— Мурашиний алгоритм з паралельним виконанням (Parallel Ant Algorithm). У цьому варіанті кілька мурах може працювати паралельно, що значно покращує швидкодію алгоритму. Кожна мураха спеціалізується на вирішенні своєї частини проблеми, і їхні результати можуть об'єднуватися для отримання загального рішення. Цей підхід дозволяє використовувати паралельні обчислення для більш ефективного вирішення складних завдань та забезпечує прискорення збірження алгоритму.

— Мурашиний алгоритм з обмеженнями на використання феромонів (Constrained Ant Algorithm). У цій модифікації мурахам накладають обмеження на кількість феромону, яку вони можуть використовувати під час вибору шляху. Ця інновація може стимулювати більш ефективне використання інформації про феромони та підсилювати конкуренцію між мурахами. Обмеження допомагає уникнути перенасичення феромонами і сприяє більш адаптивному та збалансованому вибору шляхів у графі.

— Мурашиний алгоритм з адаптивною інтенсивністю випаровування (Adaptive Evaporation Ant Algorithm). Ця модифікація включає динамічну зміну інтенсивності випаровування феромону. Зазвичай, на етапах, коли алгоритм тільки починає свою роботу або коли знайдено кращі рішення, інтенсивність випаровування може бути зниженою. Це допомагає зберігати цінну інформацію у вигляді феромонів на шляхах, які призводять до потенційно оптимальних рішень. Навпаки, коли розв'язок не змінюється протягом деякої кількості ітерацій, інтенсивність може збільшуватися для поліпшення збіжності та стимулювання додаткового пошуку в інших напрямках.

— Мурашиний алгоритм з кластеризацією мурах (Ant Algorithm with Ant Clustering). В даній модифікації можливо розділити мурах на кластери, де кожен кластер спеціалізується на певних областях графу або вирішує свою частину задачі. Це сприяє ефективнішому розподілу ресурсів та концентрації уваги мур у кожному кластері на конкретних аспектах проблеми. Періодичний обмін інформацією між кластерами дозволяє мурахам користуватися колективним досвідом, сприяючи загальній ефективності алгоритму в пошуку оптимального рішення.

— Мурашиний алгоритм зі змінною кількістю мурах (Ant Algorithm with Variable Ants). Це варіація алгоритму, де кількість мур може динамічно змінюватися в залежності від умов задачі чи обчислювальних ресурсів. Такий підхід дозволяє ефективно використовувати наявні ресурси та адаптувати інтенсивність пошуку до конкретних умов вирішуваної задачі. Гнучкість

управління кількістю мур дозволяє оптимізувати роботу алгоритму під різноманітні вхідні умови, сприяючи його ефективності та швидкодії.

— Мурашиний алгоритм для неповних графів (Ant Algorithm for Incomplete Graphs). Ця модифікація передбачає роботу алгоритму в умовах, коли граф, представляючи задачу, не є повністю з'єднаним. У таких ситуаціях мурахам доводиться адаптуватися до відсутності деяких зв'язків між вузлами та знаходити оптимальні шляхи в обмежених або змінених умовах. Ця модифікація дозволяє мурахам ефективно оптимізувати маршрути в умовах, коли не всі можливі зв'язки між точками доступні, розширюючи сферу застосування алгоритму.

— Мурашиний алгоритм зі змінною стратегією вибору шляху. Кожна мураха може володіти декількома стратегіями вибору шляху. Наприклад, одна стратегія може бути спрямована на експлорацію (вивчення нових шляхів), інша - на експлуатацію (використання вже відомих шляхів). Мурахи можуть динамічно змінювати свою стратегію в залежності від обставин.

— Мурашиний алгоритм зі змінною стратегією вибору шляху (Ant Algorithm with Adaptive Path Selection Strategy). У цьому варіанті кожна мураха обладнана декількома стратегіями вибору шляху, які можуть динамічно змінюватися в залежності від обставин. Наприклад, одна стратегія може бути спрямована на експлорацію, стимулюючи мураху вивчати нові шляхи, тоді як інша стратегія може підтримувати експлуатацію, спрямовану на використання вже відомих шляхів. Ця гнучкість дозволяє мурахам більш ефективно адаптуватися до змін у середовищі та умовах завдання, забезпечуючи оптимальний вибір стратегії для кожного конкретного випадку.

— Мурашиний алгоритм для задачі про рюкзак (Ant Colony Optimization for Knapsack Problem - ACOKP) - це спеціалізована модифікація алгоритму мурашиного вибору, призначена для ефективного вирішення задачі про рюкзак. Задача полягає в виборі підмножини предметів з різними вагами та вартостями з метою максимізації загальної вартості при дотриманні визначеного обмеження на вагу. ACOKP використовує популяцію мур для

конструювання рішень шляхом ймовірнісного вибору предметів на основі знань, закодованих у феромонах. Кожна мураха приймає рішення, враховуючи компроміс між вартістю та вагою предметів, і їхні вибори впливають на конструкцію кандидатських рішень. Алгоритм динамічно оновлює рівні феромонів, дозволяючи мурахам вчитися з успіху своїх попередніх рішень. Ця адаптивність покращує збіжність алгоритму до оптимальних рішень протягом послідовних ітерацій. АСОКР може бути додатково налаштований для врахування додаткових обмежень у задачі про рюкзак, таких як обмеження об'єму, шляхом модифікації стратегій вибору предметів та оновлення феромонів. Його ітеративний характер та ймовірнісне прийняття рішень роблять його ефективним засобом для вирішення складних задач про рюкзак.

4 МОДИФІКАЦІЯ МУРАШИНОГО АЛГОРИТМУ

4.1 Варіанти модифікації елементів мурашиного алгоритму

Для модифікації було обрано стандартну мурашину колоніальну оптимізацію. Адже всі інші варіанти так само є модифікаціями, які за необідності, або з метою покращення алгоритму так само можна буде запровадити.

Основна мета модифікації – виконання пунктів поставлених у постановці задачі:

— Додвання параметра балів за вершину.

— Додавання параметру довжини маршруту, яку не можна перевищувати.

— Вибір маршруту обмеженої довжини з максимальною кількістю балів за відвідані вершини.

— Зміна критерія вибору наступного кроку з урахуванням балів за вершині і відстані до неї

— Максимізаця суми балів за відвідані вершини, замість мінімізації довжини маршруту, з урахуванням обмеження цієї довжини.

Для їх реалізації було визначено декілька ключових елементів алгоритму, які необхідно розглянути та модифікувати:

— Критерії вибору наступного кроку

— Обмеження довжини маршруту

— Обирання критерію вибору найкращого маршруту

— Вибір найкращого маршрут

— Критерій розподілення феромону

— Підрахунок балів

Реалізація кожного цього пункту може включати в себе комплекс функції, або навпаки, реалізовуватися за рахунок модифікацій різних частин алгоритму.

Ось деякі етапи, які можуть бути враховані при модифікації мурашиного алгоритму:

— Визначення ваги ребер. Вага кожного ребра (дистанції між КП) може визначатися не тільки фізичною довжиною шляху, але і вартістю відвідування КП.

— Феромони та імітація мурах. Феромони можуть представляти вартість маршруту, яку мураха отримує, відвідавши КП.

— Мурахи можуть вибирати ребра на основі комбінації ваги ребра і кількості феромону на ньому.

— Обмеження на довжину маршруту. Додавання обмеження на довжину маршруту, перш ніж мураха обере новий КП. Якщо додавання КП порушує це обмеження, мураха може шукати інші варіанти.

— Оновлення феромонів. Після кожного кроку можна оновлювати феромони, враховуючи отримані бали відвіданими КП.

— Максимізація вартості. Зміна алгоритму так, щоб максимізувати вартість маршруту, а не мінімізувати відстані.

Але кожен з пунктів потребує детального розгляду для визначення найкращого варіанту.

4.1.1 Критерії вибору наступного кроку.

Це є вхідні дані до алгоритму, які будуть визначати всю подальшу логіку алгоритму. В залежності від них можна налаштувати його під виконання різних задач. В оригіналі це є відстань між вершинами, в той час як в цій задачі додається певна кількість балів за кожну відвідану вершину, яку також необхідно врахувати.

Найпростішим варіантом є класичний вибір наступного кроку за відстанню. Але за такого випадку втрачається головний сенс змагань, де необхідно максимізувати кількість балів, бо цей параметр ніяк не враховується.

Другим варіантом як раз міг би бути вибір за вартістю КП. Але за орієнтування лише на кількість балів можлива втрата варіантів, де два дешевших КП буде вигідніше взяти, ніж робити один довгий перехід до дорожчого, або рівнозначна вірогідність при двох КП на дуже різних відстанях але однакової вартості.

Орієнтований граф. Це варіант переходу від врахування окремо відстані та балів до одного параметра. Він враховуватиме значення балів за кілометри, і буде відповідно відношенням вартості КП до відстані яку до нього треба подолати до наступного КП. Це є найбільш наближеним до реальних умов варіантом, бо при русі між двома КП, долаючи однакову відстань в різних напрямках за наступний крок ми отримаємо різну кількість балів при русі між тими самими КП. В кільцевому варіанті це дасть ту саму кількість балів, але може повпливати на вибір кращого варіанту наступного кроку від час самого виконання алгоритму. Але це викликає певні складнощі, оскільки необхідна точності в алгоритмі, інакше буде неправильно враховуватися вірогідність переходу. Також це потребуватиме додаткових розрахунків.

Не орієнтований граф, який враховує обидва параметри. Варіант приведення вартості КП та відстані до однієї змінної. Можливо встановити вартість переходу як суму балів за КП, розділену на відстань між ними. Це значно спростило б обчислення, за рахунок однакової вартості руху в обох напрямках. Але такі умови не відповідають дійсності, бо вартість одного КП впливала б на вартість вибору переходу до іншого. Також це означало б врахування вартості кожного КП двічі за маршрут. Ці теорії було перевірено на практиці на моделі графу та підверджено що вони не дають бажаний результат.

Додавання третього параметру в функцію вибору наступного кроку. В оригінальному варіанті перехід визначається параметрами феромону та відстані, із зведенням у певний степінь що означає важливість відповідного параметру. До цих двох параметрів можна було б додати ціну КП. Такий варіант додасть можливість вручну встановлювати важливість кожного параметру. Але водночас це ускладнить і так велику залежність якості

виконання алгоритму від правильного підбору параметрів. Також це є втручання в саму логіку алгоритму, тож є перспективним варіантом, але потребує детальнішого дослідження та перевірки на практиці.

4.1.2 Обмеження довжини маршруту

Це обмеження є ключовим у задачі з рогейну, і якого немає в задачі комівояжера, для якого створений класичний алгоритм мурашиной колонії. Зазвичай мапи створюються так що неможливо за відведений час зібрати усі КП. Через це спортсменам необхідно планувати яку відстань за ці змагання вони встигнуть пробігти. Це необхідно, оскільки кожен рогейн може мати різну тривалість, а кожна команда спортсменів розраховує на свої власні сили. В тому числі і на кожній тривалості змагань. Якщо трьох годинний рогейн спортсмени ще можуть бігти з високою інтенсивністю, то на 24 години треба економити сили.

Якщо алгоритм просто змусти розвертатись після досягнення половини дистанції, це може не дати бажаного результату, оскільки може не бути підходящих переходів для того що б досягнути початкової вершини. Це можна реалізувати відслідковуючи довжину поточного маршруту, та забороняти перехід до КП, якщо відстань від нього до фінішу буде більша за відстань що ще можна пройти. Це можна втілити у функції яка розраховує бажання переходу мурахи з поточної до всіх інших, додавши просту перевірку цією умови. Через те що мурахи стартують кожного разу з різних вершин може виникнути інша складність: фініш може не портапляти в оптимальну дистанцію. Мурахи запускаються з кожної вершини, а от же він гарантовано входитиме в маршрут лише у цьому випадку. А це означає що багато маршрутів будуть взагалі не входити до пошуку оптимального маршруту. Одним із варіантів є прирівнювання його до КП і додавання високої вартості, що б ця точка була привабливішою для алгоритму, та з більшою вірогідністю потрапляла в маршрут. Це може спрацювати тому що зазвичай навколо фінішу

розташовані дешеві КП, близько 3 балів, то ж присудивши йому, наприклад, 10 вірогідність буде значно більшою.

4.1.3 Обирання критерію вибору найкращого маршруту

Цей параметр відповідає в першу чергу за кінцевий результат. Але в процесі пошуку, саме він відповідає наскільки якісно будуть сортуватися створені маршрути та визначатися кращий серед них.

Першим варіантом є сумування очок за маршрут. Гарний варіант, бо в кінцевому результаті, для точності, бали за кожен маршрут будуть сумуватися і буде проводитися порівняння між ними. Так само це є основним критерієм найкращого маршруту. Але постає питання при виборі між декількома маршрутами різної довжини. Він може задовольняти встановленому обмеженню по дистанції, але при рівній кількості балів у команд звертається увага на час подолання дистанції. І відповідно та команда яка завершить маршрут раніше посяде вище місце. І не можна забувати що самим спортсменам буде простіше фізично подолати коротшу дистанцію. Окрім того, на деяких змаганнях, в яких місцевість не дозволяє створити карту з достатньою дистанцією для такого часу організатори після фінішу пропонують другу карту, зі значно меншою кількістю КП, але яка відповідно дозволяє отримати ще трохи балів. В такому випадку може додатися другорядне порівняння маршрутів за довжиною. Це буде важливо так само для алгоритму, бо якщо дистанція буде завелика, то обмеження на вартість маршруту не буде ніяким чином призводити до пошуку оптимальних переходів.

Іншим варіантом може слугувати щільність маршруту. Це параметр який буде поєднувати в собі як раз обидва значення: кількості балів та дистанції. Для того що б її розрахувати нам в будь якому випадку необхідно спочатку порахувати суму балів за маршрут і його довжину. Потім відповідно знаходити щільність для кожного конкретного маршруту. Якщо ж використовувати в якості вхідних даних матрицю щільності, одразу до сумування кожного з цих

відрізків перейти не вийде, бо сума цих цифр не відображає дійсності в умовах всього графу, і при русі одним і тим же маршрутом в різні сторони буде різною. Однак за першого варіанту все ще будуть певні нюанси, адже щільність може бути однаковою за різних довжин маршруту, але кількість балів може сильно відрізнятись. Це особливо може бути помітно на мапах різної дистанції. Це доволі гарний загальний показник маршруту, який дає ясність при бажанні порівняти маршрути спортсменів: перші, значно сильніші спортсмени могли пробігти дистанцію не сильно зважаючи на оптимальність маршруту, а лише за рахунок фізичної сили, а інша команда може побудувати значно ефективніший маршрут тої ж вартості, але в результаті витратить на це менше зусиль. Також це гарний показник при виборі яку зону з КП відвідувати, яка з частин мапи дасть більше балів за меншу витрату зусиль. Це все відноситься до планування та аналітики в реальних умовах. За умови алгоритму цей параметр може розраховуватися як поточно, так і при порівнянні, з двох інших параметрів. Він хіба що може допомогти уникнути зайвого циклу при кінцевому розрахунку. Але в будь якому випадку цей параметр має використовуватися як допоміжний до суми балів за маршрут, а от же визначення між двома варіантами може бути хіба при бажанні оптимізувати швидкість виконання алгоритму.

