МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М.Є. Жуковського
«Харківський авіаційний інститут»

Факультет ракетно-космічної техніки

Кафедра вищої математики та системного аналізу

Пояснювальна записка до дипломної роботи

магістра
(освітньо-кваліфікаційний рівень)

на тему «Двохагентна задача комівояжера»

ХАІ.405.463М.103145.18010001.15В

Виконав: студент 6 курсу групи №463м
Спеціальність: 124«Системний аналіз»
(код та найменування)
Освітня програма: «Системний аналіз і управління»
(найменування)
Гріщенко Д.В.
(прізвище й ініціали студента)
Керівник: Курєннов С.С.
(прізвище й ініціали)
Рецензент: Дорошенко В. О.
(прізвище й ініціали)

Харків – 2019

Міністерство освіти і науки України
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

Факультет ракетно-космічної техніки
(повне найменування)
Кафедра вищої математики та системного аналізу
(повне найменування)
Рівень вищої освіти другий (магістерський)
Спеціальність 124 «Системний аналіз»
(код та найменування)
Освітня програма Системний аналіз та управління - магістр

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Ніколаєв О.Г.
(підпис) (ініціали та прізвище)
«___» _____ 2019 р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТУ
Гріщенко Даніїла Віталійовича
(прізвище, ім'я та по батькові)

1. Тема дипломного проекту (роботи) «Двохагентна задача комівояжера»
керівник дипломного проекту (роботи) Курєннов С.С. д. т. н. професор
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом Університету від «___» _____2019__ року №_____-уч
2. Строк подання студентом дипломного проекту (роботи) _____
3. Вихідні дані до роботи _____
4. Зміст пояснювальної записки (перелік завдань, які потрібно розв'язати)
   1) Аналіз інформації по заданій тематиці.
   2) Системний аналіз об'єкта дослідження.
   3) Аналіз існуючих алгоритмів маршрутизації.
   4) Програмна реалізація алгоритмів та їх модифікацій.
   5) Аналіз вихідних даних.
   6) Розрахунок собівартості програмного продукту.
5. Перелік графічного матеріалу

_____
_____

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Загальна частина	д. т. н., професор каф. 405 Курєннов С.С.		
Економічна частина	.		

Нормоконтроль _____ _____ «____» _____20__ р.
(підпис) (ініціали та прізвище)

7. Дата видачі завдання «___» _____ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Огляд інформації з заданої тематики		
2	Визначення актуальності проблеми, цілей і завдань дослідження		
3	Системний аналіз об’єкта дослідження		
4	Розробка програмного продукту		
5	Оцінка економічної ефективності застосування програмного продукту і визначення його ціни		

Студент _____ Гріщенко Д.
(підпис) (ініціали та прізвище)

Керівник проекту (роботи) _____ Курєннов С.С.
(підпис) (ініціали та прізвище)

Реферат

Пояснювальна записка до роботи магістра: ___ сторінок, ___ розділів, __ рисунка, ___ таблиць, та не містить додатків.

Maple, ЗАДАЧА КОМІВОЯЖЕРА, ДВОХАГЕНТНА ЗАДАЧА, ОПТИМІЗАЦІЯ, ДИСКРЕТНЕ ПРОГРАМУВАННЯ, СИСТЕМНИЙ АНАЛІЗ.

Об'єктом дослідження є алгоритм розв’язання некласичної задачі дискретного програмування.

Метою даної роботи є дослідження методів оптимізації маршрутів для двох агентів, аналіз існуючих алгоритмів та методик, пошук шляхів розв’язку існуючих і побудова нових задач.

Предмет дослідження: двохагентна задача комівояжера та її різновиди, задачі дискретної оптимізації за наявністю обмежень, побудова алгоритмів розв’язку.

Основні завдання:

- аналіз інформаційних джерел на тему розв’язання узагальнених задач комівояжера.
- аналіз існуючих задач оптимальної маршрутизації та методів їх розв’язання.
- проведення морфологічного, функціонального та інформаційного аналізу задачі маршрутизації.
- Розробка програмного продукту, зо реалізує основні алгоритми маршрутизації.

Методи дослідження:

- класифікаційне дослідження об'єкта.
- морфологічний аналіз.
- функціональний аналіз.
- інформаційний аналіз.

Реферат

Пояснительная записка к работе магистра: __ страниц, __ разделов, __ рисунков, __ таблиц и не содержит приложений.

MAPLE, ЗАДАЧА КОММИВОЯЖЕРА, ДВУХАГЕНТНАЯ ЗАДАЧА КОММИВОЯЖЕРА, ОПТИМИЗАЦИЯ, ДИСКРЕТНОЕ ПРОГРАММИРОВАНИЕ, СИСТЕМНЫЙ АНАЛИЗ.

Объектом исследования является алгоритмы решения неклассической задачи дискретного программирования.

Целью данной работы является исследование методов оптимизации маршрутов для двух агентов, анализ существующих алгоритмов и методик, поиск путей решения существующих и построение новых типов задач.

Предмет исследования: двухагентная задача коммивояжера и ее разновидности, задачи дискретной оптимизации при наличии ограничений, построение алгоритмов решения. Основные задачи:

- анализ информационных источников на тему маршрутизации.

- анализ существующих задач оптимальной маршрутизации и методов их решения.

- проведение морфологического, функционального и информационного анализа задачи маршрутизации.

- разработка программного продукта, который реализует основные алгоритмы маршрутизации для двух агентов.

Методы исследования:

- классификационное исследование объекта.

- морфологический анализ.

- функциональный анализ.

- информационный анализ.

ABSTRACT

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

БПЛА – безпілотний літальний апарат;
ЗК – задача комівояжера;

ВСТУП

В даний час у зв'язку з бурхливим розвитком логістичних мереж компаній, а також вимог сучасного темпу розвитку бізнесу і конкурентних відносин, важливо знаходити оптимальні рішення щодо зниження логістичних витрат, частиною яких є транспортні витрати.

Можливість їх зменшення завжди цікавить перевізників. В даній роботі сформульовано кілька задач, пов'язаних з цією проблемою. Серед них – узагальнена задача комівояжера, а саме задача для двох комівояжерів (агентів). Її метою є мінімізація вартості, або часу об'їзду міст не одним, а двома комівояжерами.

На практиці постановка даної задачі може мати безліч варіантів. Можливо, потрібно буде брати до уваги максимальне завантаження одного комівояжера, затримку при відвідуванні кожного міста, можливість існування декількох баз, і т.д. Ми не будемо враховувати такого роду деталі і наведемо спрощену постановку цієї задачі. Це дозволить нам досліджувати базових алгоритмів, які будуть основою практичної реалізації, яка враховує всі тонкощі.

Наявність двох агентів збільшує кількість варіантів формулювання задачі. Наприклад, можуть досліджуватися наступні варіації цієї задачі:

1) Сумарна вартість (час обслуговування, довжина) всіх маршрутів повинна бути мінімальною.
2) Маршрути кожного з агентів мають містити однакову кількість вузлів або кількість вузлів у маршрутах може відрізнятися на одиницю, і при цьому забезпечувати мінімальну сумарну вартість маршрутів.
3) Маршрути кожного з агентів містять, або не містять одну спільну точку (спільну для кожного з агентів базу).
4) Спільний час обслуговування маршрутів є мінімальним.
5) Всі вузли мають бути відвідані один раз, або можуть бути не відвідані вузли (за обмеженої протяжності маршрутів).
6) Агенти мають однакову швидкість або розрізняються швидкістю.

7) На маршрутах існують (або відсутні) умови або потреби у відвідуванні деяких вузлів заданим агентом, або відвідуватися двічі – по разу кожним з агентів.
8) Загальна кількість відвідуваних агентами вузлів має бути максимальною за умов виконання обмежень на довжину маршруту кожного з агентів (відвідування всіх вузлів за певних причин неможливе і необхідно відвідати найбільшу кількість з них)
9) Комбінації перелічених вище задач.

Як відомо, задача комівояжера - одна з оптимізаційних NP-важких задач, яка не вирішується за поліноміальний час. Вона може бути поставлена в комбінаторної формулюванні як задача про знаходження послідовності з n міст з мінімальною вартістю циклічного об'їзду. Складність задачі комівояжера в тому, що спроба знайти якийсь локальний розв’язок не дозволяє побудувати шлях для вирішення всієї задачі.

В даний час методи пошуку точних і наближених рішень ЗК вивчалися багатьма дослідниками і запропоновано безліч алгоритмів, що дозволяють знайти такі розв’язки. В останній час бурхливий розв’язок мають евристичні алгоритми, які мають назву «мурашиний алгоритм», і які моделюють рух по системі великої кількості агентів («мурах»), які залишають за собою феромони, що дають підказку іншим агентам. Але такі алгоритми не завжди можуть вказати точний розв’язок задачі. У роботі застосовано класичний алгоритм гілок та меж, який модифіковано на даний тип задач.

Ще одним важливим напрямом застосування багатоагентної задачі комівояжера є маршрутизація польоту безпілотних літальних апаратів (БПЛА). Автоматичне формування завдання польоту ставить перед експлуатантами низку завдань і проблем. Однією з таких проблем є раціональне формування маршрутів польоту безпілотних апаратів. Актуальність цієї проблеми обумовлена тим, що раціональна маршрутизація польоту малих літальних апаратів є прямим шляхом до підвищення ефективності їх цільового функціонування. Особливістю застосування БПЛА у багатьох задачах є їхнє

групове застосування. Такими задачами є задачі військової і не тільки розвідки, задачі доставки вантажів, контролю і моніторингу території, налагоджування зв’язку, тощо. Відомим прийомом є зведення задачі маршрутизації до задачі комівояжера. Це зрозуміло, якщо врахувати, що однією з найбільш поширених схем польоту розглянутих апаратів є польоти по замкнутому маршруту, який повинен з'єднати набір точок з відомим місцем старту та приземлення БПЛА. Класичним критерієм при такому трактуванні завдання маршрутизації є тривалість польоту за маршрутом. Мінімізація цього критерію дозволяє знизити витрату пального, підвищити оперативність відповідної системи і знизити ризики знищення БПЛА противником.

Групове застосування БПЛА відкриває нові можливості, але потребує нових заходів, моделей та підходів до планування і виконання польотних завдань. Це становить перед спеціалістами з математичного моделювання нові задачі і потребує створення нових методів або розвитку класичних методів математичного моделювання і оптимізації.

1. АНАЛІЗ ЗАДАЧІ

1.1 Актуальність роботи

Математична постановка задачі комівояжера (ЗК) зводиться до знаходження $n(n-1)$ булевих змінних $x_{ij} \in \{0;1\}$, $(i, j = 1,2,...,n;\ i \neq j)$, які мінімізують час відвідування всіх заданих точок, тобто забезпечують

$$L = \sum_{i=1}^{n} \sum_{j=1}^{n} t_{ij} x_{ij} \to \min, \quad i \neq j\,. \tag{1.1}$$

де t_{ij} - це час переміщення точки з номером *i* в точку з номером *j*. Якщо $x_{ij} = 1$, то рухаючись по маршруту, агент з точки *i* переміщається безпосередньо в точку *j*. Якщо $x_{ij} = 0$ то при русі по маршруту, переміщення з точки з номером *i* в точку з номером *j* не відбувається.

При цьому повинні бути виконані обмеження:

$$\sum_{i=1}^{n} x_{ij} = 1;\ \sum_{j=1}^{n} x_{ij} = 1\,. \tag{1.2}$$

Обмеження відображають те умова, що агент повинен побувати в кожній з заданих точок рівно один раз.

Як відомо, постановка задачі є завданням про призначення, рішення якої містить n змінних рівних одиниці, а решта $n(n-1)$ є нульовими. Рішення може складатися з декількох простих вершинно-непересічних циклів, так званих підциклів, що проходять через меншу ніж *n* число точок. Тому, для отримання маршруту, що зв'язує n точок, необхідно доповнити наведені вище обмеження умовами, що забезпечують зв'язність шуканого циклу.

Якщо у системі два агента, швидкості яких можуть у загальному випадку відрізнятися, то задача значно ускладнюється. Вочевидь розв’язком її буде два цикли, але при цьому в задача буде ускладнюватися додатковими умовами або критеріями оптимальності.

Багатоагентні задачі комівояжера природнім чином виникають при оптимізації маршрутів доставки товарів за точками реалізації двома і більше машинами (агентами), які можуть виїжджати з одного міста (депо) або з

декількох місць, та обслуговувати загальну систему місць реалізації товару. При цьому маршрути повинні враховувати різну швидкість та грузомісткість машин (агентів). Багатоагентні задачі комівояжера виникають також при складанні маршрутів патрулювання, складанні кільцевих маршрутів для автоматичних транспортних засобів на виробництві, де для запобігання аварій та заторів використовуються непересічні маршрути, зв'язок між якими тем не менш зберігається у вершинах графу (вузлах). Відомі також багатоагентні задачі комівояжера при дизайні мереж, де декілька непересічних мереж захищають мережу від похибок роботи. Двохагентна задача комівояжера виникає і в теорії розкладів [1].

Всі перелічені задачі є NP-складними. Тобто для них не існує або не відомі одночасно швидкі і точні алгоритми розв'язання. Існують або точні методи розв'язання, але вони є лише варіаціями методу прямого перебору, або наближені методи, які базуються на деякої евристиці, тобто на деяких міркуваннях відносно форми розв'язку. Такі міркування можуть бути наслідком досліджень точних розв'язків, які побудовано для відносно простих задач. Прикладами таких міркувань є непересічність маршрутів кожного з агентів, розбиття вузлів на групи, тощо.

Дослідження таких задач і побудова точних або наближених методів розв'язання є актуальною задачею не лише з точки зору прикладних досліджень маршрутизації, але і математичному аспекті.

1.2. Проблеми задач з маршрутизації і огляд літературних джерел

Проблема комівояжера може бути успішно розв'язана перебором маршрутів, якщо кількість пунктів призначення невелика. Принциповою особливістю завдання комівояжера є те, що при збільшенні числа пунктів обчислювальна складність її розв'язання істотно і швидко зростає, і для деякої розмірності задачі вона вже не може бути розв'язана за прийнятний час.

Відомі методи розв’язання: комбінаторний перебір, використання теорії графів, метод гілок і меж, метод динамічного програмування, генетичні методи, алгоритм мурашиної колонії, тощо.

В роботі [2] проведено дослідження публікацій стосуються алгоритму рою частинок. На момент публікації дослідження в 2007 році вже налічувалося понад 1100 робіт на англомовному ресурсі, третина з яких були пропозиціями щодо поліпшення, а решта способами практичного застосування алгоритму, що включають в себе крім комбінаторної оптимізації також біологію і медицину, кластеризації, дизайн, електронні мережі, електроніку , автомобільну промисловість, розваги і т.д. Алгоритм мурашиної колонії розробляється з 1993 року і докладно описується в роботі М. Дориго і Т. Штютцле [3] стосовно до різних завдань і наводиться ефективність алгоритму. Застосування даного алгоритму поширюється і на інші галузі [4]. В останні роки також активно досліджуються багатоагентні підходи до вирішення завдань, в зв'язку з чим з'являються нові алгоритми, такі як бджолиний алгоритм [5] і алгоритм краплі [6, 7]. Метод рою часток, вперше запропонований в 1995 році для імітації соціальної поведінки людей, пізніше був спрощений і застосований для вирішення задачі комівояжера та її узагальнень.

Розподіл задачі між роєм агентів не тільки дозволяє скоротити час пошуку, збільшити ймовірність знаходження глобально кращого рішення, але і є досить актуальною прикладною проблемою. Активна комп'ютеризація та роботизація різних галузей життя створює широкий простір для застосування многоагентних алгоритмів від оптимізації перевезення вантажів до рою бойових роботів. У той же час задачі багатьох комівояжерів приділено менше уваги. Досліджуються генетичні алгоритми для розв’язання цієї задачі, а також перетворення завдання багатьох комівояжерів в кілька звичайних завдань комівояжера. Область застосування таких задач тільки розширюється, оскільки часто при вирішенні питання перевезення вантажів кількість агентів-перевізників більше одного і тоді виникає необхідність розділити, наприклад, місто між агентами якомога ефективніше.

При збільшенні числа комівояжерів виникає проблема поділу графа між ними. Тобто розбиття множини вершин V на k частин таким чином, щоб кожна з частин не містила вершин з інших підмножин, крім точки старту. Розбиття серйозно впливає на якість виконання завдання, оскільки кожен з подграфів ізольований від інших і відповідно не має ребер між цими вершинами. Тому є ймовірність втрати зручних для переходу коротких ребер, відсутність яких унеможливить досягнення глобально найкращого результату. Одним з варіантів розбиття буде кластеризація графа на k кластерів максимально віддалених один від одного. Таким чином, мінімізується шанс втрати скільки-небудь коротких ребер, через безпосереднього відстані між кластерами.

1.3 Цілі та задачі дослідження

Проведений огляд і аналіз наявної інформації дозволив сформувати мету, об'єкт, предмет, завдання і методи дослідження.

Мета дослідження – побудувати алгоритми розв’язання задачі комівояжера для двох агентів, заснований на методі гілок та меж.

Об'єкт дослідження – маршрути двох агентів у графі.

Предмет дослідження – алгоритми оптимізації маршруту БПЛА.

Основні завдання роботи:

- Аналіз інформації по заданій тематиці.
- Системний аналіз об’єкта дослідження.
- Аналіз існуючих алгоритмів розв’язання задач дискретного програмування.
- Формулювання задачі маршрутизації в термінах задач дискретного програмування.
- Створення алгоритму розв’язання.
- Програмна реалізація вказаного алгоритму.
- Аналіз результатів числового дослідження.
- Розрахунок собівартості програмного продукту.

Методи дослідження:

- Інформаційний аналіз.
- Методи цілочислового програмування.
- Математичне моделювання.

2. РОЗГЛЯД ЗАДАЧІ З ТОЧКИ ЗОРУ СИСТЕМНОГО АНАЛІЗУ

2.1 Морфологічний опис системи

До складу системи входять:

1) Кількість агентів (задається швидкість кожного).
2) Стан середовища (атмосфера для БПЛА, дорожній рух у місті для автомобілів і т.п.).
3) Сукупність координат вузлів.
4) Обмеження (пропускна здатність, довжина маршруту, тощо).
5) Цільова функція.

Задача комівояжера має велику кількість різновидів за вказаними вище ознаками та критеріями. Схематично класифікація задач наведена на рис. 2.1.