4.1.4 Підрахунок балів

Підрахунок балів, відповідно до попереднього розділу буде виконуватися саме сумуванням кількості балів за маршрут. Але сам розрахунок може виконуватися двома варіантами. Створюватися або постфактум, при необхідності використати цей параметр, на основі даних про створений маршрут, так як цей список я основою алгоритму про пройдений мурахою шлях. Або поточно, в додатковий масив даних з довжинами всіх маршрутів що прокладаються. І хоча одним із варіантів було прирівняння фінішу до КП з великою вартістю, підрахунок балів не має його враховувати. Адже навіть при порівняння це буде спотворювати вибір найкращого маршруту в подальшому,

кількість відкладання феромону, яка буде базуватися на цьому параметрі, та реальну кількість білів яку наберуть спортсмени за маршрут, адже в кінцевому результаті цей резутьтат необхідно буде вивести.

4.1.5 Вибір найкращого маршруту

Головне завдання рогейну – зібрати якомога більше балів. Це і є основним критеріем. Після побудови усіх маршрутів на поточній ітерації має зберегтися найкращий зі створених маршрутів. Але оскільки не всі маршрути можуть містити фініш, найкраща послідовність обов’язково має враховувати це. При порівнянні маршрутів в кінці кожної ітерації пошук має виконуватися лише серед маршрутів, які містять фініш, незважаючи на те що інші маршрути можуть мати більшу кількість балів. А це швидше за все ставатиметься, оскільки при підрахунку балів фініш нічого не вартує. Це може виконуватися відбором у циклі, або запитом на сортування. Це буде залежати від подальших вимог реалізації. Але далі сама серед цієї відібраної частки маршрутів буде обиратися найдорожчий маршрут та порівнюватися зі збереженим найдорожчим маршрутом. Другорядним критерієм має виступати довжина маршруту, оскільки переваги такого підходу, з додатковим параметром порівняння маршруту вже були описані. Він має використовуватися лише за умови що маршрут має таку саму кількість балів.

4.1.6 Критерій розподілення феромону

Феромон у мурашиному алгоритмі є сигналом, залишеним мурахами на пройденому шляху. Це відзначення служить для визначення шляхів та обирання оптимальних маршрутів. Феромон функціонує як "пам'ять", утримуючи інформацію про пройдені шляхи та впливаючи на вибір мурахами майбутніх шляхів. Вплив феромону забезпечує колективне прийняття рішень мурахами, допомагаючи їм знаходити та обирати оптимальні маршрути в

умовах обмежень.

Тож правильний розподіл є ключовою задачею. У класичному варіанті кількість відкладання феромону залежала від пройденої мурахою відстані – чим коротший маршрут, тим більше феромону відкладалося. Оскільки мартиця була симетричною, однакова кількість феромону відкладалася на обох частинах від діагоналі матриці. Але оскільки у модифікованому варанті алгортиму, вибір наступного кроку відбувається вже не за впливом відстані, критерії відкладання феромону також має змінитися відповідно до критерію вибору найкращого маршруту.

Окрім того, за варіанту вибору критерія наступного кроку у вигляді орієнтованого графу має бути протестовано, чим має сенс відкладання феромону лише один раз, на відповідному напрямку переходу, оскільки в кінцевому результаті все одно маршрут вийде кільцевий. Це може призвести до пошуку більшого числа варіантів переходу, а от же до виходу з локальних оптимумів. Або ж навпаки, симетричне відкладання феромону пришвидшить приходження до найкращого варіанту за заданої довжини.

При відкладанні феромону алгоритм буде посилатися на маршрути, які мурахи побудували. Але існує два критичні моменти, які треба врахувати. Оскільки мурахи запускаються з різних вершин, фініш може потрапляти не до кожного маршруту. Друге, це те що більшість ітерацій будуть виконуватися без знаходження все кращих маршрутів. Хоча це і є ключем у методі знаходження ефективного маршруту. У класичному варіанті те що маршрут на ітерації не знайшов більш вигідну комбінацію не є проблемою, оскільки це має накопичувальний ефект за рахунок сумування відкладеного феромону. Але тут постає питання саме у поєднанні з першим фактором. Необхідно визначити як краще виконувати розподілення феромону за цих умов.

Якщо маршрут не буде включати в себе фініш, він все ще буде орієнтуватися на ті самі критерії обирання ефективних переходів що і маршрути з фінішем. Такі маршрути можуть допомагати знаходити стійкі ефективні переходи, або найкращі комбінації КП, які в подальшому можуть

використовуватися маршрутами що будуть включати фініш. Лише необхідно вирішити питання відкладання феромону на маршрутах цих двох типів. Можна розглянути варіант відкладання різної кількості феромону на маршрутах що включають фініш і що ні. Це може додати привабливості до маршрутів що включать фініш. Але враховуючи що вибір найкращого маршруту буде виконуватися лише серед маршрутів що включають фініш, і якщо при обиранні від буде враховуватися як КП, то можливо таке розподілення не має сенсу.

Друге питання можна вирішити за рахунок впровадження однієї з модифікацій класичного мурашиного алгоритму - рангового мурашиного алгоритму. Ранговий мурашиний алгоритм - це розширення базового мурашиного алгоритму, яке додає концепцію рангів для управління впливом мурах на вибір шляхів. Кожній мурасі присвоюється ранг в залежності від якості її знайденого маршруту. Мурахи, які знаходять більш оптимальні рішення, отримують вищі ранги. При виборі наступної вершини мураха враховує як феромон, так і ранги інших мурах, які вже пройшли цей шлях. Вершина з більшим рангом стає більш ймовірним вибором. Ранги мурах можуть змінюватися в залежності від якості їхніх маршрутів. Мурахи, які знаходять оптимальні рішення, підвищують свій ранг, що збільшує їх вплив на вибір маршруту для інших мурах. Деякі шляхи можуть бути визначені як критичні або важливі. Мурахи, які пройшли ці шляхи, отримують більший бонус для підвищення свого рангу. Феромони також оновлюються відповідно до рангів мур. Мурахи з вищим рангом залишають більше феромону, що підсилює їхній вплив на вибір маршруту. Ранговий мурашиний алгоритм дозволяє кращим мурахам більш ефективно впливати на вибір шляхів та сприяє збільшенню швидкості збіжності алгоритму до оптимального рішення. Тож можна скористатися головною відмінністю яка допоможе у вирішені другого питання, з розподілення феромонів на маршрутах різної оптимальності, відкладаючи тим більше феромону, чим дорожчий маршрут він знайшов. Класичний варіант виконує приблизно ту саму роль, але у значно менших масштабах, то ж це буде значне збільшення різниці і набагато суттєвіша

прибавка до кращих маршрутів. Вибір серед цих варіантів буде відбуватися на практиці. Бо з однієї сторони рангове відкладання феромону може пришвидшити сходження до оптимального варіанту, а з іншої ускладнить алогоритм додатковим циклом з сортування.

В класичному варіанті відкладання феромону відбувається за рахунок ділення константи на довжину маршруту. Чим коротший маршрут, тим відповідно менший дільник, тим більше в результаті феромону буде відкладено. Але за модифікованого варіанту, коли за вибір найкращого маршруту відповідає кількість балів, то це вже буде не зворотній показник. Один з варіантів, як раз інвертувати оригінальну формулу, і вже кількість феромону розділяти на константу. Це як раз один із елементів, саме підбір якого під конкретну задачу буде впливати на точність та швидкість алгоритму. Задля того що б врахувати дистанцію маршруту, що б вона так само мала вплив можна використати її замість константи. Таке співвідношення як раз буде демонструвати щільність маршруту, а от же співвідношення балів до довжини маршруту. Важливість того чи іншого параметру можна коригувати додаючи множник до відповідного параметра.

4.2 Елементи модифікації мурашиного алгоритму

Серед висунутих варіантів до кожного розділу необхідно обрати ті що будуть підлягати реалізації або порівнянню, тестуванню та обиранню серед них.

Для критеріїв вибору наступного кроку найбільш перспективними виглядають орієнтований граф або додавання третього параметр.

Орієнтований граф відображатиме щільність розподілення КП на мапі, а от же є реальним принципом за яким спортсмени можуть обирати варіанти комбінацій КП чи районів для відвідування, а саме відношення вартості КП до відстані, яку до нього треба подолати. Орієнтований граф лише змінює вхідні дані до алгоритму, які отримуються з все того ж оригінального графу. Він

використовує 2 коефіцієнти у виборі варіантів переходу, звівши їх до одного параметра. Водночас, їх вплив можна спробувати коригувати додаючи вагові коефіцієнти під час розрахунку матриці щільності.

Додавання третього параметру в функцію вибору наступного кроку теж має схожу дію. Буде враховуватися відстань від поточного КП до наступного і кількість балів за цільове КП. В цій функції важливість кожного параметру коригується показником степеня. І відповідно головним недоліком варіанту третього параметру є те що додається ще один коефіцієнт вдалий підбір якого буде впливати на результативність алгоритму. Але водночас це буде слідування початковій ідеї алгоритму де за вплив на показники відповідають саме показники степеня. Завдяки цьому можна буде підбирати коефіцієнти під кожну конкретну мапу окремо, оскільки вони можуть мати різний принцип побудови і розподілення КП на мапі. Якщо не буде дорогих КП, наприклад їх буде багато і вартість буде обмежуватися 6 балами, то навіть кожна «трійка» буде мати вплив на результативність. І саме ефективний маршрут, з більшою кількість КП, а відповідно коротшими переходами буде найбільш ефективним.

Ці два варіанти показують себе дуже перспективними, а от же кінцевий вибір між ними найкраще зробити реалізувавши обидва та зробивши порівняльний аналіз на швидкість сходження до оптимального варіанту.

Обмеження довжини маршруту буде реалізовано через додавання умови при визначенні вірогідності переходу до наступного КП. Створюється масив в який будуть поточно розраховуватися відстані маршрутів. При розрахунку вірогідності переходу виконуватиметься перевірка, чи буде відстань від цільового КП до фінішу менша, ніж дистанція що ще можна буде пройти.

Для більшої вірогідності вибору фінішу на маршрутах що не починаються з нього фініш буде прирівняно до КП з вартістю у 10 балів. Це буде задаватися про розрахунку матриці з усіма вартостями КП, і впливати на розрахунок матриці щільності. Результат цього буде враховуватися при розрахунку вірогідності переходу до наступного КП. Але коли буде

розраховуватися вартість маршруту, фініш як і має бути, буде коштувати 0 балів, бо це лише точка з якої починається та закінчується маршрут.

В якості критерію вибору найкращого маршруту вибирається кількість набраних балів за маршрут. Для його підрахунку у вхідні дані алгоритму буде додано список номерів КП, з якого і буде розраховуватися масив з вартостями КП у відповідних індексах. Цей параметр буде використовуватися при розрахунку орієнтованого графу, виборі найкращого маршруту і при розрахунку кількості феромону що необхідно відкласти.

Підрахунок балів за кожен маршрут буде виконуватися в функції вибору наступного кроку. Цей параметр буде розраховуватися в циклі вибору наступної вершини, відповідно додаючи кількість балів за обрану наступну вершину до масиву в якому за відповідним індексом буде зберігатися вартість кожного пройденого маршруту. Так як фініш прирівняно до КП, функція не має додавати бали за нього. А от же має пропускати додавання балів за цю вершину. Такий варіант вибрано до відповідного огляду у попередньому розділі що б уникнути аби уникнути викликання зайвого циклу розрахунку вартості коли цей параметр знадобиться.

Вибір найкращого маршруту відбувається після побудови маршрутів усіма мурахами. Має виконатися пошук маршрутів які містять фініш, а вже серед них має обратися маршрут з найбільшою кількістю балів. Найкращий результат попередніх ітерацій зберігається в змінну, з якою і відбувається порівняння кращого знайденого маршруту на даній ітерації. Для більш результативного пошуку має дожатися друга умова, яка буде порівнювати маршрути за довжиною, якщо вони мають рівну кількість балів, і обирати коротший маршрут, записуючі довжину, вартість маршруту і список КП як нові найкращі.

Способи відкладання феромону будуть відрізнятися в залежності від критерія вибору наступного кроку та бажаності збільшити вплив вартості маршруту чи його довжини. Це елемент ефективність якого буде залежати саме від правильного підбору до кількості феромону що буде відкладатися, оскільки

оригінальна концепція буде змінюватися. У випадку орієнтованого графу можуть розглянутися такі варіанти кількості відкладання феромону:

— Кількість балів, набрана за маршрут
— Різна кількість балів на маршрути з фінішем і без
— Щільність маршруту
— Щільність маршруту з ваговими коефіцієнтами
— Симетричне відкладання феромону
— Асиметричне відкладання феромону
— Обмеження мінімального значення феромону

Основною ідеєю при відкладанні феромону є перехід від константи діленої на довжину маршруту до кількості балів за пройдений маршрут. Далі вже можуть додаватися такі обмеження як додавання різних вагових коефіцієнтів на маршрути що включають фініш, і що ні. Або ділення цієї змінної на довжину маршруту, отримуючи щільність, і тим самим враховуючи важливість довжини маршруту. Сюди так само можна додати вагові коефіцієнти що б збільшити чи зменшити вплив того чи іншого параметру. Так сама необхідно перевірити сенс від симетричного чи асиметричного відкладання феромонів у матриці, бо з однієї сторони це має додавати ваги одному й тому самому переходу, а з іншої буде збільшувати вірогідність обирати саме маршрути що включають фініш.

У випадку вибору додавання третього параметру в критерії вибору наступного кроку неактуальним з перерахованих варіантів може вважатися хіба асиметричне відкладання феромону, так як цей критерій посилається на безпосередньо відстань між КП, а ця матриця є симетричною.

Для вибору між цими варіантам їх всі необхідно реалізувати та оцінити на практиці їх ефективність для швидкості знаходження оптимального маршруту.

5 РЕАЛІЗАЦІЯ ТА АНАЛІЗ РОЗРОБЛЕНИХ ФУНКЦІЇ

5.1 Вибір мови програмування

Головна задача роботи – це розробка алгоритму. Але для оцінки його роботи необхідна програмна реалізація, яка дасть можливість наочно демонструвати результати його роботи. Для цього буде використовуватися попередньо розроблений застосунок для операційної системи Windows, який був створений за допомогою мови програмування C#. Для розробки клієнтської частини застосунку було використано технологію Windows Forms. Це зручний фреймворк для створення інтерфейсів для застосунків призначених для Windows. Область застосування як раз передбачає особисте користування спортсменами саме для аналізу та тренування, а не безпосереднього використання на змаганнях. Комп’ютери дадуть зручніше переносити дані з мапи для застосунку а також зручніше зберігати та роздивлятися отриманий результат. Також поки що немає особливих вимог до швидкодії алгоритму, він лише має не перевищувати прийнятний для користувача час. Саме тому, для реалізації алгоритму було обрано ту ж саму мову що і у розробленого додатку, тобто C#. Загалом платформа .NET багато можливостей для майбутньої модифікації чи інтеграції даного алгоритму. Його можна буде портувати на мобільні пристрої чи створити веб сайт який надасть доступ до алгоритму у будь яких умовах. І все це доступно за допомогою фреймворків .NET

5.2 Опис робочого застосунку

Попередньо розроблений застосунок є важливою частиною алгоритму, бо саме через нього будуть вводитися та передаватися вхідні дані для роботи як класичного мурашиного алгоритму, так і його модифікації[14].