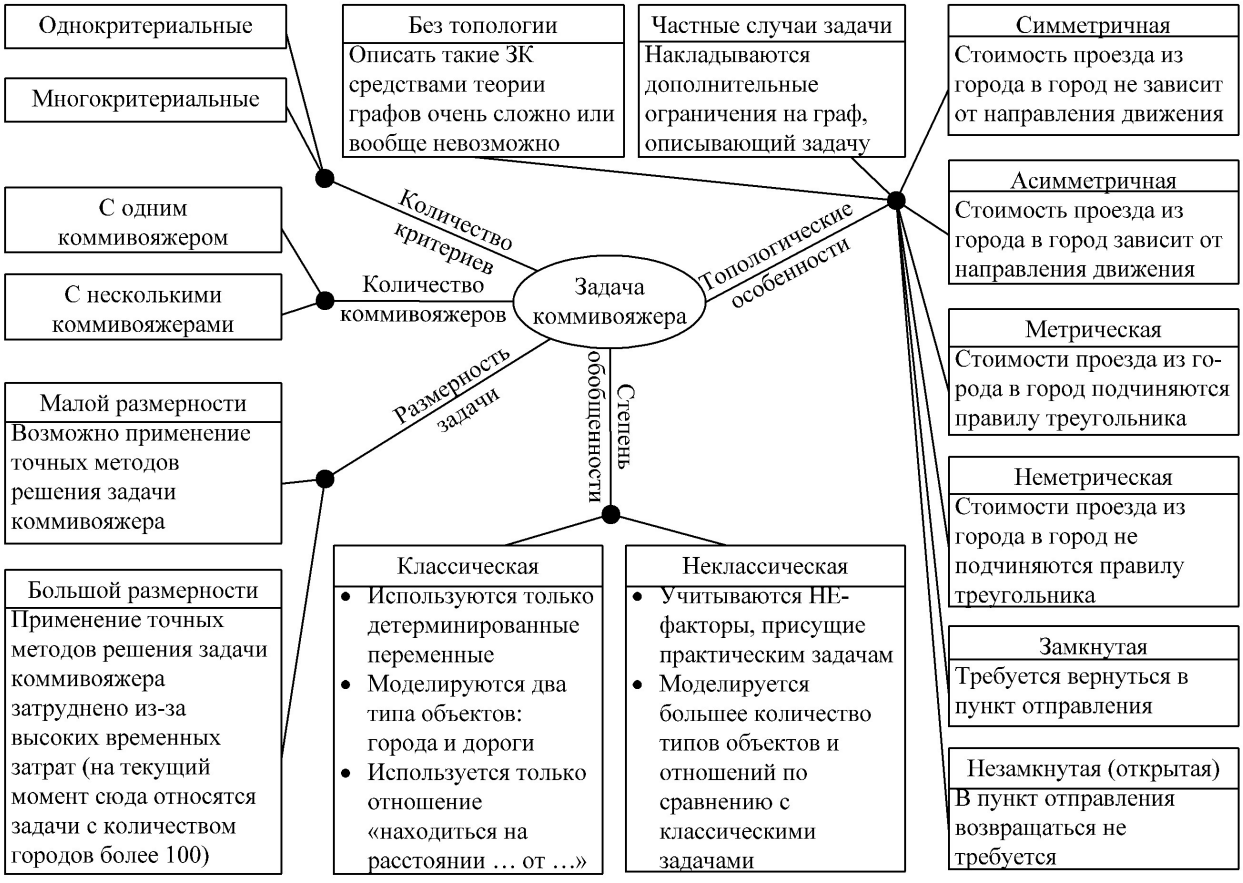

Рисунок 2.1 - Система повітряної розвідки

1) Эмерджентність системи

Системі притаманна емерджентність, тому-що кожен з елементів має певні властивості (якщо не взяти до уваги якийсь фактор, який впливає на

маршрут, то знайдений оптимальний маршрут не буде відповідати дійсності), але тільки система в цілому може знайти оптимальний маршрут в залежності від властивостей елементів системи.

2) Цілістність

Система є цілісною, тому що зміни одного з елементів системи призведе до зміни в усій системі. Наприклад, якщо зміняться швидкість вітру, або властивості одного вузла, то може змінитися весь оптимальний маршрут.

3) Аддитивність

Системі не властива адитивність. Задача не є лінійною і структура розв’язку змінюється з додаванням нового елементу системи.

4) Синергізм

Системі притаманний синергізм, тобто якщо змінити кілька чинників, то їх вплив на систему можна представити у вигляді появи якісно нового. Наприклад, наявність великої швидкості вітру може зробити розв’язок задачі неможливим.

5) Прогресуюча систематизація

Простежується зменшення самостійності об'єктів: кожен фактор, що впливає на маршрут у свою чергу сам складається або залежить від низки факторів, значущість яких зменшується.

2.2 Функціональний опис системи

Функціональне опис системи - це опис як основної функції, виконуваної системою, так і функцій підсистем та елементів даної системи із зазначенням параметрів системи, підсистем і елементів. Функціональне опис також містить структурний опис системи, опис системоруйнівний і системоутворюючих чинників.

Таблиця 2.1 – Функціональний опис системи

Код	Елемент	Функція
1.	Маршрут	Розрахунок маневрів агентів
Функції підсистем другого рівня		
1.1	Агенти	Розрахунок впливу характеристик агентів на маршрути
1.2	Середовище	Розрахунок впливу параметрів середовища на вигляд маршруту.
1.3	Місцевість	Розрахунок впливу координат вузлів на місцевості та елементів місцевості на маршрут
Функції підсистем третього рівня		
1.1.1	Швидкість агентів	Розрахунок часу руху між вузлами
1.1.2	Максимальна дальність польоту	Розрахунок обмежень на довжину маршруту
1.2.1	Швидкість на напрямок вітру. Або наявність пробок на вулицях	Розрахунок часу руху між вузлами
1.3.1	Координати вузлів маршруту	Розрахунок часу рухц між вузлами
1.3.2	Координати старту і фінішу агентів	
1.3.3	Коефіцієнти значущості вузлів	Розрахунок цільової функції ефективності польоту

3. ДВОХАГЕНТНА ЗАДАЧА КОМІВОЯЖЕРА

3.1 Формулювання задача комівояжера для кількох агентів

Припустимо що необхідно по одному разу відвідати групу точок на площині з цілями проведення повітряної розвідки. Для цього застосовується декілька безпілотних літальних апаратів (БПЛА), які розташовано або у **о**дному місті, або у різних місцях. Треба прокласти маршрути кожного БПЛА таким чином, щоб задовольнити обмеження на протяжність польоту кожного БПЛА, максимізувати корисність зібраної інформації та мінімізувати час виконання завдання розвідки. Схема маршрутів показана на рис. 3.1.

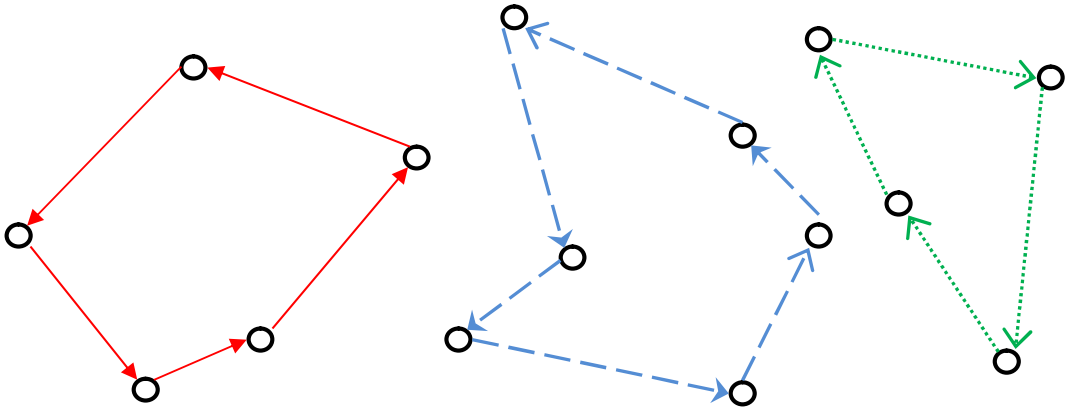

Рисунок 3.1 - Система точок та маршрути трьох БПЛА

Підкреслимо, що в даній постановці БПЛА (агенти) не конкурують між собою, польотне завдання ставиться перед ними перед виконанням польоту і не змінюється у польоті за рахунок нової інформації. То, що агенти не конкурують між собою, суттєво відрізняє цю задачу від низки аналогічних задач, де агенти конкурують і формує відповідним чином критерій оптимальності [7, 8].

Розглянемо задачу побудови маршрутів для **двох** літальних апаратів (агентів).

Маємо набір із N вузлів (точок), які треба облетіти БПЛА. Кожний вузол має свій коефіцієнт корисності p_i, $i = 1,2,...,N$. Маємо також набір часу у

польоті між вузлами i та j першого та другого БПЛА, які позначимо як $a_{i,j}$ та $b_{i,j}$. Зазвичай за наявності вітру $a_{i,j} \neq a_{j,i}$, тобто матриці $\{a_{i,j}\}$ та $\{b_{i,j}\}$ не симетричні. , Ця умова виключає виникнення двох альтернативних розв'язків, які описують один і теж маршрут, але у протилежних напрямках. І тому введення вітру в модель значно зменшує ризик виникнення альтернативних розв'язків і є одним з факторів, що зумовлюють однину розв'язку.

Для формалізації запису задачі введемо булеві змінні, які можуть приймати значення лише 0 та 1. А саме

$$x_{i,j} = \begin{cases} 0, & \text{якщо перший БПЛА не потрапив із вузла } i \text{ в } j \\ 1, & \text{якщо перший БПЛА потрапив із вузла } i \text{ в } j \end{cases}$$

$$y_{i,j} = \begin{cases} 0, & \text{якщо другий БПЛА не потрапив із вузла } i \text{ в } j \\ 1, & \text{якщо другий БПЛА потрапив із вузла } i \text{ в } j \end{cases}$$

Умова про відвідування кожного вузла з номером j лише один раз, або жодного разу, має вигляд

$$\sum_{i=1}^{N} x_{i,j} + \sum_{i=1}^{N} y_{i,j} = 1, \quad j = 1,2,...,N,\ i \neq j. \tag{3.1}$$

Аналогічна умова про виліт з кожного вузла лише один раз чи жодного разу має вигляд

$$\sum_{j=1}^{N} x_{i,j} + \sum_{j=1}^{N} y_{i,j} = 1, \quad i = 1,2,...,N,\ j \neq i. \tag{3.2}$$

Сума $x_{i,j}$ за індексом i означає кількість відвідувань вузла з номером j першим БПЛА зі всіх інших вузлів, а сума $y_{i,j}$ за індексом i означає кількість відвідувань вузла з номером j другим БПЛА.

Зауважимо, що умови (3.1) і (3.2) не гарантують того, що знайдені розв'язки будуть циклами, вони лише гарантують що кожний пункт буде відвідано агентами лише один раз. Томі ці умови слід доповнити ще умовами

$$\sum_{i=1}^{N} x_{i,j} - \sum_{k=1}^{N} x_{j,k} = 0, \quad j = 1,2,...,N,\ i \neq j. \tag{3.3}$$

$$\sum_{i=1}^{N} y_{i,j} - \sum_{k=1}^{N} y_{j,k} = 0, \quad j = 1,2,...,N, \; i \neq j \; . \tag{3.4}$$

Ці умови вказують на те, що кожний пункт відвідано та залишено однакову кількість разів. В сукупності з обмеженнями на зміні $x_{i,j}$, $y_{i,j}$, а саме на те, що змінні приймають значення 0 або 1, з (3.3) та (3.4) слідує, що кожний вузол або не відвідується даним агентом взагалі, або відвідується один раз та один за покинуто.

Крім того, вводиться ще умова на то що кожні **ненульові** змінні $x_{i,j}$ та $y_{i,j}$ утворюють **цикл**, який містить точку старту та посадки. Цикл для кожної зі змінних повинен бути єдиним, бо умови (3), (4) не гарантують того що цикл буде містити лише один замкнений маршрут, а не кілька кіл, на кшталт маршруту $x_{1,2}$, $x_{2,1}$, $x_{3,4}$, $x_{4,3}$.

Тобто

$$\begin{matrix} \{x_{i,j}\} - \text{один цикл, де } x_{i,j} = 1 \\ \{y_{i,j}\} - \text{один цикл, де } y_{i,j} = 1 \end{matrix} . \tag{3.5}$$

Цільова функція – це час виконання загального завдання, даного для двох агентів, якій потрібно мінімізувати. Але оскільки ми маємо два апарати, то треба прокласти маршрут кожного з апаратів таким чином, щоб **мінімізувати максимальний час** у польоті кожного з двох БПЛА, тобто

$$\max\left\{\left(\sum_{j=1}^{N}\sum_{i=1}^{N} a_{i,j}x_{i,j}\right),\left(\sum_{j=1}^{N}\sum_{i=1}^{N} b_{i,j}y_{i,j}\right)\right\} \to \min \tag{3.6}$$

Такий вигляд цільової функції зумовлений тим, що час виконання польотного завдання визначається лише часом виконання свого завдання тим з двох апаратом, який повернувся в точку старту останнім. А час виконання свого завдання апаратом, який повернувся першим – для нас не важливий.

Якщо позначити час виконання свого завдання першим апаратом T_1, а другим апаратом - T_2, то цільова функція набуває вигляду

$$F = \max\{T_1, T_2\} \to \min .$$

На рис. 3.2 показано графіки ліній рівня цієї функції в координатах T_1 і T_2.

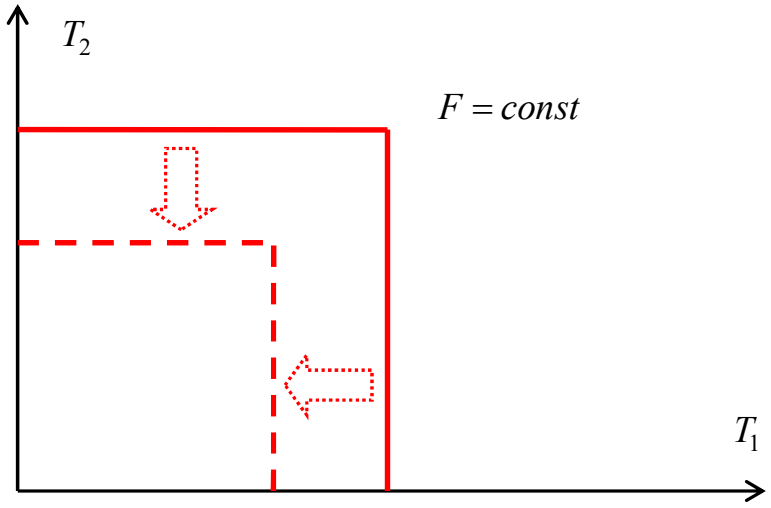

Рисунок 3.2 - Область оптимальності розв'язку

3.2 Зведення задачі до задачі квадратичного програмування

Вочевидь така цільова функція є нелінійною. Тому для розв'язання поставленої задачі потрібно або будувати ітераційний алгоритм який дозволяє на кожній ітерації коректувати цільову функцію и зводити задачу до лінійної, або замінити цільову функцію нелінійною, яка за можливості близька до даної нелінійної функції. Найбільш простий вигляд такої функції має квадратична цільова функція. Це зумовлено тим, що для побудови розв'язків задач квадратичного програмування розроблений симплекс-алгоритм, тобто ітераційна процедура яка за скінчену кількість ітерацій будує розв'язок відповідної задачі квадратичного програмування. Тому обираємо саме таку структуру цільової функції.

Інтуїтивно можна припустити, що якщо обрано маршрут, який включає більше вузлів для першого БПЛА, ніж для другого, то $T_1 > T_2$. І при зменшенні кількості вузлів для першого БПЛА і збільшенні для другого (оптимальним чином, безумовно!), T_1 буде зменшуватися, а T_2 - зростати. Тобто для

оптимальних маршрутів для кожного набору точок має наступна залежність T_1 і T_2, що показана на рис. 3.3:

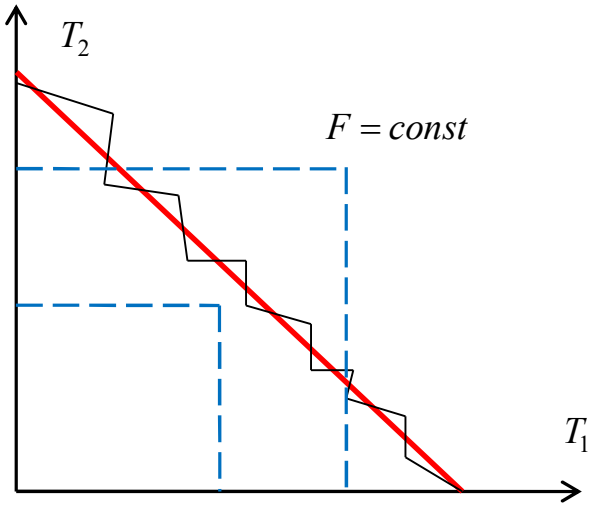

Рисунок 3.3 – Область оптимальності розв'язку

Ламана лінія ілюструє реальний розподіл точок та реальний вигляд графіку залежності T_1 і T_2, пряма лінія – ідеалізований вигляд графіку, а штрихові лінії – це графіки функцій $F = const$. З даної ілюстрації можна помітити, що при оптимальних маршрутах для кожного з БПЛА час виконання відповідного завдання кожним з них буде майже однаковим. Тобто різниця $|T_1 - T_2|$ буде малою. Але взяти цю умову як цільову функцію неможливо, то що існує велика кількість **неоптимальних маршрутів** які мало відрізняються за часом. Навіть може існувати такі неоптимальні маршрути, для яких різниця $|T_1 - T_2|$ буде меншою, ніж для оптимальних маршрутів.

Тобто необхідно мінімізувати $|T_1 - T_2|$ за умови мінімуму T_1 і T_2. Тому пропонується призначити цільову функцію наступного вигляду:

$$F = T_1^2 + T_2^2 + k\left(T_1 - T_1\right)^2 \to \min \tag{3.7}$$

Доданок $\left(T_1 - T_1\right)^2$ вказує на необхідність зменшення різниці $|T_1 - T_2|$, а сума квадратів T_1 і T_2 - вказує на необхідність зменшення T_1 і T_2. Коефіцієнт k, що входить до формули (3.7) слугує для налаштування алгоритму и підбирається емпіричним шляхом.

На рис. 4 показано графіки ліній рівня цільових функцій (3.7) для $k = 0{,}8$ та $k = 1{,}4$ і ілюструє вплив цього коефіцієнта на вигляд ліній рівня.

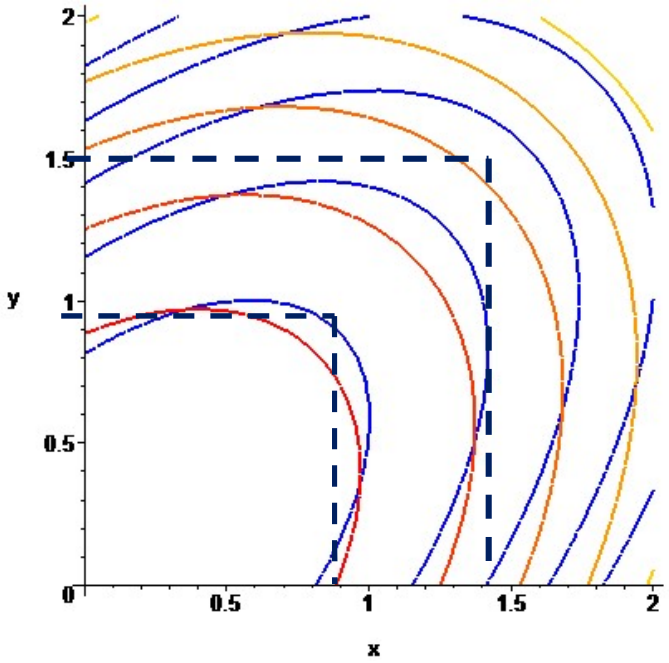

Рис. 4. Лінії рівня квадратичної цільової функції

Як бачимо, графіки близькі до показаних на рис. 2. Керуючи параметром k можна налаштувати алгоритм для покращення його роботи.