Для розробки клієнтської частини застосунку використано технологію Windows Forms — зручний фреймворк для створення інтерфейсів призначених

для Windows. Основним елементом є pictureBox, призначений для відображення зображень мап. Він має співвідношення сторін √2, що відповідає стандарту форми паперу ISO-216. Такий підхід спрощує завантаження мап у горизонтальному форматі, як часто використовується для друку на папері формату А. Розмір форми та інших елементів інтерфейсу адаптовані навколо pictureBox. Форма має фіксований мінімальний розмір для належного відображення елементів інтерфейсу та зручного використання ними. Щоб не зинакшувати розміри зображення і мати можливість роздивитися усю мапу було використано скроллбари, завдяки якім можна прокручувати зображення. Також форма підтримує повноекранний режим, залишаючи незмінним функціональний блок та збільшуючи розміри pictureBox.

На рисунку 5.1 зображено головну віконну форму, з вкладкою на якій елементи якої відповідають за основну логіку застосунку.

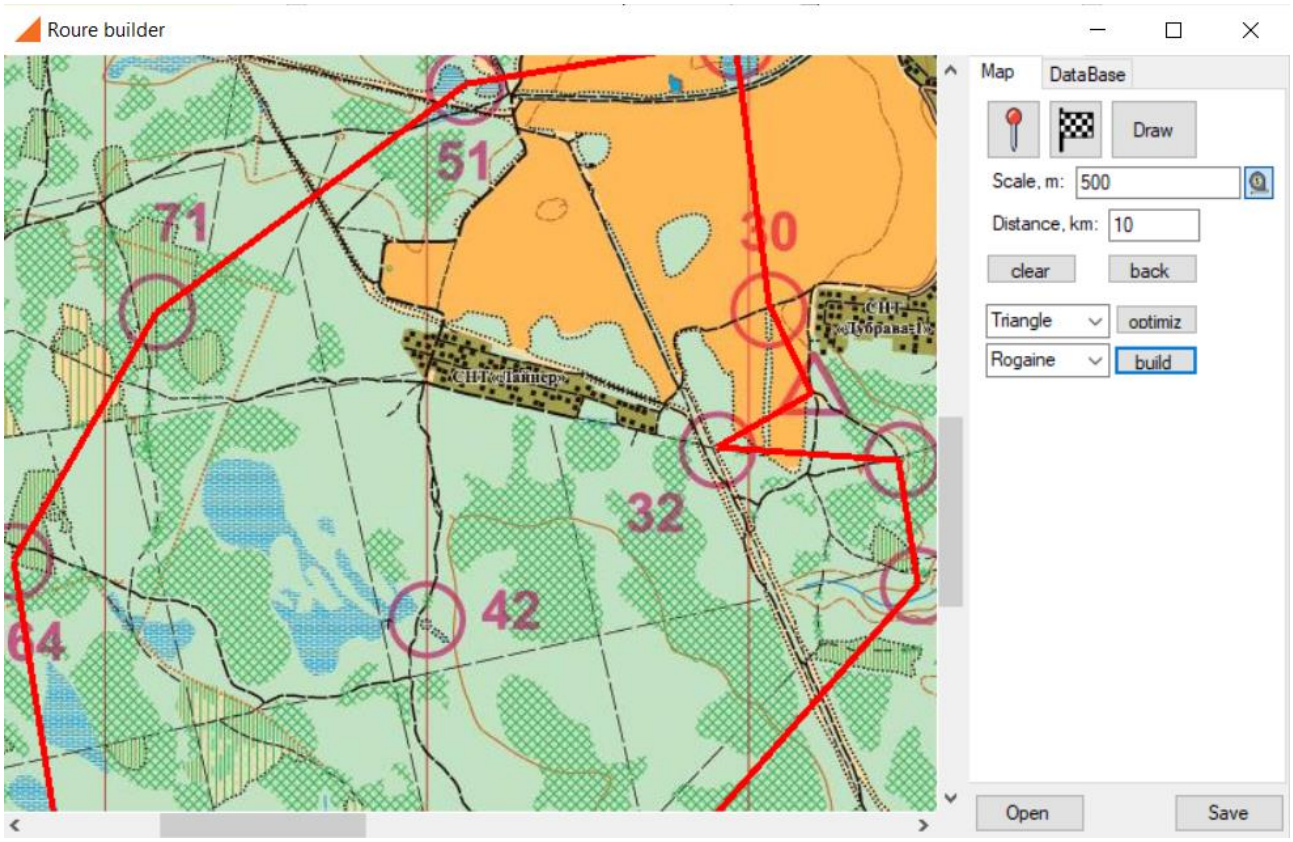

Рисунок 5.1 – Головна віконна форма з вкладкою побудови маршруту

В нижній частині головної форми розміщено дві кнопки: одна для

додавання зображення в застосунок, а інша для його збереження на комп’ютері. Кнопка "Open" відповідає за відкриття додаткової форми, яка дозволяє користувачу вибрати зображення на комп’ютері та завантажувати його в застосунок. Крім того, реалізовано зручний спосіб завантаження зображень через метод "Drag-and-drop", коли зображення можна просто перетягувати за допомогою миші безпосередньо у pictureBox. Кнопка "Save" відкриває іншу допоміжну форму, яка дозволяє користувачу зберегти зображення, відкрите у pictureBox, на своєму комп’ютері.

На головній формі також розташовано елемент TabControl, який дозволяє створювати вкладки для логічного розділення функціоналу та зручного перемикання між ними, завжди залишаючи у доступі зображення мапи. Це створює два функціональні блоки застосунку. Перша вкладка відповідає за основний функціонал та містить кнопки, до яких прив’язані функції основної логіки застосунку. У цьому функціональному блоку розташовані кнопки з піктограмами прапорця та піна, які, насправді, є чекбоксами. Це означає, що певна логіка буде виконуватися, доки чекбокс натиснутий. Кнопка з піктограмою піна відповідає за позначення контрольних точок на мапі. При натисканні активується функція, яка дозволяє користувачеві визначати КП на мапі, додаючи їх до списку вершин графа при кліку на зображенні. Після кліку на зображенні малюється червона точка, і відкривається віконна форма для введення номеру КП. Номер КП може бути двозначним або тризначним. Після натискання "Ок" або клавіші Enter віконна форма закривається, і вершина зберігається до графа.

Кнопка з прапорцем відповідає за встановлення точки, яка вказує на початок та кінець маршруту. При встановленні фінішу форми для введення номеру КП не відбувається, оскільки її було вирішено позначати як вершину з номером 1. Це суперечить логіці номерів КП, які зазвичай мають двозначний формат. Валідація номерів вкладена у форму введення, де перша цифра вказує вагу КП, а друга - порядковий номер, що дозволяє встановлювати декілька КП однакової ваги. Це рішення дозволяє програмі відрізнити фініш від КП,

незалежно від того, коли було позначено фініш і на якому місці в списку вершин він знаходиться. Коли отримано список усіх пов'язаних з фінішем вершин, починається побудова оптимального маршруту. Важливо відзначити, що на мапі можна позначити лише один фініш. Також додано логіку, яка дозволяє відміняти додавання КП, видаляючи точку з зображення та вершину з графа. Кнопка «optimize» містить біля себе combo box, в якому можна обрати варіанти оптимізації графу для видалення надлишкових ребер, що може пришвидшити роботу алгоритму. Кнопка «build» відповідає за виклик головної функції побудови маршруту. Біля неї розташовано combo box, на який можна додавати, а потім викликати різні функції побудови маршруту. В даному випадку це класичний та модифікований варіанти мурашиного алгоритму. Зображення рулетки, поле масштабу та поля параметрів маршруту виконують функцію для взаємодії з даними, введеними користувачем. При активації рулетки, яка є також чекбоксом, з'являється можливість відзначити елемент на мапі, масштаб якого користувач зможе вказати.. Для цього він повинен клікнути на зображенні у двох місцях для передачі точок з координатами. Між цими точками також автоматично малюється лінія. В разі помилки користувач може повторно вказати елемент на мапі. Дані з текстових полів передаються у функцію побудови маршруту при її запуску.

Також система виступає у ролі сховища, що дозволяє організовано зберігати мапи і розроблені маршрути для їх подальшого аналізу. В застосунку також Друга вкладка відповідає за роботу з базою даних, включаючи відображення змісту та виклик функцій. На даній вкладці реалізовані кнопки, які відповідають за функції збереження, відкриття, редагування та видалення записів. Для додавання зображення користувач повинен внести зображення у pictureBox, вказати назву мапи у текстовому полі, а після цього натискати кнопку збереження. Відкриття мапи відбувається шляхом виділення запису, який користувач бажає відкрити, та натисканням відповідної кнопки. Збережені зображення та назва відображаються у відповідних рядках. Якщо були збережені дані щодо контрольних точок або маршруту, вони конвертуються у

об'єкт графа за допомогою відповідних функцій. Для зміни запису необхідно спочатку його відкрити, після чого можна змінити назву або зображення у pictureBox і натиснути кнопку "Змінити". Відповідний запис у базі даних буде змінено. Для видалення запису необхідно виділити бажаний запис у DataGridView та натиснути кнопку "Видалити". Кнопка "Зберегти граф" відповідає за збереження графу на мапі. При виконанні дані графу конвертуються у рядки, які може зберігати база даних. Зі списку вершин створюються два рядки із значенням та координатами КП. Якщо було побудовано ребра, то їх довжини також зберігаються. Якщо ребра не існували, записується 0. Якщо був побудований маршрут, його порядок КП записується у вигляді послідовності номерів КП. Кнопка "Пошук" відповідає за виконання пошуку, при якому здійснюється пошук записів, що відповідають тексту, введеному в текстовому полі.

5.3 Реалізація алгоритму мурашиної колонії

Створення алгоритму для побудови маршрутів на рогейн має розпочатися з реалізації класичного алгоритму мурашиної колонії. Вже безпосередньо до нього будуть додаватися визначені модифікації. Власна реалізація краще дасть зрозуміти принцип роботи алгоритму, всі його нюанси, варіанти функцій та нюанси модифікації. Це є важливим кроком, оскільки дозволяє ретельно вивчити принципи роботи та внутрішні взаємодії, що лежать в основі алгоритму. Крім цього, для аналізу маршрутів обмеженої довжини буде корисно бачити оптимальний маршрут на всій мапі, так як він буде відображати найкращі варіанти переходів. Це надасть можливість для додаткової евристичної оцінки маршруту.

Перед безпосередньою реалізацією алгоритму було створено його блок-схему, що зображена на рисунку 5.1.

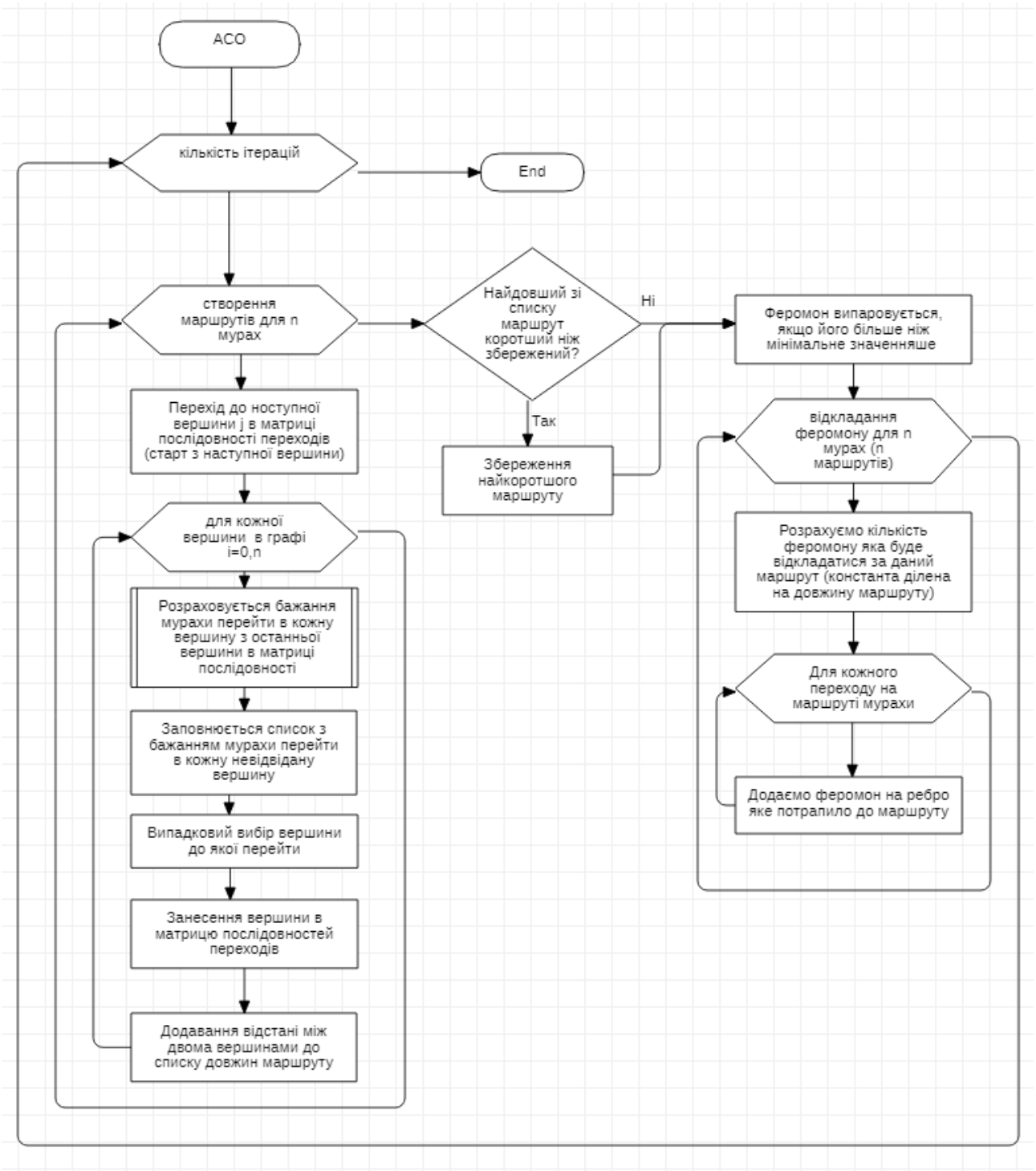

Рисунок 5.1 – Блок-схема алгоритму мурашиної колонії

Створення блок схем - особливо корисний підхід для таких складних алгоритмів як мурашиний, бо надає візуальне представлення логіки алгоритму. Це допомагає розуміти порядок виконання кожного етапу та взаємозв'язки між ними, структуру алгоритму, послідовність дій і визначати можливі шляхи оптимізації та покращення алгоритмів до їх програмної реалізації.