Таким чином, задачу можна сформулювати наступним чином:

$$F = \left(\sum_{j=1}^{N} \sum_{i=1}^{N} a_{i,j} x_{i,j} \right)^2 + \left(\sum_{j=1}^{N} \sum_{i=1}^{N} b_{i,j} y_{i,j} \right)^2 + k \left(\sum_{j=1}^{N} \sum_{i=1}^{N} a_{i,j} x_{i,j} - \sum_{j=1}^{N} \sum_{i=1}^{N} b_{i,j} y_{i,j} \right)^2 \to \min$$

За умов:

$$x_{i,j} = \{0;\ 1\}; \quad y_{i,j} = \{0;\ 1\};$$

$$\sum_{i=1}^{N} x_{i,j} + \sum_{i=1}^{N} y_{i,j} = 1, \quad j = 1,2,...,N,\ i \neq j.$$

$$\sum_{j=1}^{N} x_{i,j} + \sum_{j=1}^{N} y_{i,j} = 1, \quad i = 1,2,...,N,\ j \neq i.$$

$$\sum_{i=1}^{N} x_{i,j} - \sum_{k=1}^{N} x_{j,k} = 0, \quad j = 1,2,...,N,\ i \neq j.$$

$$\sum_{i=1}^{N} y_{i,j} - \sum_{k=1}^{N} y_{j,k} = 0, \quad j = 1,2,...,N,\ i \neq j$$

$$\{x_{i,j}\} - \text{один цикл, для } x_{i,j} = 1$$
$$\{y_{i,j}\} - \text{один цикл, для } y_{i,j} = 1$$

Тобто маємо **задачу квадратичного бінарного програмування**. Розв'язання якої може викликати не аби які труднощі. Це пов'язано з неможливістю застосувати градієнтні методи (внаслідок дискретності змінних) та класичний симплекс-метод для розв'язання задачі дискретного програмування (внаслідок нелінійності цільової функції).

3.3. Модифікація задачі та застосування лінійного програмування

Як відмічено вище, розв'язання задачі квадратичного бінарного програмування може викликати не аби які труднощі. Тому доцільно звести задачу комівояжера для двох БПЛА до задачі лінійного бінарного програмування. Вигляд цільової функції, показаний на рис. 3.2 дозволяє це зробити. Для цього можна цільову функцію (3.6) розкласти наступним чином:

$$\begin{aligned} F &= \sum_{j=1}^{N}\sum_{i=1}^{N} a_{i,j}x_{i,j} \to \min \\ \sum_{j=1}^{N}\sum_{i=1}^{N} b_{i,j}y_{i,j} &\le \sum_{j=1}^{N}\sum_{i=1}^{N} a_{i,j}x_{i,j} \end{aligned} \tag{3.8}$$

або

$$\begin{aligned} F &= \sum_{j=1}^{N}\sum_{i=1}^{N} b_{i,j}y_{i,j} \to \min \\ \sum_{j=1}^{N}\sum_{i=1}^{N} b_{i,j}y_{i,j} &\ge \sum_{j=1}^{N}\sum_{i=1}^{N} a_{i,j}x_{i,j} \end{aligned} \tag{3.9}$$

Сформулювати таку постановку можна як зменшення часу у польоті одного з БПЛА за умов що інший БПЛА пробуде у польоті ще менше або такий же час.

Цей підхід ілюструють рис. 3.5.

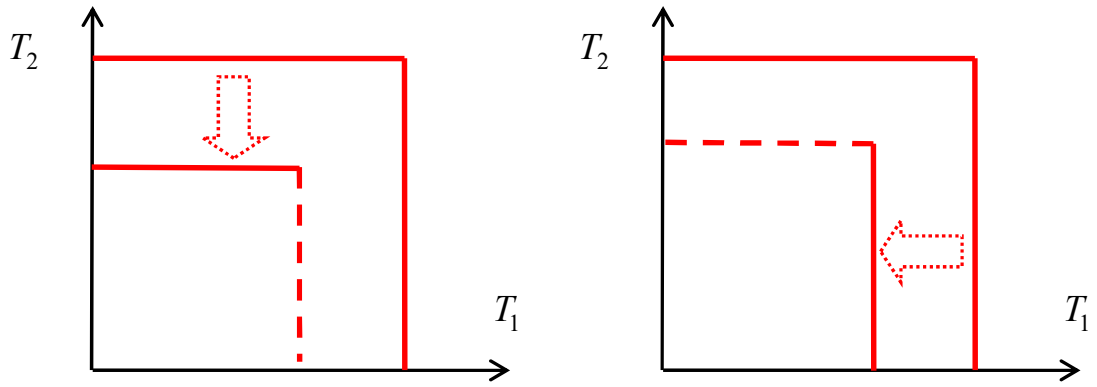

Рисунок 3.5 – Області оптимальності розв'язку

Таким чином, задачу можна сформулювати у двох постановках, які розрізняються змінними що входять до цільової функції та одним обмеженням:

$$F = \sum_{j=1}^{N}\sum_{i=1}^{N} a_{i,j}x_{i,j} \to \min$$

За умов

$$-\sum_{j=1}^{N}\sum_{i=1}^{N} a_{i,j}x_{i,j} + \sum_{j=1}^{N}\sum_{i=1}^{N} b_{i,j}y_{i,j} \le 0$$

$$\sum_{i=1}^{N} x_{i,j} + \sum_{i=1}^{N} y_{i,j} = 1, \quad j = 1,2,...,N,\ i \ne j.$$

$$\sum_{j=1}^{N} x_{i,j} + \sum_{j=1}^{N} y_{i,j} = 1, \quad i = 1,2,...,N,\ j \ne i.$$

$$\sum_{i=1}^{N} x_{i,j} - \sum_{k=1}^{N} x_{j,k} = 0, \quad j = 1,2,...,N,\ i \ne j.$$

$$\sum_{i=1}^{N} y_{i,j} - \sum_{k=1}^{N} y_{j,k} = 0, \quad j = 1,2,...,N,\ i \ne j$$

$x_{i,j} = \{0;\ 1\}$; $y_{i,j} = \{0;\ 1\}$ - бінарні змінні;

$\{x_{i,j}\}$ – утворюють один цикл, для $x_{i,j} = 1$

$\{y_{i,j}\}$ – утворюють один цикл, для $y_{i,j} = 1$

Та ще одну задачу:

$$F = \sum_{j=1}^{N}\sum_{i=1}^{N} b_{i,j} y_{i,j} \to \min$$

За умов

$$\sum_{j=1}^{N}\sum_{i=1}^{N} a_{i,j} x_{i,j} - \sum_{j=1}^{N}\sum_{i=1}^{N} b_{i,j} y_{i,j} \le 0$$

$$\sum_{i=1}^{N} x_{i,j} + \sum_{i=1}^{N} y_{i,j} = 1, \quad j = 1,2,...,N,\ i \ne j.$$

$$\sum_{j=1}^{N} x_{i,j} + \sum_{j=1}^{N} y_{i,j} = 1, \quad i = 1,2,...,N,\ j \ne i.$$

$$\sum_{i=1}^{N} x_{i,j} - \sum_{k=1}^{N} x_{j,k} = 0, \quad j = 1,2,...,N,\ i \ne j.$$

$$\sum_{i=1}^{N} y_{i,j} - \sum_{k=1}^{N} y_{j,k} = 0, \quad j = 1,2,...,N,\ i \ne j$$

$x_{i,j} = \{0;\ 1\}$; $y_{i,j} = \{0;\ 1\}$ - бінарні змінні;

$\{x_{i,j}\}$ – утворюють один цикл, для $x_{i,j} = 1$

$\{y_{i,j}\}$ – утворюють один цикл, для $y_{i,j} = 1$

Яку ж форму цільової функції та обмеження застосовувати, (3.6) чи (3.7) ? Формально змінні рівноправні, і заздалегідь невідомо, час польоту якого БПЛА буде більшим, я кого – меншим. Тому найкращім варіантом буде **розв'язання двох задач за критеріями (**3.**8) та (**3.**9)** відповідно і **вибір найкращого розв'язку** з двох отриманих.

Ще один можливий варіант – розв'язувати задачу за допомогою одного і того алгоритму (програми), два рази, змінюючи властивості кожного з агентів відповідним чином. Тобто у перший раз коефіцієнти $a_{i,j}$ розраховуються для першого з агентів, $b_{i,j}$ - для другого агента. А на другому разі $a_{i,j}$

розраховуються виходячи зі швидкості другого агента, а $b_{i,j}$ - для першого агента.

Для розв'язання задачі дискретного лінійного програмування зазвичай застосовується метод гілок та границь [11-21]. Але умова циклічності розв'язку ускладнює класичну задачу призначень. Для розв'язання таких задач, які загалом мають зазву «задачі комівояжера» існують модифікації методу гілок та границь, де на кожній ітерації розриваються заборонені підцикли [20]. Саме такий підхід і взятий за основу для створення алгоритму розв'язання задачі комівояжера для двох агентів.

Розглянемо алгоритм, заснований на схемі (3.8). Для цього спочатку введемо поняття **основного циклу.** Так будемо звати два підцили (які становляться циклами на фінальному етапі) для $x_{i,j}$ та $y_{i,j}$, які містять точки старту та посадки першого та другого БПЛА відповідно.

На чергової ітерації k алгоритму виконуються наступні кроки:

Крок 1. Припинити розрахунки якщо основний список задач порожній. У протилежному випадку взяти одну з задач (задача задається двома матрицями $\{a_{i,j}\}$, $\{b_{i,j}\}$), виключивши її з основного списку.

Крок 2. Розв'язати відповідну задачу бінарного програмування (3.8), (3.1), (3.2). Якщо отримане оптимальне значення цільової функції більше чи дорівнює поточному значенню цільової функції L то повернутися до ***Кроку 1***, не змінюючи L. (Початкове значення L на першій ітерації може дорівнювати будь-якому великому числу). У протилежному випадку перейти до ***Кроку 3***.

Крок 3. Якщо отриманий оптимальний розв'язок задачі (3.1), (3.2), (3.8) складається з двох **основних циклів** (для $x_{i,j}$ та $y_{i,j}$ відповідно), то запам'ятати його, присвоїти L значення цільової функції, яке отримане на даній ітерації і повернутися до ***Кроку 1***. Якщо отриманий розв'язок окрім основних циклів містить ще підцикли – перейти до ***Кроку 4***.

Крок 4. В отриманому розв'язку знайти найменші підцикли для обох БПЛА, тобто підцикли, які містять найменшу кількість відрізків (якщо маршрут одного з агентів складається лише з основного циклу – то обрати лише один підцикл у маршруті для іншого БПЛА). Кожному $x_{i,j}=1$ і $y_{i,j}=1$, які входять до даних підциклів поставити у відповідність відповідну задачу бінарного програмування, додавши її до основного списку і прийняв відповідні значення в матриці часу перельотів $a_{i,j}=\infty$ та $b_{i,j}=\infty$ (або завеликому числу). А всі інші коефіцієнти залишити незмінними, тобто такими ж, як і задачі, обраній на кроці 1. Перейти до ***Кроку 1***.

Наприклад, якщо ми маємо підцикли $\{x_{3,4},\ x_{4,3}\}$ та $\{y_{10,12},\ y_{10,12}\}$ то на кроці 4 алгоритму маємо додати до основного списку чотири задачі, в яких $a_{3,4}=\infty, b_{10,12}=\infty$, $a_{3,4}=\infty, b_{12,10}=\infty$, $a_{4,3}=\infty, b_{10,12}=\infty$ та $a_{4,3}=\infty, b_{12,10}=\infty$.

Другий алгоритм відрізняється від наведеного вище лише тим, що замість умови (3.8) застосовується умова та цільова функція (3.9). При розв'язанні задачі потрібно розв'язати за обома алгоритмами, та обрати найкращій з двох отриманих розв'язків. Або розв'язати задачу двічі за одним алгоритмом, змінивши на другому разі позначення агентів.

Слід відмітити, що основний список містить набори матриць $\left[\{a_{i,j}\},\ \{b_{i,j}\}\right]$. Таких задач основний список може містити сотні, що призводить до швидкого використання всієї оперативної пам'яті комп'ютера. Але ці матриці відстаней для кожній задачі відрізняються лише декількома елементами, яким замість реальній відстані присвоєне завелике значення для заборони попадання відповідного відрізку у оптимальний розв'язок. Тому для зменшення використання ресурсів доцільно зберігати не самі ці матриці $\left[\{a_{i,j}\},\ \{b_{i,j}\}\right]$ а лише списки елементів для кожної матриці, які треба замінити на завеликі числа. Для цього треба пронумерувати елементи матриць від 1 до N^2 і

зберігати лише номера елементів у списках. При цьому списки мають бути нефіксованого розміру і містити від 1 елементу до N^2. А на кожній ітерації за цим списком відповідною процедурою слід створювати транспортні матриці $\{a_{i,j}\}$, $\{b_{i,j}\}$ шляхом відповідної модифікації початкових матриць. У програмі реалізований саме такий підхід.

3.4 Модифікація матриці відстаней для розімкненого маршруту

В деяких випадках виникає задача побудови найкоротшого маршруту обльоту системи точок, в якої точка старту і точка посадки ПБЛА не співпадають, тобто маршрут є розімкненим.

Така задача виникає за умов, наприклад, значної довжини маршруту, коли БПЛА не може повернутися до точки старту внаслідок своїх технічних характеристик. Або у випадках коли точка старту стає відомою противнику і противник може чекати повернення БПЛА з метою його знищення. У такому випадку доцільно здійснити посадку БПЛА на контрольованій території, але без повернення до точки старту.

Розв’язання цієї задачі у такий постановці можливо без зайвих труднощів за допомогою зведення її до розглянутої вище двохагентної задачі комівояжера. Нехай, наприклад, точкою старту першого агенту є вузол з номером s, а точкою посадки є точка з номером f. У цьому разі політ починається з точки s і завершується точкою f. Щоб алгоритм провів розрахунок маршруту необхідно у матриці відстаней першого агенту a_{ij} відповідний час $a_{f,s}$ дорівняти нулю. Крім того необхідно заборонити всі маршрути, які виходять с точки f, для чого час відповідних маршрутів призначається достатньо великим. І також необхідно заборонити всі маршрути яки входять в вузол s. Тобто призначаємо наступні значення елементам матриці:

$$a_{f,s} = 0;$$

$$a_{f,i} = \infty, \quad i = 1,...,N, \; i \neq s;$$

$$a_{i,s} = \infty, \quad i = 1,...,N, \; i \neq f;$$

Ці параметри забезпечують обов'язковий виліт першого БПЛА з точки s, проліт до точки f, та замикання маршруту через відрізок (f,s), який дорівнює нулю. Це ілюструє рис. 3.6.

Аналогічним чином у разі потреби модифікується матриця відстаней другого агенту $b_{f,s}$.

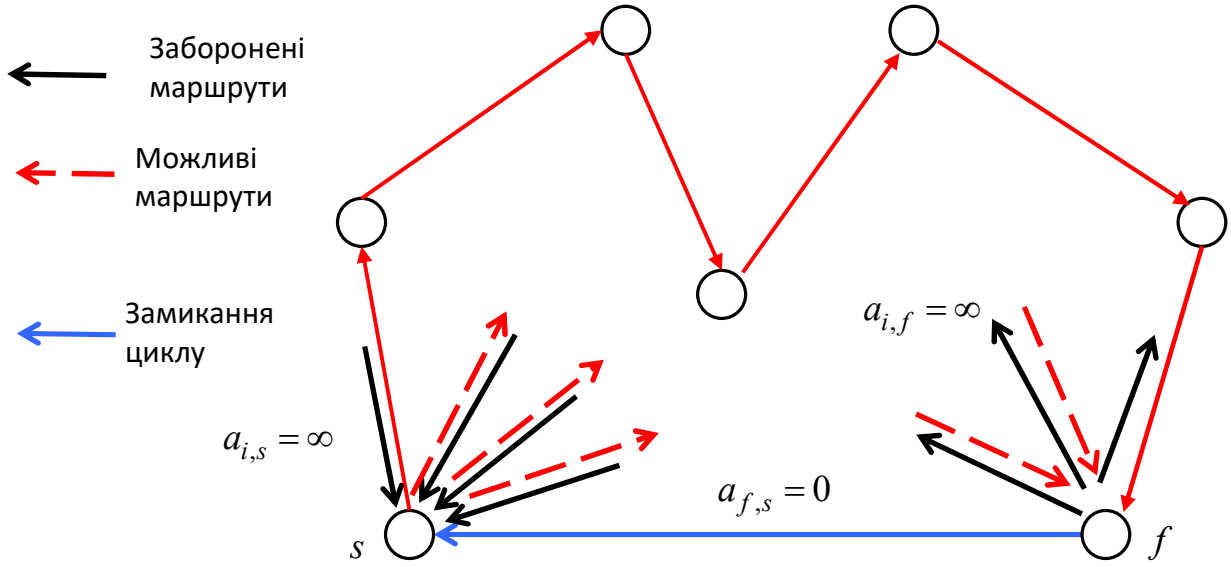

Рисунок 3.6 - Схема замикання маршруту з фіксованою точкою старту і посадки

3.5 Модифікація обмежень для забезпечення базування двох агентів у спільному вузлу

Часто виникає потреба в побудові розв'язку задачі, де один з вузлів є «базою», а бо «складом» спільним для обох комівояжерів (агентів). Схема таких маршрутів показана на Рис. 3.7.

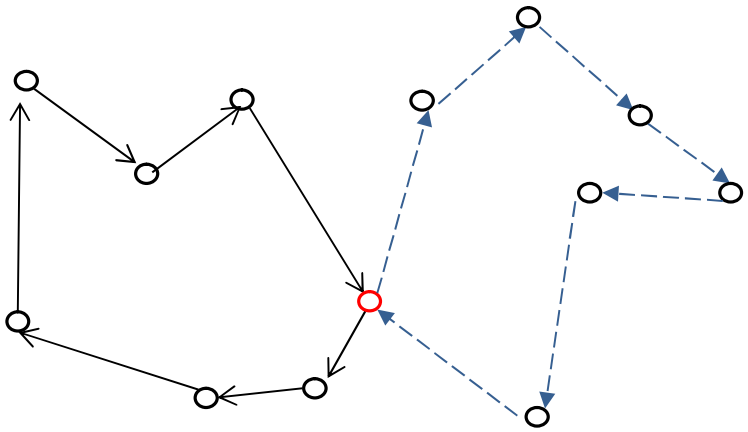

Рисунок 3.7 – Схема маршрутів зі спільною точкою

Як бачимо, обидва маршрути мають одну спільну точку – заданий в умовах задачі вузол. Дана постановка задачі також виникає при розробці маршрутів двох БПЛА у випадку старту та посадки обох БПЛА в однієї точці, що є поширеною практикою застосування даної техніки.