Алгоритм складається з 3 основних циклів. Перший виконується задану кількість разів, або до моменту виконання умови зупинки, яку можна додати. Другий цикл виконується для кожної мурахи, яких створюється за кількість вершин у графі. Це робиться для того що б на кожній ітерації маршрут мав початок у кожної вершини. Завдяки цьому забезпечується пошук ефективних переходів особливо для віддалених вершин. В цьому циклі ініціалізуються всі тимчасові змінні, в які записуються результати поточної мурахи, такі як список вершинам маршруту та довжина маршруту. Ці змінні мемоізуються у масив для кожної мурахи що б мати результат для порівняння після проходження всіх мурах. Надалі в циклі вже виконується безпосередня логіка пошуку оптимального маршруту однією мурахою, де розраховується бажання мурахи перейти в кожну невідвідану вершину з поточної вершини. Після цього обирається вершина, заноситься в список маршруту і додається дистанція до загальної довжини маршруту. Цей цикл виконується доки вся вершини не будуть відвідані. Це відбувається для кожної мурахи, і після того як кожна з них створила свій маршрут і дані для кожної збережені, порівнюється найкращий отриманий результат на цій ітерації з найкращим збереженим і оновлюється у випадку кращого результату. Після цього викликається оновлення феромону, який для початку випаровується, а потів відкладається у відповідності до довжини створеного маршруту, чим коротше тим більше, до ребер що потрапили у даний маршрут.

Також було вирішено впровадити одну з існуючих модифікацій мурашиного алгоритму – максмінний мурашиний алгоритм. Він вводить додаткові правила для управління рівнем феромону на ребрах графа. Ця модифікація призначена для поліпшення стійкості алгоритму та підвищення його здатності знаходження оптимальних рішень. Використано було обмеження на мінімальне значення феромону на ребрі[15], яке ребро не може перевищити під час етапу випаровування. Це обмеження введене для забезпечення стабільності і тривалості феромонного сліду. Воно впливає на швидкість випаровування феромону на ребрах графа та регулює, як швидко мурахи

можуть "забувати" про попередні рішення. Також забезпечує стійкість ребер графа, навіть якщо мурахи не використовують певний шлях протягом декількох ітерацій і допомагає утримувати інформацію про найкращі маршрути, забезпечуючи стійкість та уникнення занадто швидкого згасання феромону.

Вхідними даним для класичного алгоритму буде лише матриця відстаней у графі, так як він визначає лише найкоротший маршрут що охоплює всі вершини.

Після написання самого коду методом підбору було підібрано найкращі значення необхідних констант [16], а саме α, β, ρ, Q. Ще одним важливим параметром є кількість разів виконання основного циклу, але вона вже варіювалася заради експериментів та не була константою

Константа α відповідає за ваговий коефіцієнт впливу феромону при виборі наступного кроку мурахи. Вона враховує важливість феромону на ребрі при прийнятті рішення мурахою про вибір конкретного напрямку. Збільшення значення α призведе до більшої ваги феромону в ймовірності вибору маршруту, що може сприяти більшому врахуванню інформації від попередніх мурах. Константа β визначає важливість вартості ребра при наступного кроку мурахи. Збільшення значення β призведе до більшої ваги вартості в ймовірності вибору маршруту, що може сприяти врахуванню локальної інформації про близкість між двома містами. Константи α і β є значенням на яке зводяться відповідно значення феромону і довжина переходу між конкретними вершинами в функції розрахунку бажання мурахи перейти з поточного міста у всі невідвідувані вершини.

Константи Q та ρ відповідають за розподілення феромону. Константа ρ у відповідає за коефіцієнт випаровування феромону на ребрах графу. Ця величина визначає, яка частина феромону залишається на ребрі після кожного кроку, наскільки швидко феромон випаровується, а отже впливає на стійкість феромонного сліду. Чим більше значення ρ, тим швидше феромон випаровується, що призводить до більшої збору інформації про найкоротший маршрут в реальному часі. З іншого боку, менше значення ρ призведе до

повільного випаровування феромону, що дозволяє алгоритму враховувати попередні вибори мурах протягом більшого часу. Значення Q відповідає за кількість феромону яка буде відкладатися. Зазвичай, більше значення Q призводить до більшого внеску феромону, що може пришвидшити процес збіжності мурашиного алгоритму. На оптимальне значення Q впливає розмір графу. Для великих графів, де мурахи проходять багато ребер, може бути ефективно використовувати більше феромону, більше Q.

На рисунку 5.2 забражено приклад результату роботи алгоритму при погано підібраних константах.

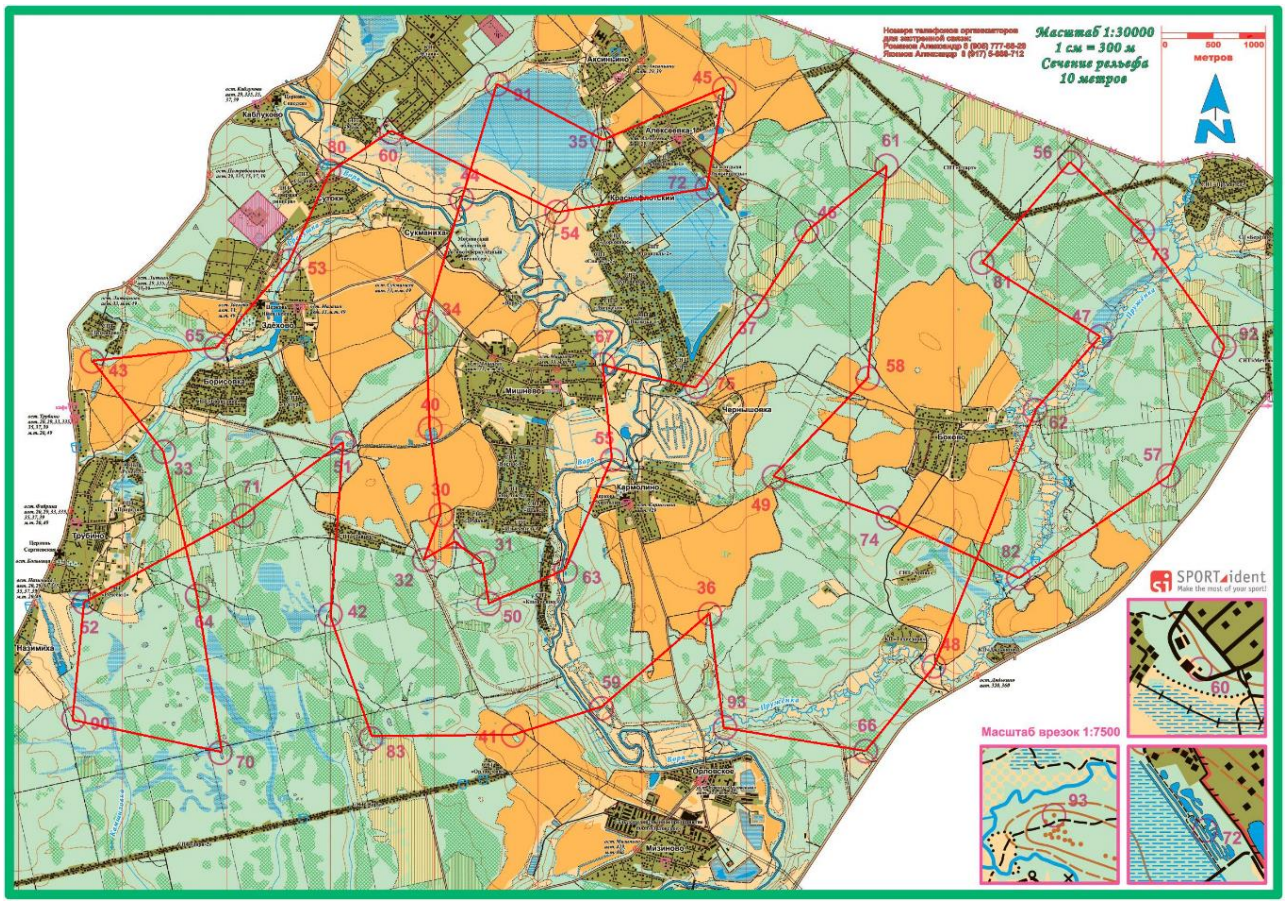

Рисунок 5.2 – Результат роботи алгоритму без підібраних констант

Дуже явною ознакою неефективності маршруту є перехрещення шляхів, оскільки дві діагоналі завжди будуть довші за два ребра, які утворилися б в наслідок обміну кінцевих точок таких переходів.

В застосунку реалізована функція розрахунку довжини маршруту в

кілометрах, яка могла б підрахувати довжину маршруту. Зроблено це в першу чергу для реалізації обмеження на максимальну довжину маршруту. Для перетворення масштабу мапи з метрів на віртуальний у пікселях на мапі позначається відрізок і вказується якій довжині у метрах він відповідає. Між цими точками також визначається відстані у пікселях, після чого розраховується масштаб у пікселях. В даному випадку та в подальших розрахунках і порівняннях буде використовуватися саме дистанція у пікселях. Це обумовлено тим що порівняння в кілометрах є менш точним, через ряд факторів. По-перше, відстань на зображенні мапи доведеться кожного разу зазначати вручну, що створює певну похибку. По-друге, саму відстань на мапі, окрім випадків з візуальним позначенням масштабу на мапі доведеться розраховувати вимірюючи певну відстань між об’єктами на фізичній мапі, або знайшовши карту в мережі і порівнюючи з реальними об’єктами, або спираючись на певні знання в побудові мап. По-третє, при побудові самої мапи можуть виникати похибки та неточності, в розташуванні об’єктів чи чомусь іншому. То ж для завдяки цій функції було встановлено довжину цього маршруту у 12281 піксель.

Шляхом підбору було визначено значення констант за яких алгоритм видавав найкоротші шляхи: $\alpha = 1$, $\beta = 4$, $\rho = 0{,}5$, $Q = 600$. Тестування проводилося на мапах з 25, 48 і 63 КП, з різною щільністю КП та їх розподіленням. Всі ці мапи будуть надалі використовуватися для аналізу модифікованого алгоритму з обмеженою максимальною довжиною маршруту.

Після підбору констант, перш за все вдалося позбутися перехрещень шляхів, що вже означало знаходження маршруту близького до оптимального. На рисунку 5.3 можна побачити найкращий отриманий результат роботи алгоритму з підібраними константами на тій самій мапі, на якій проводився підбір. Його довжина склала 10836 пікселі. Після цього алгоритм було перевірено і на інших мапах, що б впевнитися, що ці значення можна вважити універсальними для повного обходу графу на мапах з різною кількість КП і їх різною цільністю.

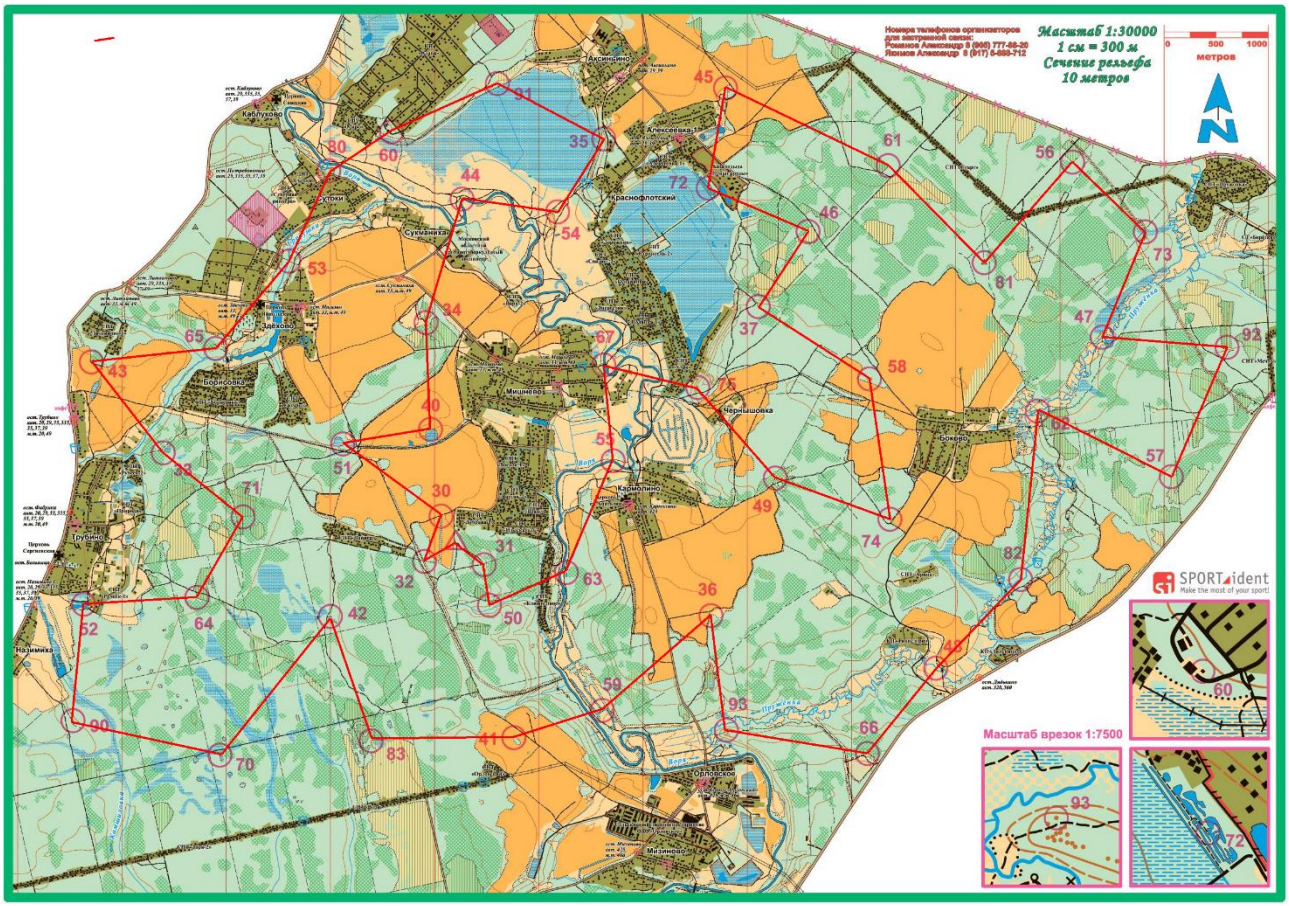

Рисунок 5.3 – Результат роботи алгоритму з підібраними константами

Через вірогідносний вибір алгоритм не завжди приходив до найкращого результату за відведений час та надану кількість ітерацій. Було проведено багато тестів і розраховано закономірність, по кількості ітерацій, необхідній для знаходження оптимального рішення. Алгоритм запускався з кроками 50, 100, 200, 500, 1000, 2000 ітерацій. Нажаль алгоритм рідко приходив до найкращого знайденого рішення, але достатньо оптимальний результат був знайдений вже після 50 кроків. Чим більше КП на мапі, тим складніше алгоритму прийти до оптимального рішення. Для цієї мапи це значення становило близько 10910 пікселів. Суттеві зміни іноді з’являлися лише на тисячному кроці. Але набагато більш привабливою ідеєю виглядає запуск алгоритму з невеликою кількістю кроків багато разів, ніж одноразовий запуск алгоритму з великою кількістю кроків.

Для кращого бачення було побудовано графік залежності часу від

кількості ітерацій алгоритму, зображений на рисунку 5.4. Тести було зроблено на 3 мапах. На ньому добре продемонстровано розраховану часову складність алгоритму $O(m * n^2)$, де m – це кількість ітерацій виконання алгоритму, а n – кількість вершин у графі. По осі x добре видно лінійну залежність часу на виконання від кількості ітерацій. А по осі y можна порівняти час необхідний на виконання алгоритму з 25, 48 і 63 КП, що демонструє приблизно квадратичну залежність від точок у графі.