Така постановка задачі потребує відповідної зміни обмежень на кількість відвідувань вузлів. Це зумовлено тим, що у даний вузол прибивають та покидають обидві агенти-комівояжери. Зрозуміло, що вузлі, який є «базою» для обох агентів умови (3.3) та (3.4) повинні виконуватися. Але умови (3.1) і (3.2) мають бути змінені. Якщо вузол, який є «базою» для двох агентів має номер M (нумерація вузлів довільна і на розв'язок не впливає), то умови (3.1) і (3.2) набудуть вигляду

$$\sum_{i=1}^{N} x_{i,M} + \sum_{i=1}^{N} y_{i,M} = 2, \qquad \sum_{j=1}^{N} x_{M,j} + \sum_{j=1}^{N} y_{M,j} = 2.$$

Ці умови вказують на той факт, що кількість відвідувань і від'їздів агентів з вузла M дорівнює двом. Однак з того факту, що змінні $x_{i,j}$ та $y_{i,j}$ є бінарними, тобто дорівнюють або нулю або одиниці, а також з умов (3.3) і (3.4) випливає що обидва агенти мають відвідати та покинути вузол M.

Але ці умови не виключають можливості пересічення двох циклів одного агента у вказаній точці. Для виключення цієї можливості слід додати ще

одну умову. Умову про те, що один з агентів входить або виходить зі вказаного вузла лише один раз. Наприклад умову вигляду

$$\sum_{i=1}^{N} x_{i,M} = 1.$$

При цьому умови $\sum_{i=1}^{N} y_{i,M} = 1$, $\sum_{i=1}^{N} x_{M,i} = 1$, $\sum_{i=1}^{N} y_{M,i} = 1$ будуть виконуватися автоматично.

Як бачимо, наявність спільної «бази» для обох агентів потребує лише зміни правої частини однієї умов (3.3) і (3.4) і введення однієї додаткової умови. Вочевидь така заміна у крайових умовах дозволить створити не тільки одну «базу», а декілька.

У разі виникнення задачі про пошук оптимальної точки старту, яка обирається з набору заданих точок, то вірогідніше за все слід розв’язати відповідні задачі для кожної точки, а потім обрати найкращий розв’язок.

3.6 Розрахунок часу перельоту між двома вузлами маршруту

Особливістю обговорюваної задачі маршрутизації є необхідність врахування вітру, оскільки повітряна швидкість апаратів розглянутого типу порівняно невелика. Одночасно з цим для них характерні невеликий радіус дії і нетривалий за часом політ. Це дозволяє використовувати в якості прогностичної моделі вітру такий її найпростіший варіант, коли напрямок і швидкість вітру покладаються постійними для всієї зони і часу польоту і відповідними характеристиками вітру в точці старту. Дана задача досліджена у роботах [11, 12] і наведені нижче формули було отримано у вказаних роботах.

Задача побудови оптимального маршруту двох БПЛА припускає апріорне знання швидкості і напрямку вітру на маршруті, а також швидкості кожного з двох агентів відносно нерухомого повітря. При наземної підготовки маршруту до уваги беруться параметри вітру на висоті польоту, прогнозовані в залежності від швидкості вітру на поверхні землі. Ці оцінки можуть

відрізнятися від реальних внаслідок, по-перше, впливу характеру земної поверхні, по-друге, відмінності реального стану атмосфери, наприклад градієнта температури по висоті, від прогнозованого. Це викликає необхідність корегування маршруту в процесі польоту на основі знов отриманих оцінок параметрів вітру. Величина і напрямок швидкості вітру можуть визначатися на основі прямого виміру вектора земної швидкості БПЛА бортової навігаційної системою (курсового кута і величини земної швидкості), і вектора повітряної швидкості (кута рискання і величини повітряної швидкості), які спостерігаються бортовою інерціальною системою. Або ж можуть використовуватися дані метеорологічних служб, в тому числі і міжнародних.

Напрямок від точки А до точки В збігається з напрямком вектора $AB = \vec{B} - \vec{A} = \begin{pmatrix} x_B - x_A \\ y_B - y_A \end{pmatrix}$. Відстань між точками А і В одно модулю вектора АВ, $|AB| = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$. Вітер буде описуватися вектором, модуль якого дорівнює швидкості вітру, а його напрямок відповідає напрямку вітру.

На площині, яка визначається системою координат Оху, напрямок вітру будемо задавати кутом α. кут α відраховується від осі Ох в напрямку проти годинникової стрілки в межах від 0 до 180 градусів і в напрямку за годинниковою стрілкою в межах від 0 до -180 градусів. Таким чином, якщо α становить 180 або -180 градусів, то мова йде про одне й те ж напрямку вітру - проти осі Ох. Проекції вектора $\vec{V}_0$ на осі Ох і Оу визначаються відповідно за формулами.

$$V_{0x} = \left|\vec{V}_0\right| \cos\alpha \,,\; V_{0y} = \left|\vec{V}_0\right| \sin\alpha \,.$$

Елемент *i, j* матриці комівояжера дорівнює часу, яке витратить БПЛА на переліт з точки *i* в точку *j* в умовах дії вітру. Діагональні елементи матриці покладаються рівними нескінченності.

Особливістю матриці комівояжера, сформованої з урахуванням дії вітру, є те, що вона виявляється несиметричною. Іншими словами, елемент *i, j* такий

матриці не дорівнює елементу *j, i.* Це зрозуміло, оскільки вітер для будь-якої пари об'єктів виявляється в одному напрямку «попутним», а в зворотному «зустрічним». Особливим випадком є траверзное до вектору АВ напрямок вітру, при якому наддіагональний елемент матриці комівояжера дорівнює відповідному поддіагональному елементу. При розрахунку елемента *i, j* матриці комівояжера враховується, що вектор швидкості руху апарату розгорнуто щодо лінії, що з'єднує об'єкт *i* з об'єктом *j* на певний кут. Величина цього кута знаходиться з умови «попадання» з точки *i* в точку *j* з урахуванням дії вітру.

Розглянемо докладніше переліт з точки А в точку В з урахуванням дії вітру (рис. 3.8):

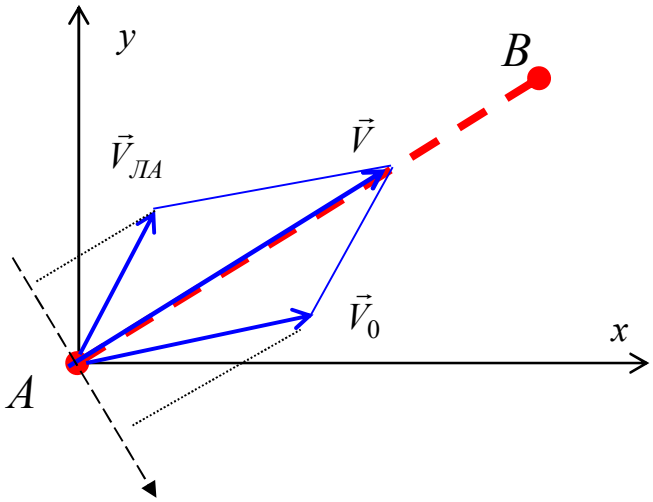

Рисунок 3.8 - Діаграма швидкостей

Введемо наступні позначення:

$\vec{V}_{ЛА}$ – вектор швидкості БПЛА відносно маси повітря;

$\vec{V}_0$ – вектор швидкості вітру (повітряної маси) відносно земної поверхні;

$\vec{V}$ – вектор швидкості БПЛА відносно земної поверхні.

Таким чином $\vec{V}$ - це швидкість абсолютна; $\vec{V}_{ЛА}$ - переносна; $\vec{V}_0$ - відносна. Вочевидь

$$\vec{V} = \vec{V}_0 + \vec{V}_{ЛА}.$$

Еквівалентом даного виразу є два співвідношення. Перше визначає модуль вектору $\left|\vec{V}\right|$ як суму проекції векторів $\vec{V}_{ЛА}$ і $\vec{V}_0$ на напрям вектору AB.

$$\left|\vec{V}\right| = Пр_{AB}\vec{V}_{ЛА} + Пр_{AB}\vec{V}_0 . \qquad (3.10)$$

Друге співвідношення виражає ту обставину, що проекції векторів $\vec{V}_{ЛА}$ і $\vec{V}_0$ на напрям, що визначається вектором, перпендикулярним до вектора AB, і який позначимо $\perp AB$, рівні за величиною і протилежні за знаком:

$$Пр_{\perp AB}\vec{V}_{ЛА} + Пр_{\perp AB}\vec{V}_0 = 0 . \qquad (3.11)$$

Другий доданок (3.6) визначається за формулою

$$Пр_{AB}\vec{V}_0 = \frac{\left(AB, \vec{V}_0\right)}{\left|AB\right|} = \frac{V_{0,x}x_{AB} + V_{0,y}y_{AB}}{\left|AB\right|} .$$

При визначені другого доданку (3.11) враховуємо, що

$$V_{ЛА}^2 = \left(Пр_{AB}\vec{V}_{ЛА}\right)^2 + \left(Пр_{\perp AB}\vec{V}_{ЛА}\right)^2 ,$$

Звідки знаходимо $Пр_{AB}\vec{V}_{ЛА} = \sqrt{V_{ЛА}^2 - \left(Пр_{\perp AB}\vec{V}_{ЛА}\right)^2}$, або з врахуванням (3.11),

$$Пр_{AB}\vec{V}_{ЛА} = \sqrt{V_{ЛА}^2 - \left(Пр_{\perp AB}\vec{V}_0\right)^2} .$$

Невідомий доданок під коренем можна визначити за формулою

$$Пр_{\perp AB}\vec{V} = \frac{\left(\vec{V}_0, \perp AB\right)}{\left|\perp AB\right|} = \frac{V_{0,x}x_{\perp AB} + V_{0,y}y_{\perp AB}}{\left|\perp AB\right|} .$$

Де $\perp AB = \begin{pmatrix} x_{\perp AB} \\ y_{\perp AB} \end{pmatrix}$ - вектор з довільним модулем $\left|\perp AB\right|$, перпендикулярний до вектору AB. Фактично вектор $\perp AB$ потрібен тільки для завдання напряму, перпендикулярного вектору AB. Координаті $x_{\perp AB}, y_{\perp AB}$ визначимо з врахуванням ортогональності векторів AB і $\perp AB$.