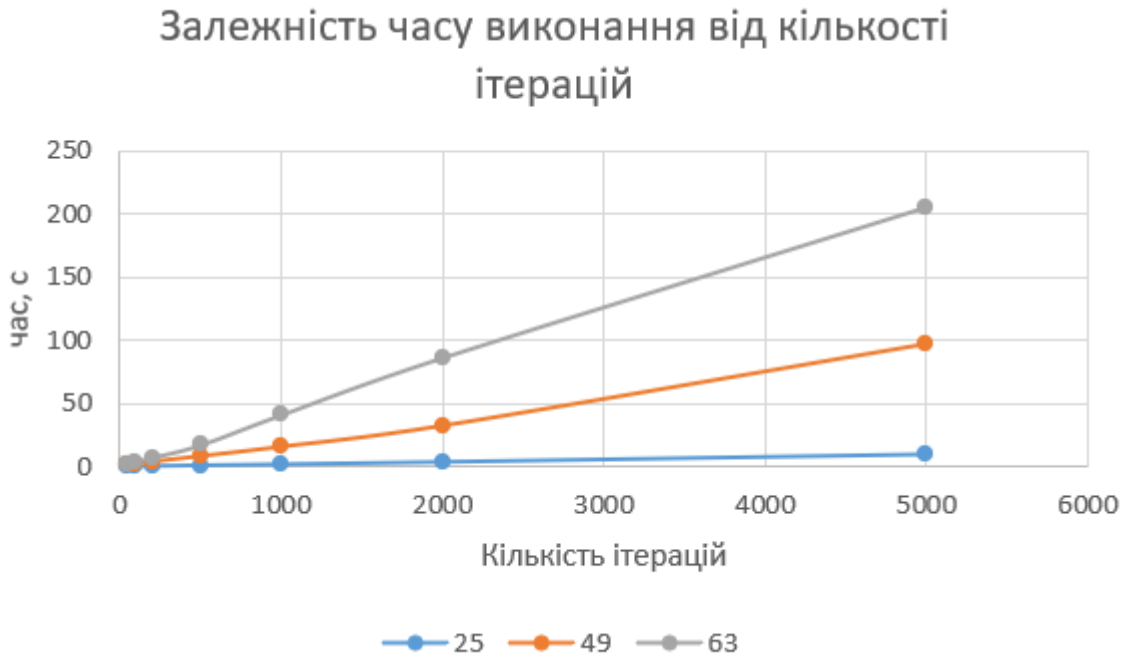

Рисунок 5.3 – Графік залежності часу виконання від кількості ітерацій

В попередньо розробленому застосунку була функція оптимізації графу, що видаляла неефективні переходи між КП. Неочікуваним виявилося те шо бажання оптимізувати граф, не давало працювати алгоритму. Він будував дуже короткі маршрути, замість охоплення усього графу. Аналіз показав, що іноді алгоритм заходив в глухий кут, коли в нього не залишалося варіантів переходу, зберігав результат, і так як умова пошуку вимагала саме найкоротший маршрут, видавала даний порядок КП за результат роботи алгоритму. Це є важливим питання для його подальшої модифікації, яка б включала вже поєднання з

реальними умовами мапи. Деякі переходи між КП на реальних змаганнях можуть бути взагалі неможливими, або дуже невигідними. Наприклад можуть бути встановлені заборонені зони, між ними можуть бути річки, які зазвичай неможна перепливати. Або ж просто болота чи густий ліс, які буде швидше обійти, подолавши більшу дистанцію ніж дертися наскрізь. Або принаймні взяти всі КП з однієї сторони, а потім перейти і взяти всі з іншої. Це питання буде актуальне і для маршруту на рогейн, де залежно від цього оптимальний маршрут може сильно змінюватися. Одним з рішень може бути інший спосіб введення даних, де переходу між двома КП можна буде або самотужки встановлювати значення, або коефіцієнт, який робитиме перехід значно менш привабливим.

5.4 Впровадження модифікацій до класичного алгоритму

— Орієнтований граф

— Додавання третього параметру в функцію вибору наступного кроку

— Обмеження довжини маршруту

— Прирівнювання фінішу до КП при розрахунку вірогідності переходу

— Розрахунок балів за кожну вершину

— Розрахунок балів за кожен маршрут

— Пошук найкращого маршруту серед маршрутів з фінішем

— Порівняння з кращим маршрутом та призначення змінних

— Відкладання феромону за кількістю набраних балів

Початковий крок у впровадженні модифікацій полягає у внесенні змін до блок-схеми алгоритму, щоб отримати чітке уявлення про загальний план модифікацій. Це допоможе визначити необхідні функції які конкретні функції вимагають модифікацій. Змінена блок-схема зображена на малюнку 5.4.

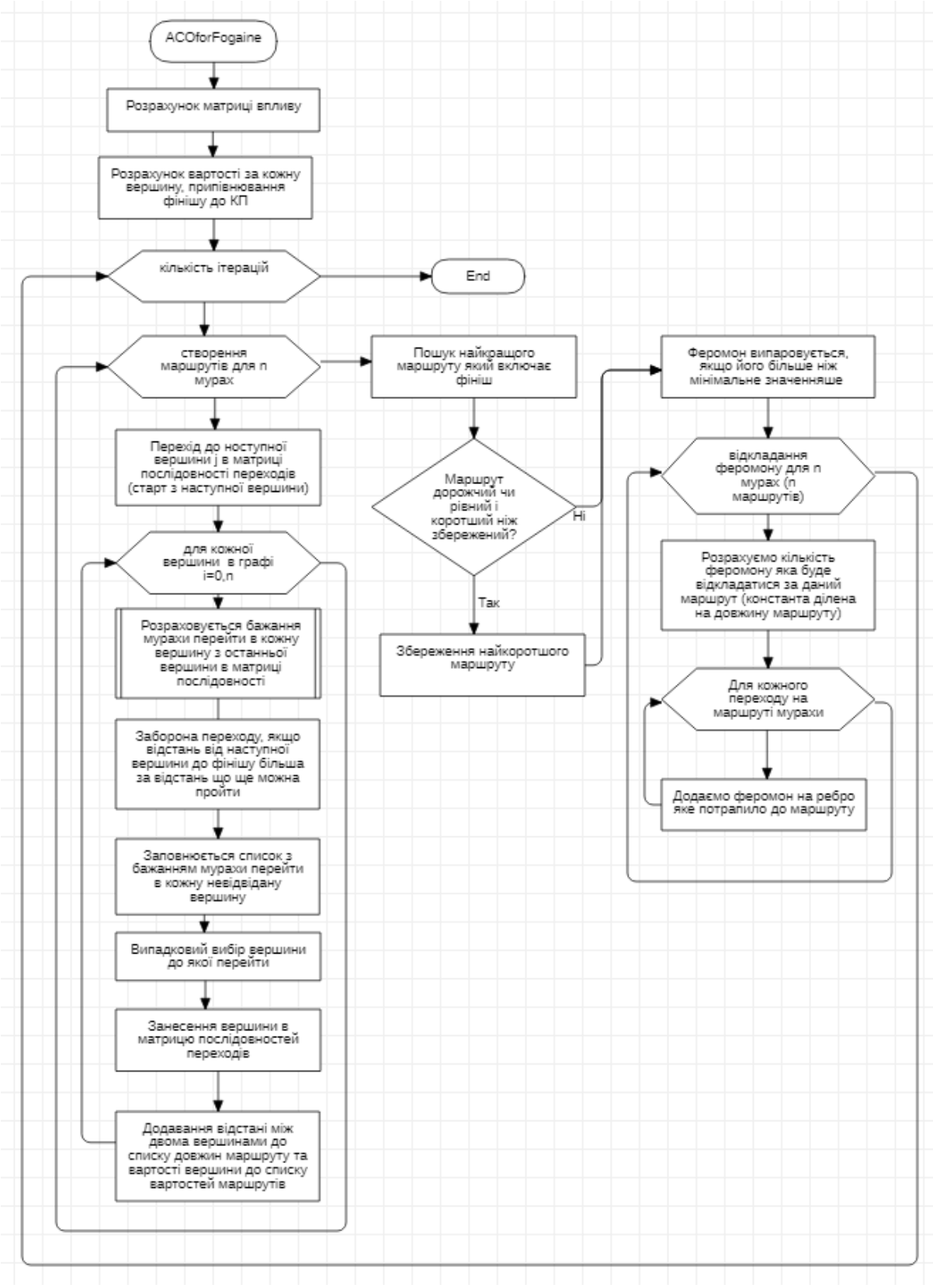

Рисунок 5.4 – Блок-схема модифікації алгоритму мурашиної колонії для пошуку маршруту для рогейну

Перша зі змін що відобразилися на блок-схемі це створення матриці впливу, яка буде представленням матриці на основі якої буде розраховуватись вірогідність переходу кожної мурахи. Для кожного варіанту алгоритму вона може представляти свої дані. В даному випадку буде розглядатися два варіанти: орієнтований граф, що представляє щільність шляхів, та матриця близькості вершин. Для розрахунку кожного варіанту буде створено відповідні функції. У випадку з матрицею щільності це відношення вартості КП до відстані що необхідно подолати до нього. Матриця близькості представляє з себе просто зворотне значення до відстані між вершинами. На неї ділиться якась константа для того що б зробити значення не таким малим.

В конструктор класу буде доданий ще один параметр, який представляє масив з номерами КП. В конструкторі ж з нього будуть розраховуватися вартість кожного КП у масив під відповідними індексами. В застосунку створена логіка завдяки якій фініш завжди позначається номером 1, завдяки чому його легко можна знайти при будь якому порядку його позначення. Його індекс зберігається в окрему змінну, що б не викликати функцію пошуку кожного разу коли знадобиться дія пов’язана з логікою його знаходження. Перш за все це призначення фінішу 10 балів в масиві ватростей КП що б зробити його більш привабливим для вибору алгоритмом. Крім цього створено ще один масив для кожного мурахи в який будуть сумуватися бали за пройдений маршрут.

Для виконання основного обмеження що накладає задача з рогейну, а саме обмеження максимальної дистанції у функцію розрахунку маршруту додано параметр довжини запланованої дистанції. Він далі буде передаватися в модифіковану функцію вибору наступного кроку, що викликається при виборі наступного міста для кожного мурахи, що буде забезпечувати відповідну логіку. Базова функція лише робила нульовою вірогідність переходу до вершин що є у списку. Тепер додалася ще одна перевірка на те чи буде відстань від вершини до якої можливий перехід до початкового КП більшим за відстань що ще можна пройти. Простими словами чи вистачить нам запасу дистанції якщо

ми будемо рухатися від стартового КП що б повернутися до нього. Якщо ні, то вірогідність переходу так само стає нульовою. В іншому випадку розраховується вірогідність переходу на основі феромону та відповідно обраної матриці впливу.

В цій функції мають відображення обрані варіанти критерія вибору наступного кроку. Орієнтований граф записаний в матрицю впливу, і просто змінює дані на основі яких розраховується вірогідність. Другим варіантом є додавання параметру вартості КП, залишивши початкову матрицю близькості. Йому так само додається показник степеня, що буде коригувати важливість цього параметру.

Розрахунок балів за маршрут буде виконуватися після вибору вершини до якої переходити шляхом додвавання вартості обраної вершини. Через те що фініш тепер теж має певне значення буде проводитися перевірка, і якщо індекс цільового КП відповідає фінішу, то балів додаватися не буде.

Після того як всі мурахи проклали маршрут серед них виконується пошук маршрутів які включають фініш, і вже серед них обирається маршрут який набрав найбільшу кількість балів. Індекс найкращого маршруту зберігається, після чого проводиться порівняння. Воно відбувається в 2 етапи задля отримання максимального результату. Першим етапом просто порівнюється кількість набраних балів зі збереженим попереднім найкращим результатом. За хибної умови перевіряється чи рівні маршрути за балами, і якщо так то порівнюється довжина нового маршруту. Зроблено це для того що б за будь яких обставин максимально оптимізувати шлях. Але особливо це важливо якщо планований маршрут значно більший ніж потрібна дистанція на мапі. В такому випадку, без обирання коротшого маршруту шлях може хаотично прокладатися по мапі і ні про яку оптимальність вже не може бути мови. У випадку проходження цієї комплексної умови проводиться оновлення даних про найкращий маршрут, дистанцію та кількість набраних балів.

Функція феромону, яку загалом можна виділити – це зменшення привабливості дальніх переходів за рахунок випаровування феромону. В

класичному алгоритмі на етапі відкладання феромону присутня одна з констант вдалий підбір якої відповідає за швидкість сходження алгоритму. Ця константа ділиться на довжину маршруту, що приводить до зворотної залежності: чим коротший шлях, тим більше феромону буде відкладено. Перш за все слід зазначити перехід від довжини маршруту до кількості балів за маршрут. Це вже мінятиме зворотну залежність на пряму. Якщо цей параметр в класичному алгоритмі потребував підбору значень, модифікований варіант не стане виключенням. Вибір варіанту відкладання феромону буде залежати в тому числі від обраного критерію вибору наступного кроку. Але в будь якому випадку саме тестування різних варіантів покаже найбільш ефективний з них. В першу чергу було виділено 2 варіанти, на яких базуватиметься кількість феромону що має відкладатися:

— Кількість балів, набрана за маршрут
— Щільність маршруту

До обох цих варіантів може додатися низка дрібних модифікацій, які теоретичного можуть дати ширше поле для пошуку, чи швидшу збіжність до оптимального значення. Але розраховується що їх вплив може бути значно меншим ніж від основних самого варіанту вибору відкладання, то ж за відсутності видимого впливу вони не будуть використовуватися Ці два варіанти це: різна кількість балів на маршрути з фінішем і без, симетричне і асиметричне відкладання феромону. До кожного основного і додаткового параметру можуть додаватися вагові коефіцієнти з метою покращити швидкість збіжності.

5.5 Тестування та порівняння впроваджених функцій

Перед тестуванням розроблених функцій було сформовано вимоги, на основі яких буде проводитися аналіз алгоритмів:

— Максимізація кількості балів
— Мінімізація довжини маршруту
— Мінімізація часу на виконання

— Універсальність алгоритму

Максимізація кількості балів за маршрут має абсолютну перевагу серед вимог, бо більша кількість балів є умовою перемоги на змаганнях. За умови отримання однакових значень за маршрут різними алгоритмами буде виконуватися порівняння довжин маршруту, бо якийсь з них може краще враховувати довжину дистанції. Бо кращі переходи за рівну кількість балів все ще можуть дати значну перевагу при подоланні дистанції. Серед алгоритмів перевага буде віддаватися тому який за меншу кількість ітерацій буде видавати стабільне краще значення. Алгоритм має видавати стабільне краще значення за однакових налаштувань констант для різних мап та різних бажаних довжин маршрутів. Стабільне краще значення при цьому для кожного варіанту своє.

Для порівняння буде використовуватися 3 мапи для рогейну тривалістю 3, 6, і 12 годин з 25, 49 і 63 КП відповідно. Для них буде розраховано дистанцію для середньостатистичної команди спортсменів, на основі якої і буде проводитися порівняльний аналіз. Вона буде складати для 3 годин – 15 км, для 6 годин – 23 км, для 12 годин – 46 км.

Через велику кількість можливих комбінацій параметрів, а на даному етапі вона оцінюється у 12 можливих комбінацій, тестування будуть проводитися в форматі повтору. На кожній ітерації будуть пробуватися та відсіювати неефективні рішення і модифікації. Але при знаходженні оптимальної комбінації вони знову будуть перевірятися що б не впустити раптом гарний варіант оптимізації, який дасть результат лише в поєднання з кращим варіантом. Порівняльний графік буде будуватися лише для обраних необхідних порівнянь параметрів. Результат деяких тестувань може приводитися лише у вигляді їх опису.

Через вірогідносний вибір алгоритм не завжди приходить до найкращого значення. При проведенні тестів після багаторазових спроб було виявлено що не виходить підібрати значення констант для того що б отримувати стабільно повторюваний найкращий з отриманих результат. Тому для порівняння основних модифікацій алгоритму, а саме критерію вибору наступного кроку і

варіанту відкладання феромону буде використовуватися середнє значення кількості балів і довжини дистанції на конкретній кількості ітерацій отримане з 25 результатів тесту алгоритму.

5.5.1 Тестування та вибір додаткових параметрів відкладання феромону

Для початку буде перевірено додаткові оптимізаційні припущення на одній мапі, бо це найпростіше, і в подальшому зменшить кількість необхідних тестів. Вони можуть знову перевіряться лише при знаходженні загального оптимального варіанту.

Перше тестування – це можливі комбінації додаткових параметрів відкладання феромону:

- Симетричне, за маршрути з фінішем більше феромону
- Симетричне, за маршрути з фінішем стільки ж феромону
- Асиметричне, за маршрути з фінішем більше феромону
- Асиметричне, за маршрути з фінішем стільки ж феромону

За початковий варіант було обрано простішу комбінацію з орієнтованим графом. Значення інших констант встановлювалися ті що були знайдені для класичного варіанту алгоритму.

Що б відобразити поєднання 3 параметрів було побудовано 2 графіки залежності, що відображені на точковій діаграмі. Перший - залежність кількості балів від кількості ітерацій, зображений на рисунку 5.5. Він дає можливість оцінити роботу всіх додаткових варіацій алгоритму за ефективністю знаходження дорожчих маршрутів. Однозначно можна сказати що для цієї модифікації алгоритму необхідно більше ітерацій для отримання стійкого кращого результату. Але головне це порівняння кожного з них, для того що б обрати той що, умови якого пришвидшують сходимість алгоритму. Неочікуваним виявилося що результати роботи кожного дуже близькими, не даючи якоїсь очевидної переваги. Явно виділяється лише варіант де за

маршрути з фінішем більше феромону, відкладання симетричне. Дає найгірші показники за балами при малій кількості ітерації. І симетричний варіант, з однаковим відкладанням феромону за фініш, який єдиний знайшов варіант на 213 балів. Але це скоріше вийняток ніж правило, і таких результатів можна буде досягти краще налаштувавши параметри.

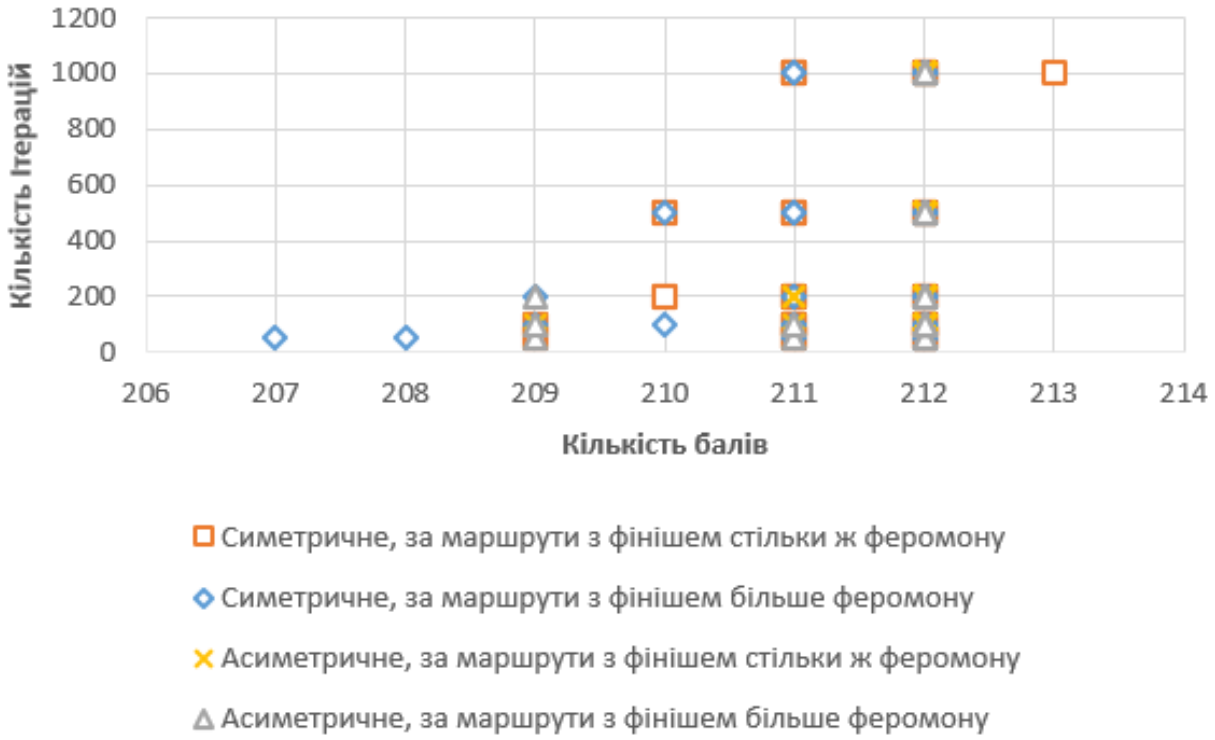

Рисунок 5.5 – Відношення кількості ітерацій до кількості набраних балів

Більшість інших маршрутів дають кращий результат у 212 балів. Для подальшого аналізу доведеться використати другорядне порівняння у за довжиною маршруту. Цей графік залежності кількості набраних балів від довжини маршруту продемонстрований на рисунку 5.6.

Цей графік добре демонструє обмеження на використання параметру щільності, який на перши погляд міг би замінити обидва параметри кількості балів та довжини маршруту у графіку відношень до кількості ітерацій. Видно, що маршрути набираючі менше балів зазвичай коротші. А от же значення були б доволі близькими, і не давали б ніякої оцінки головному параметру –

кількості набраних балів.

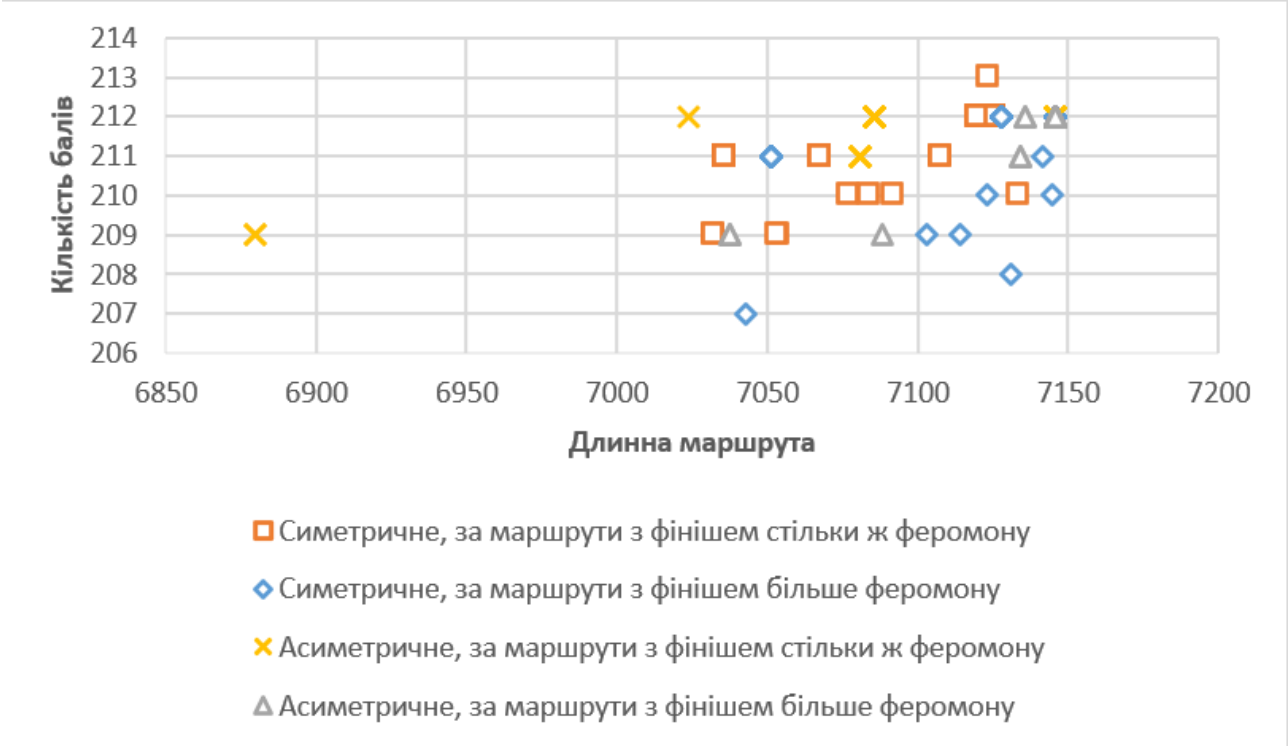

Рисунок 5.6 – Відношення кількості балів до довжини маршруту

На даному графіку необхідно обирати маршрути що полягають вище та лівіше. В даному випадку можна побачити що особливо відрізняється асиметричне і однакове по відношенню до фінішу відкладання феромону. І непоганий результат демонструє таке саме але симетричне відкладання феромону. Для остаточного аналізу було розглянуто середні загальні середні значення без врахування кількості ітерацій. Вони підтвердили ці ж спостереження. Хоча асиметричне відкладання виглядає так ніби дає навіть кращий результат, та за умови вибору третього параметру для критерію вибору наступного кроку воно може бути не стільки ж ефективним, бо матриця впливу там симетрична.

Однозначно можна сказати що теорія про більшу ефективність при відкладанні різної кількості феромону за маршрути з фінішем і без не виправдала себе. Або через те що не вдалося підібрати правильний ваговий коефіцієнт, або тому що без нього кількість феромону що відкладається є

ближчою до оптимальної її кількості. Тож користуючись принципом бритва Оккама, а також логікою яка може пришвидшити виконання програми, прибравши зайві інструкції і більшою універсальністю було обрано варіант симетричного відкладання феромону з однаковою кількістю для маршрутів з фінішем і без.

5.5.2 Тестування та вибір критерію вибору наступного кроку та кількості відкладання феромону.

Залишилося ще 4 варіанти, два з яких це критерії вибору наступного кроку, а ще два –кількість відкладання феромону:

- Орієнтований граф
- Додавання третього параметру в функцію вибору наступного кроку
- Відкладання феромону за кількістю балів за маршрут
- Відкладання феромону за щільністю маршруту

Обирання між ними так само буде полягати в їх комбінації та попередньому знаходженню оптимальних значень ступеня у випадку третього параметру та вагових коефіцієнтів для кількості відкладання феромону. для кожного поєднання. Чотири комбінації для тестування:

- Орієнтований граф і кількість балів
- Орієнтований граф і щільність маршруту
- Третій параметр і кількість балів
- Третій параметр і щільність маршруту

Для початку було проведено тестування кожного з 4 варіанту алгоритму на трьох мапах з 25, 49 і 63 контрольними пунктами на 50, 100, 200, 500 і 1000 ітерацій для отримання достатньої кількості значень що б розрахувати середню кількість балів. Середнє значення групувалося за кількістю ітерацій.

На графіках, що покажуть кількість балів, набрану кожним варіантом алгоритму на конкретній мапі (рис 5.7 – 5.9), замість кількості ітерацій буде

використовуватися показник часу . Це дає можливість одразу оцінити скільки забирає виконання кожної функції при різній кількості КП.

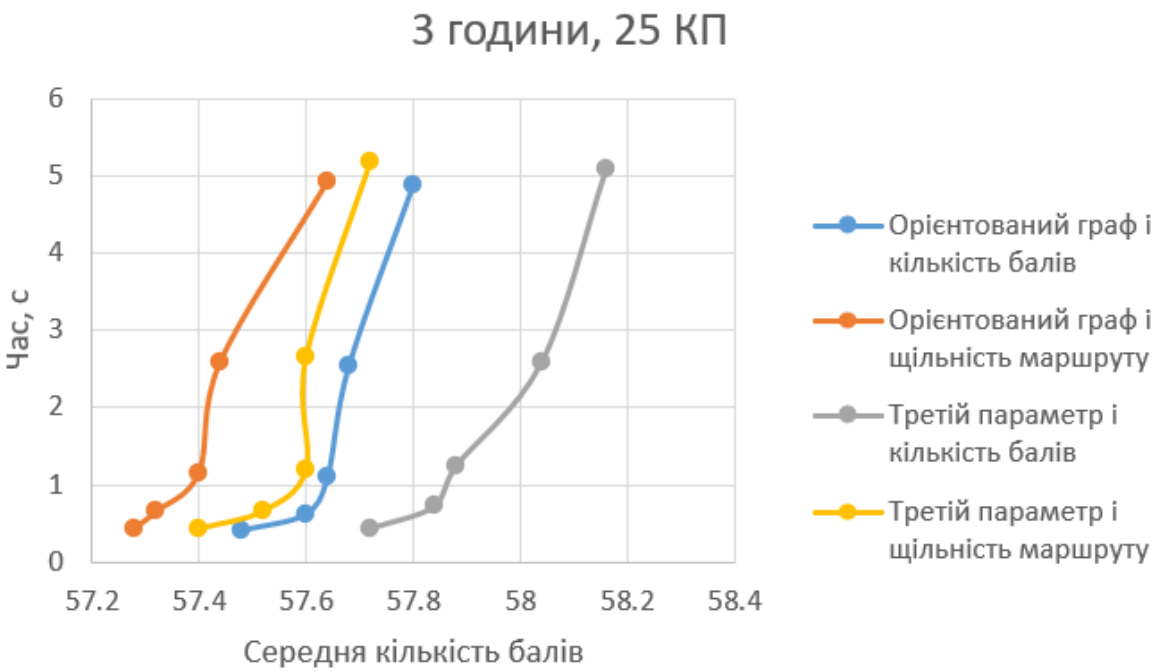

Рисунок 5.7 – Графік відношення витраченого часу до кількості набраних балів на мапі з 25 КП

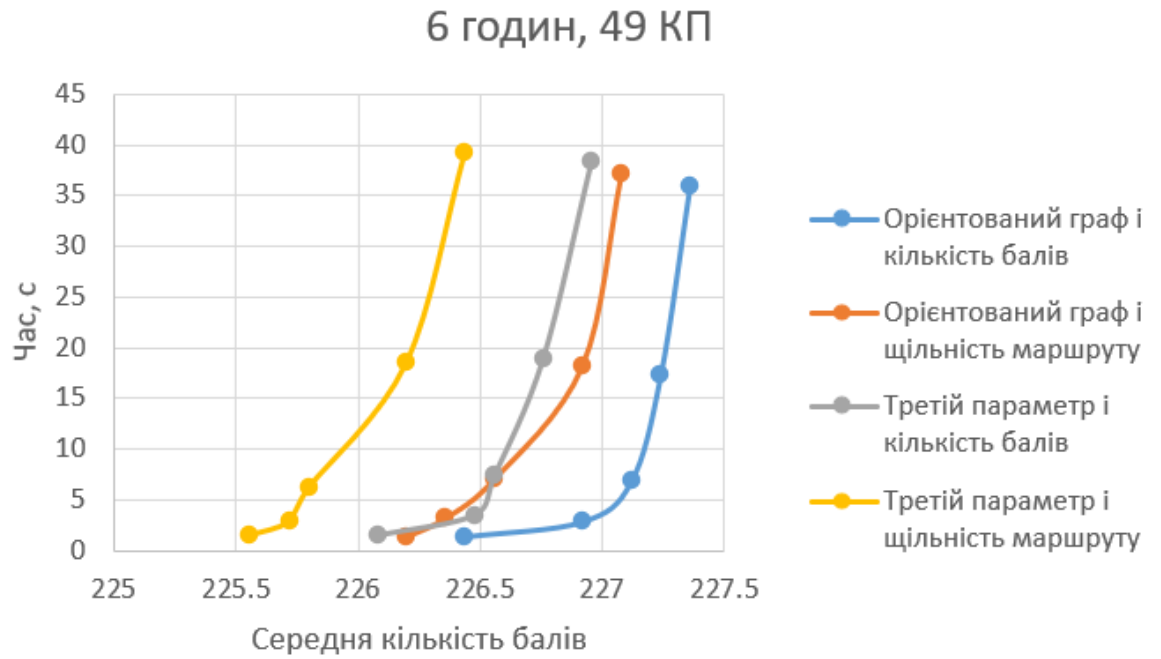

Рисунок 5.8 – Графік відношення витраченого часу до кількості набраних балів на мапі з 49 КП

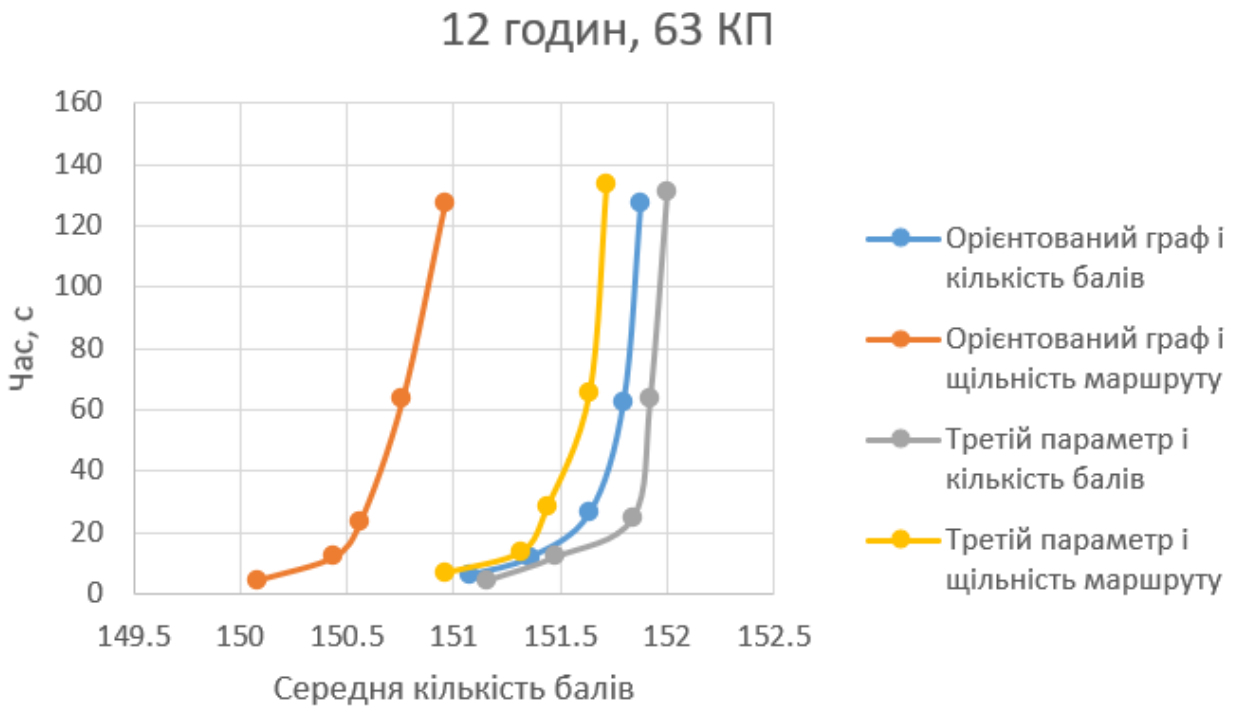

Рисунок 5.9 – Графік відношення витраченого часу до кількості набраних балів на мапі з 63 КП

Середня кількість балів не означає що зі більшу кількість ітерацій алгоритм видасть маршрут на більшу кількість балів, а лише відображає вірогідність отримання кращого рішення. І за цими графіками можна однозначно визначити що на для всіх розмірностей графу більша кількість ітерацій, дає більший шанс на отримання кращого результату. І дає одразу можливість оцінити скільки часу витрачається на побудову маршруту.

Що б об’єктивніше оцінити роботу алгоритму на кожній з мап було проведено нормалізацію значень. Цей метод полягає у тому, щоб масштабувати дані так, щоб вони знаходилися в новому заданому діапазоні, від 0 до 1, що відповідно найгірше і найкраще знайдене середнє значення. На графіках буде зображено набирання алгоритмом балів відносно максимально отриманого значення на мапі на кожній кількості ітерацій. Такі графіки дають можливість краще оцінити їх ефективність на мапах різної розмірності, попри те що на таких мапах набирається дуже різна кількість очок. Ці відношення зображені на рисунках 5.10 – 5.13.

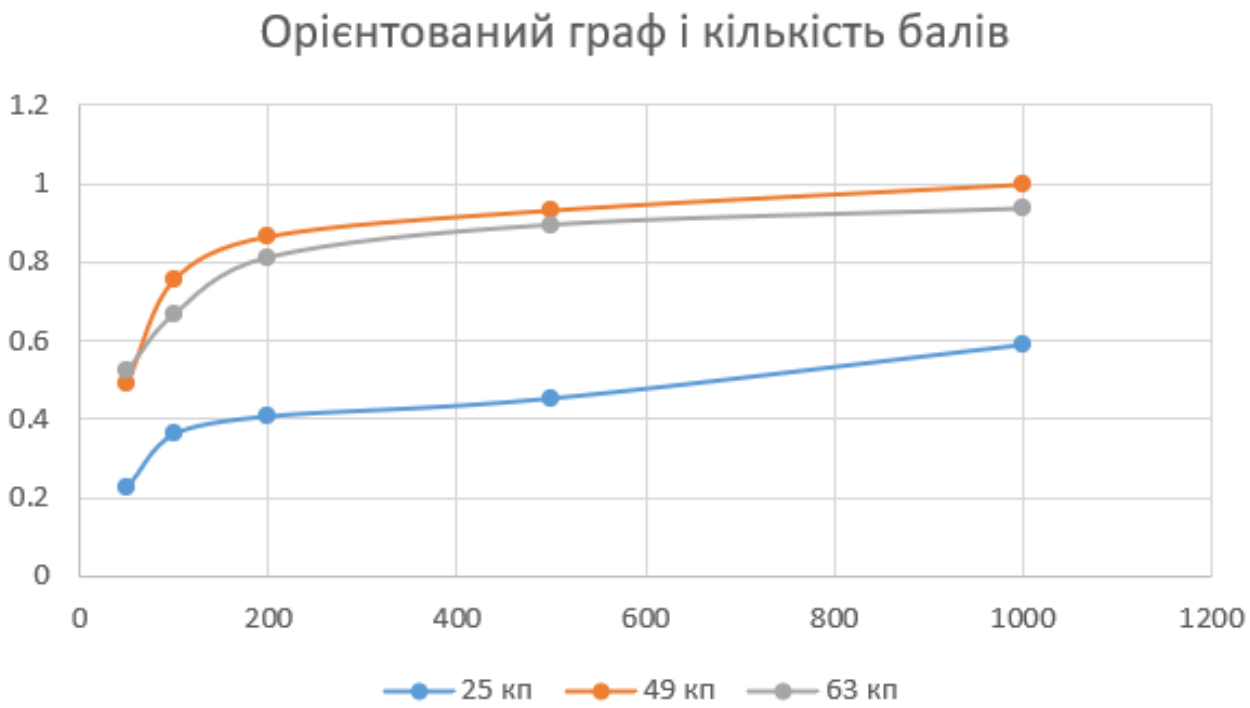

Рисунок 5.10 – Графік ефективності алгоритму з орієнтованим графом і відкладанням феромону на основі кількості балів

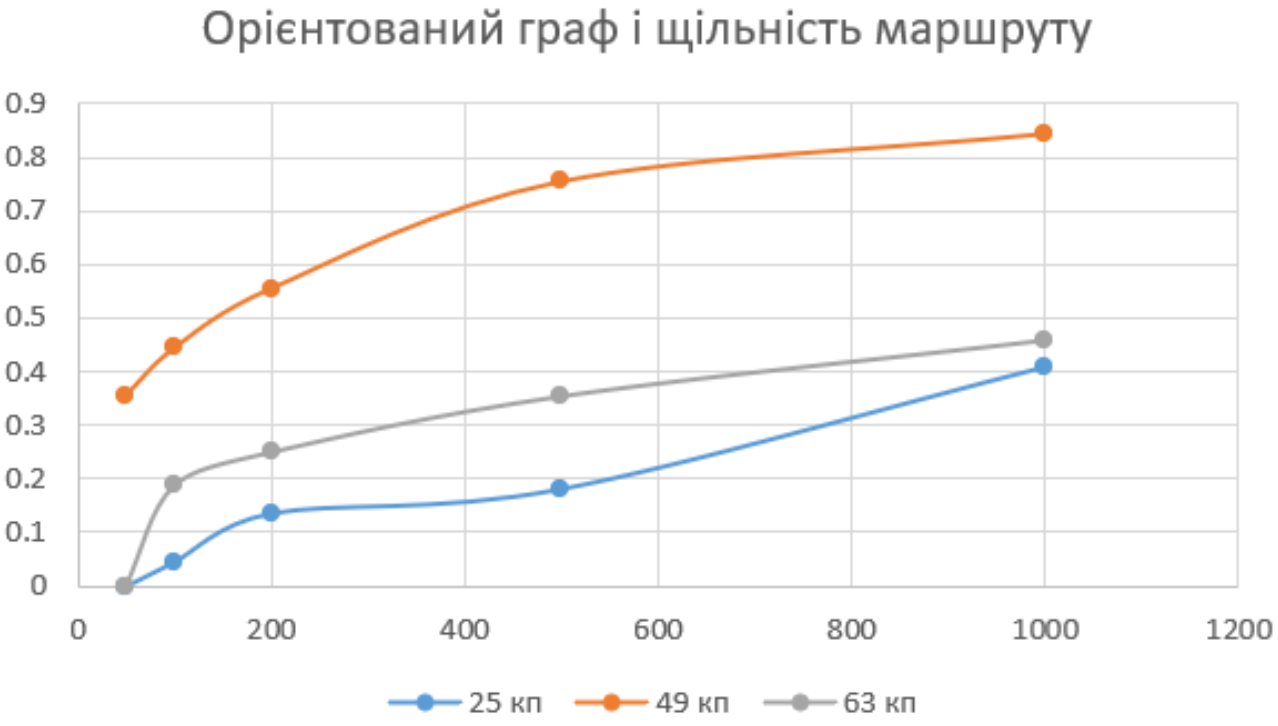

Рисунок 5.11 – Графік ефективності алгоритму з орієнтованим графом і відкладанням феромону на основі щільності маршруту

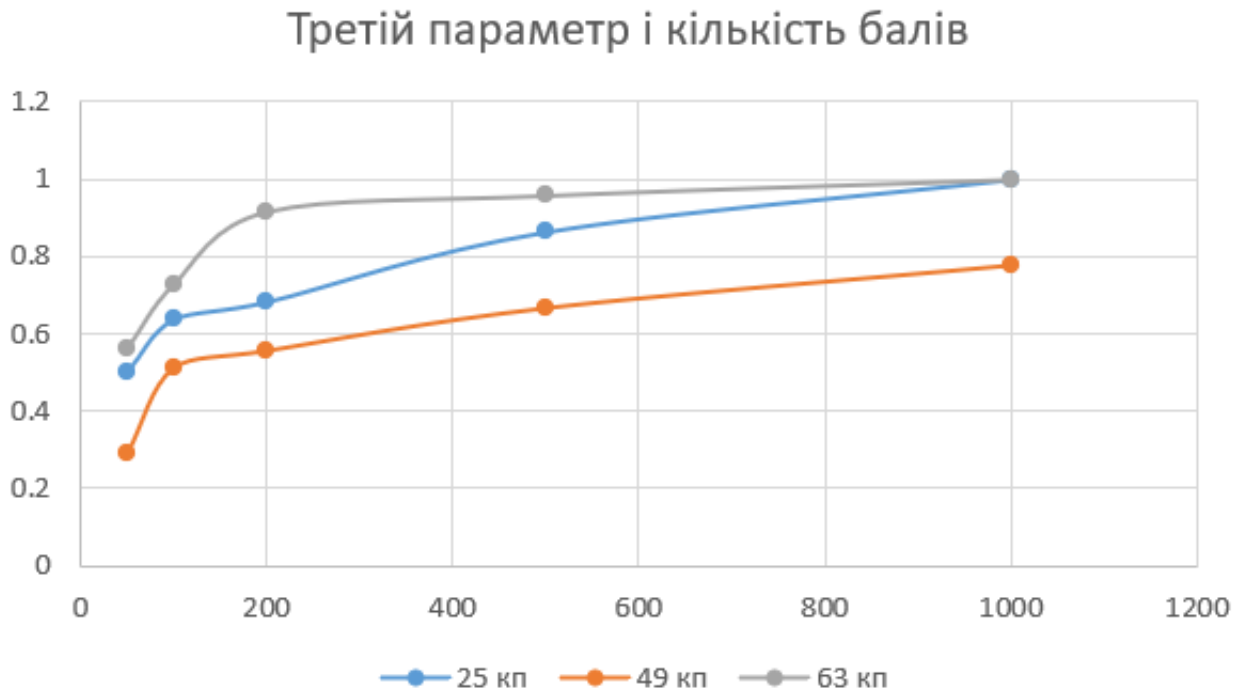

Рисунок 5.12 – Графік ефективності алгоритму з третім параметром критерію вибору і відкладанням феромону на основі кількості балів

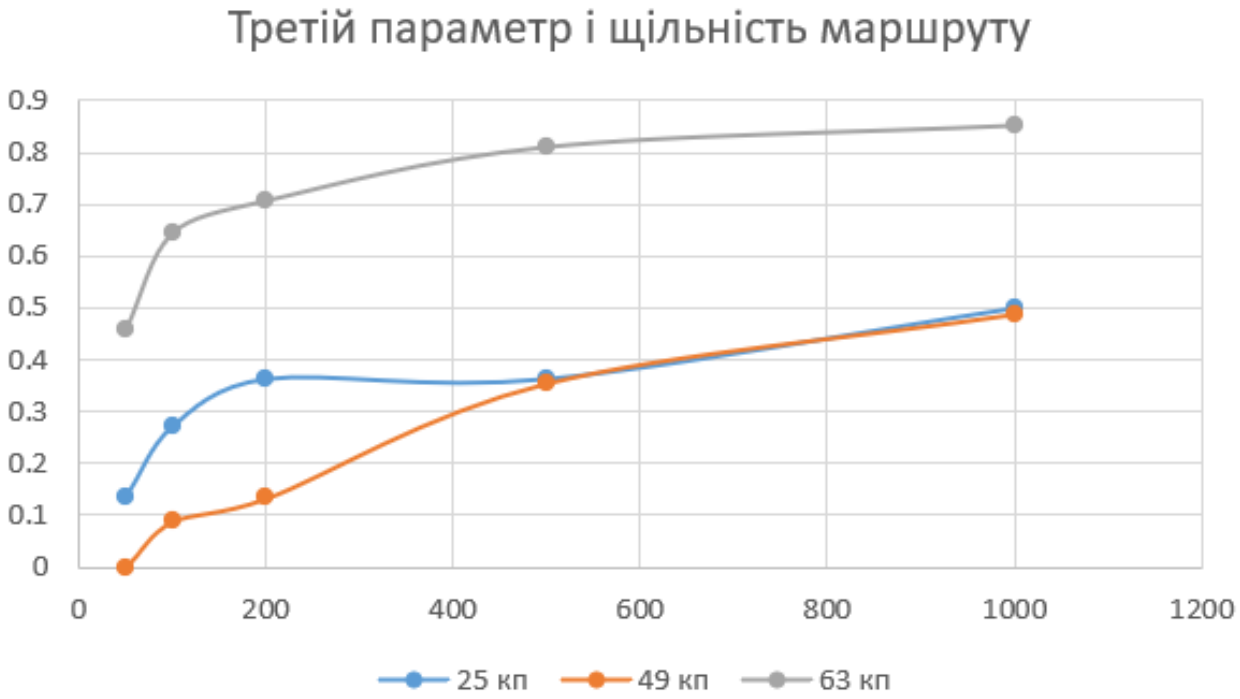

Рисунок 5.13 – Графік ефективності алгоритму з третім параметром критерію вибору і відкладанням феромону на основі щільності маршруту

Огляд цих графіків так само підтверджує спостереження про більшу ефективність за більшу кількість кроків. Це контрастує з оригінальним алгоритмом де найкраще значення зазвичай знаходилося за перші 50 ітерацій. Також в поєднанні з попередніми графіками можна однозначно виділити найменш що алгоритми з орієнтованим графом і відкладанням феромону на основі щільності маршруту і третім параметром критерію вибору і відкладанням феромону на основі щільності маршруту є найменш ефективними через свою неуніверсальність. Вони видають гарні значення для мап відповідно з 49 і 63 КП, але при цьому дуже посередні для інших розмірностей. Це не відповідає поставленим вимогам. Відповідно інші два алгоритми мають кращі значення для всіх мап. орієнтованим графом і відкладанням феромону на основі кількості балів.

Алгоритм з третім параметром критерію вибору і відкладанням феромону на основі кількості балів має найкращі значення для малих і великих графів, але втрачає ефективність на середніх розмірностях. Попри те що цей варіант має додаткові константи для підбору, а от же для можливості налаштування алгоритму, не вдалося підібрати значення для більшої універсальності і для знаходження кращих значень[17].

Варіант з орієнтованим графом і відкладанням феромону на основі кількості балів видає більш стійкі рішення для актуальніших мап. Хоча значення на мапі з 63 КП є трохи менш стабільні, та значно менш стабільні на 25 КП цей алгоритм єдиний знаходив абсолютні найкращі значення на мапах 49 і 63 КП. Також він видає найкраще значення як раз для середнього сегменту по кількості КП, мап у якому найбільше. Також через меншу кількість розрахунків його швидкодія трохи більша за інші алгоритми.

5.6 Аналіз та оцінка обраного варіанту алгоритму

На рисунку 5.14 зображено приклад маршруту які спортсмени будують на замганнях з рогейну і найкращий знайдений варіанту маршруту за допомогою

обраного алгоритму. Це мапа для дистанції на 12 годин, для якої було визначено дистанцію у 46 км.

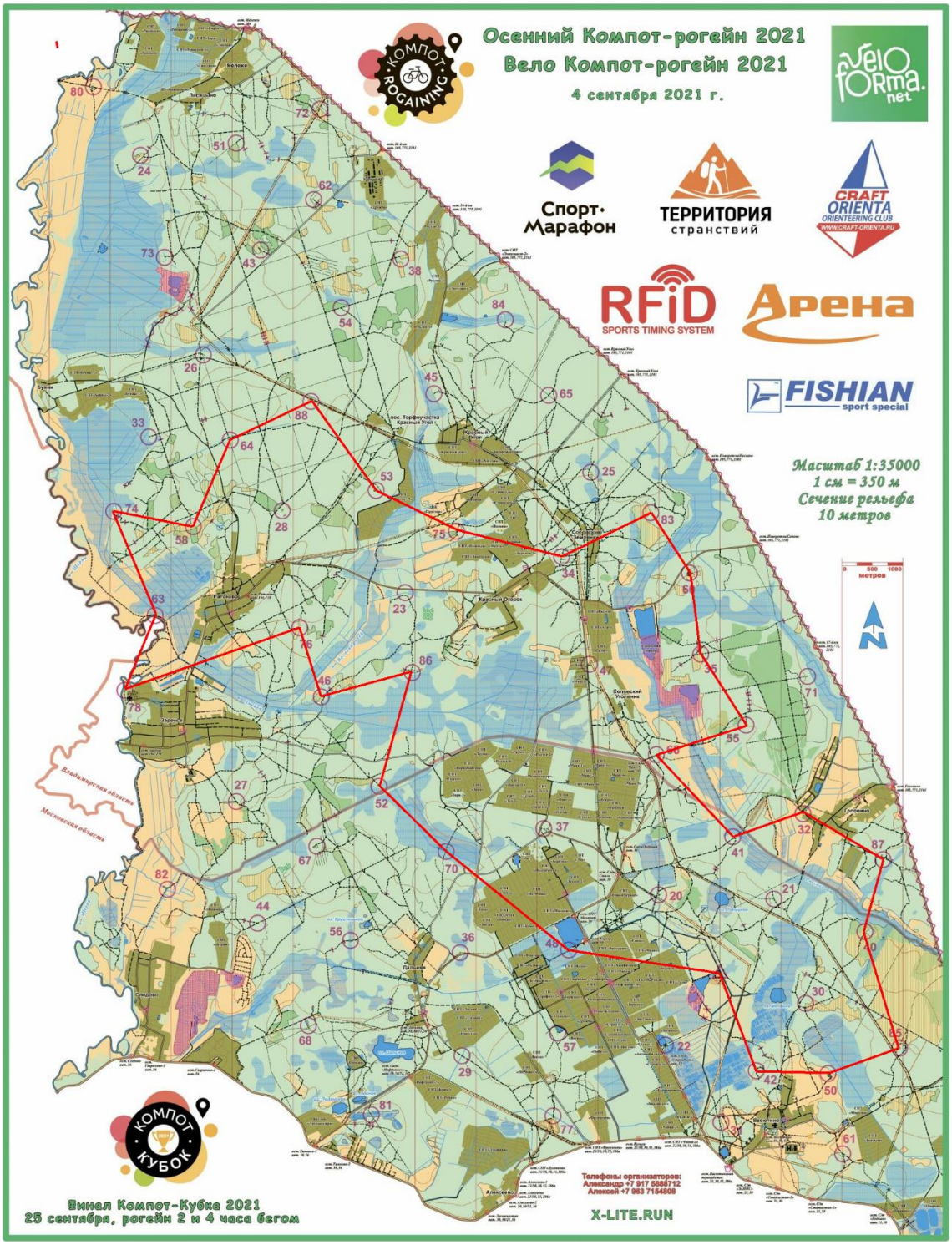

Рисунок 5.14 – Приклад маршруту на рогейн побудованого алгоритмом

Даний маршрут складає 153 бали, і має дистанцію менше 46 кілометрів,

бо алгоритм не може перевищити це значення. Для оцінки якості знайденого маршруту було розглянуто спліти зі змагань з яких було взято цю мапу. Цей ресурс дозволяє переглянути маршрути учасників, а також вказує їх довжину, як і у випадку алгоритму погуку маршруту – навпростець і кількість набраних балів. Найближчими по результатах виявилися команди що набрали 150 балів за 49 і 53 км, а також 155 балів за 52 км. А для дистанції у 52 км алгоритм набрав вже 173 бали.

То ж даний алгоритм надає цілком непогане рішення за короткий проміжок часу, який має значний запас від встановленого в вимогах. Швидкість виконання дозволяє запускати алгоритм декілька разів, що може компенсувати навіть отримання посередніх результатів відносно арени мапи, через те що алгоритм ні яким чином не враховує саму мапу а будує маршрут лише на основі даних про її граф. Це дозволяє евристично оцінити отримані варіанти та обрати саме той який спортсмени вважатимуть найкращим.

ВИСНОВКИ

В результаті проведеної роботи було виконано збір та аналіз даних для створення алгоритму для побудови маршрутів на картах рогейну. Було проведено аналіз існуючих систем та алгоритмів. Було обґрунтовано вибір алгоритму для модифікації

В результаті виконання роботи було розроблено план модифікації та реалізовано його згідно поставленої задачі.

В роботі також використовувалися дослідження, які були описані та опубліковані у вигляді тез до наукових конференцій у Харкові та Києві.

В результаті виконання роботи було створено модифікацію мурашиного алгоритму яка може будувати маршрут з заданим обмеженням довжини на мапах різних розмірів і різних принципів побудови. Були враховані не всі типи мап, зокрема тих які вимагають особливого підходу і логіки для побудови дистанції, що вимагає скоріше особистого досвіду. Алгоритм не враховує арену змагань, а саме рельєф, природні перепони, тип місцевості, бо мапа в даному випадку є лише способом для точного введення даних, а не безпосереднім об’єктом на якому прокладається маршрут. Результати роботи алгоритму видають наближене рішення у графі, а от же здатний генерувати різноманітні, але не завжди оптимальні рішення і за одних вхідних даних, а от же є лише варіантами для спортсменів та потребують евристичного аналізу.

Цей проект можна використовувати як один з елементів застосунку для побудови маршрутів, або для вирішення задач зі схожими умовами.

Подальший цього алгоритму може бути пов’язаний з покращенням способу отримання вхідних даних, автоматичного розпізнавання контрольних пунктів та масштабу мапи, враховуваня заборонених територій і об’єктів які заборонено пересікати, типу місцевості. Також корисним адаптування алгоритму для адаптації його під інші платформи і розробка способу для більшої отримання більшої універсальності а також вдосконалення для досягнення більшої стійкості в отримуванні кращих результатів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Рогейн URL: https://ru.wikipedia.org/wiki/Рогейн (дата звернення 20.06.2022)

2. Mastorakis, N. (2011). Pathway modeling and algorithm research. Hauppauge, N.Y.: Nova Science Publishers.

3. Protsenko, A., & Ivanov, V. (2022). Using the E-Netsim application for visual simulation of the movement of the autonomous robots. Advanced Information Systems, 6(1), 27–31. https://doi.org/10.20998/2522-9052.2022.1.04

4. Grebennik I. Enumeration and Generation of Permutations with a Partially Fixed Order of Elements / I . Grebennik, Grebennik. Yu. Stoyan, O. Lytvynenko, V. Kalashnikov // International Journal of Combinatorial Optimization Problems and Informatics, Vol. 8, No. 1, Jan-April 2017, pp. 19-30. ISSN: 2007-1558. ISI Web of Science (Thomson & Reuters).

5. Планирование в рогейне - это высшая математика. (2019). Retrieved 10 September 2019, from https://ziurkenas.livejournal.com/38843.html

6. Ісаєнко А.Ю., Іванов В.Г. Розробка компонентів системи для побудови маршрутів на мапах для змагань з рогейну. Інформаційні технології в соціокультурній сфері, освіті та економіці: матеріали VII Міжнар. наук.-практ. конф. студентів і молодих учених, м. Київ, / М-во освіти і науки України; Київ. нац. ун-т культури і мистецтв. Київ: Вид. Центр КНУКіМ, 2023. . -с. 46-47

7. M. Dorigo, M. Birattari, T.Stutzle, “Ant colony optimization – Artificial ants as a computational intelligence Technique”, IEEE Computational intelligence magazine, 2006.

8. Ant Colony Optimization Solutions for Path Planning of Logistic. (2018). International Journal Of Technology And Engineering Studies, 4(3). doi: 10.20469/ijtes.4.10003-3

9. Технологический институт Федерального государственного образовательного учреждения высшего профессионального образования

Южный федеральный университет в г. Таганроге. (2008). О некоторых модификациях муравьиного алгоритма.

10. Dorigo, M., & Gambardella, L. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Transactions On Evolutionary Computation, 1(1), 53-66. doi: 10.1109/4235.585892

11. Arora, P., Kaur, H., & Agrawal, P. (2011). Analysis and synthesis of enhanced ant colony optimization with the traditional ant colony optimization to solve travelling sales person problem. International journal of computers & technology, 1(1), 88-92. doi: 10.24297/ijct.v2i2b.2637

12. M. Dorigo, V. Maniezzo “Ant Colony optimization”. [Online service]. Access mode: http://informatics.indiana.edu/jbollen/I501F13/readings/ dorigo99ant.pdf.

13. M. Dorigo, V. Maniezzo, A. Colorni, “The Ant System: Optimization by a colony of cooperating agents” // IEEE Transactions on Systems, Man, and Cybernetics-Part B, 26, 1, стр. 29-41, 1996 г.

14. Ісаєнко А.Ю.,Науковий керівник - Іванов В.Г. Розробка елементів системи формування маршрутів для змагань з рогейну: Матеріали 27-го Міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті» стр.23-24. Зб. Матеріалів форуму Т. 6.ч2. –Харків: ХНУРЕ. 2023.

15. T. Stützle, H. Hoos, “MAX-MIN Ant System and local search for the traveling salesman problem” // IEEE International Conference on Evolutionary Computation, стр. 309-314, 1997 г.

16. S. C. Zhan, J. Xu, J. Wu, “The optimal selection of the parameters of the ant colony algorithm”, bulletin of Science and Technology, 19(5): pp.381- 386, 2003.

17. T. Stützle, M. López-Ibáñez, P. Pellegrini, M. Maur, M. de Oca, M. Birattari, Michael Maur, M. Dorigo, “Parameter Adaptation in Ant Colony Optimization” // Technical Report, IRIDIA, Université Libre de Bruxelles, 2010 г.