$$x_{AB} \cdot x_{\perp AB} + y_{AB} \cdot y_{\perp AB} = 0 .$$

Оскільки модуль вектору $\perp AB$ може бути довільним (важливий лише його напрямок), то будемо вважати $x_{\perp AB} = 1$. Тоді умова ортогональності набуде вигляду

$$y_{\perp AB} = -\frac{x_{AB}}{y_{AB}}.$$

Таким чином, вектор $\perp AB$ заданий, і формулу проекції можна записати у вигляді

$$Пр_{AB}\vec{V}_0 = \frac{V_{0,x} - V_{0,y}\dfrac{x_{AB}}{y_{AB}}}{\sqrt{1+\left(\dfrac{x_{AB}}{y_{AB}}\right)^2}}.$$

З врахуванням отриманого результату, формула проекції швидкості літального апарату на напрямок АВ набуде вигляду

$$Пр_{AB}\vec{V}_{ЛА} = \sqrt{V_{ЛА}^2 - \left(Пр_{\perp AB}\vec{V}_0\right)^2} = \sqrt{V_{ЛА}^2 - \frac{\left(V_{0,x} - V_{0,y}\dfrac{x_{AB}}{y_{AB}}\right)^2}{1+\left(\dfrac{x_{AB}}{y_{AB}}\right)^2}}.$$

Врахувавши очевидний факт $|AB|^2 = x_{AB}^2 + y_{AB}^2$ отримуємо

$$Пр_{AB}\vec{V}_{ЛА} = \frac{\sqrt{V_{ЛА}^2|AB|^2 - \left(V_{0,x}y_{AB} - V_{0,y}x_{AB}\right)^2}}{|AB|}.$$

Отриманий результат дозволяє записати (3.6) у вигляді

$$|\vec{V}| = \frac{\sqrt{V_{ЛА}^2|AB|^2 - \left(V_{0,x}y_{AB} - V_{0,y}x_{AB}\right)^2}}{|AB|} + \frac{V_{0,x}y_{AB} + V_{0,y}x_{AB}}{|AB|}. \qquad (3.12)$$

Елемент матриці комівояжера дорівнює часу t_{AB}, яке потребує БПЛА для перельоту із точки А в точку В з врахуванням дії вітру $t_{AB} = \dfrac{|AB|}{|\vec{V}|}$.

Формула (3.10) дозволяє записати цей вираз у вигляді

$$t_{AB} = \frac{|AB|^2}{\sqrt{V_{ЛА}^2|AB|^2 - \left(V_{0,x}y_{AB} - V_{0,y}x_{AB}\right)^2} + V_{0,x}y_{AB} + V_{0,y}x_{AB}}. \qquad (3.13)$$

3.7 Кінцеве формулювання задачі

Таким чином, в роботі розглядається двохагентна задача комівояжера у наступній постановці:

1. У системі діє два агента (комівояжера, БПЛА, тощо), які мають різні швидкості (можуть мати і однакову швидкість), при цьому час руху між вузлами розраховується за формулою (3.13) і складаються матриці $\left\{a_{i,j}\right\}$, і $\left\{b_{i,j}\right\}$ де $a_{i,i} = b_{i,i} = \infty$. Вводяться бінарні змінні $x_{i,j}$ (для першого агента) і $y_{i,j}$ (для другого агента), які дорівнюють одиниці, якщо переміщення відповідного агента з пункту i у пункт j відбулося, і нулю, якщо не відбулося.

2. Кожен вузол системи (точка, місто і т.п.) мають бути відвідані першим або другим агентом лише один раз. І залишені кожним агентом також лише один раз. Для цього вводяться обмеження

$$\sum_{i=1}^{N} x_{i,j} + \sum_{i=1}^{N} y_{i,j} = 1, \quad j = 1,2,...,N,\ i \neq j.$$

$$\sum_{j=1}^{N} x_{i,j} + \sum_{j=1}^{N} y_{i,j} = 1, \quad i = 1,2,...,N,\ j \neq i.$$

Якщо у системі є вузол, спільний для обох агентів, тобто спільна точка старту і посадки БПЛА, або «депо» чи «гараж» для транспортних засобів, які виїжджають звідти робити розвозку товару за маршрутами і повертаються туди, то відповідні умови для цього вузла (який має номер, наприклад M) змінюються на

$$\sum_{i=1}^{N} x_{i,M} + \sum_{i=1}^{N} y_{i,M} = 2, \quad \sum_{j=1}^{N} x_{M,j} + \sum_{j=1}^{N} y_{M,j} = 2.$$

З додаванням умови

$$\sum_{i=1}^{N} x_{i,M} = 1$$

3. Щоб виключити з розв’язку ситуацію, коли у вузол входить перший агент, а виходить другий, який не заперечують подані вище обмеження,

додатково вводяться ще обмеження на рівність кількостей прибуття і виходу кожного агента з вузла

$$\sum_{i=1}^{N} x_{i,j} - \sum_{k=1}^{N} x_{j,k} = 0, \quad j = 1,2,...,N,\ i \neq j,$$

$$\sum_{i=1}^{N} y_{i,j} - \sum_{k=1}^{N} y_{j,k} = 0, \quad j = 1,2,...,N,\ i \neq j.$$

Якщо агент не прибуває у вузол i, то і не повинен відбувати з нього. А якщо прибув, то і відбуває стільки ж разів, скільки і прибував.

4. Цільова функція – це час виконання всієї операції, тобто найбільший з двох інтервалів часу виконання роботи кожним з агентів.

$$\max\left\{\left(\sum_{j=1}^{N}\sum_{i=1}^{N} a_{i,j}x_{i,j}\right),\left(\sum_{j=1}^{N}\sum_{i=1}^{N} b_{i,j}y_{i,j}\right)\right\} \to \min$$

Ця умова є нелінійною, але її можна подати як сукупність мінімізації часу виконання роботи одним з агентів за умови виконання своєї роботи другим агентом за ще менший час. Тобто маємо

$$F = \sum_{j=1}^{N}\sum_{i=1}^{N} a_{i,j}x_{i,j} \to \min$$

За умови

$$\sum_{j=1}^{N}\sum_{i=1}^{N} b_{i,j}y_{i,j} - \sum_{j=1}^{N}\sum_{i=1}^{N} a_{i,j}x_{i,j} \leq 0.$$

5. **Остання умова** – розв’язок має містити лише **один цикл для кожного з агентів**. Перелічені вище умови не гарантують цього і при розв’язанні потрібно корегувати отримані розв’язки та відкидати ті, які не задовольняють цій умові. Дана обставина значно ускладнює розв’язання задачі, навіть у порівнянні з класичною задачею комівояжера. Це викликано тим, що для певної системи вузлів кількість змінних у даній задачі буде в два рази більша, ніж у класичній задачі комівояжера.

4. ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМУ

Для програмної реалізації обраного алгоритму і розв'язання поставленої задачі обрано систему комп'ютерної алгебри Maple. Зумовлено це рядом причин, а саме – легкістю програмування математичних операцій та візуалізації результатів, легкістю перевірки результатів виконання операцій з матрицями (візуалізація матриць), а у першу чергу - наявністю у програмі Maple вбудованої операції `LPSolve`, (модуль `Optimization`) яка дозволяє розв'язувати задачі лінійного та дискретного програмування. Остання обставина є найважливішою, бо дискретне програмування є необхідним елементом розв'язання задачі комівояжера і для скорочення часу доцільно застосовувати вже готові програмні рішення, ніж писати власні алгоритми для розв'язання широко відомих задач.

Однак, нажаль, в ході виконання роботи було з'ясовано, що процедура `LPSolve` не завжди коректно працює при розв'язанні задач бінарного програмування. Часто вона навіть за простих умов і малої розмірності задачі видає сповіщення про похибку, чи зависає (розв'язання зациклюється). Але за відсутності умов бінарності змінних, лише за умов їхньої невід'ємності процедура працює без похибок. Тому на основі команди `LPSolve` було створено розв'язання задач бінарного програмування. Яка розв'язує задачу за наявності обмежень на бінарность лише для деяких змінних, корегуючи їх список до тих пір, поки всі змінні не стануть бінарними. Зазвичай більша частина змінних приймає значення 0 або 1 без автоматично, лише завдяки умовам задачі (3.2)-(3.4). Корегування потребує лише 5-10% змінних і за декілька ітерацій отримуємо розв'язок задачі бінарного програмування.

Слід зауважити, що у роботі для розв'язання задачі застосовується метод гілок та меж, який є ні чим іншим, ніж покращеним методом перебору. Тому розв'язання потребує виконання великої кількості ітерацій – розв'язання близьких за суттю задач, які відрізняються одна від іншої лише деякими

коефіцієнтами у матрицях $\{a_{i,j}\}$ і $\{b_{i,j}\}$, і відповідно у цільовій функції та одному обмеженні, до яких ці коефіцієнти входять. Тому для оптимізації оперативної пам'яті комп'ютера у програмі зберігаються не самі ці модифіковані матриці (які мають великі розміри), а так звані «задачі», тобто списки заборонених переміщень, які розривають таким чином підцикли, що увійшли у розв'язок на попередній ітерації. Для полегшення роботи вказані заборонені переміщення для $x_{i,j}$ та $y_{i,j}$ входять у список («задачу») у наскрізній нумерації (одноіндексової).

Наприклад «задача» яка задана списком [[22], [145]] вказує на те, що для змінної $x_{i,j}$ вводиться заборона на переміщення за номером 22, а для $y_{i,j}$ вводиться заборона на переміщення за номером 145, а відповідні елементи в матрицях $\{a_{i,j}\}$ та $\{b_{i,j}\}$ на черговій ітерації потрібно замінити завеликим числом, яке значно більше за інші елементи цих матриць. Таким чином відповідні змінні $x_{i,j}$ і $y_{i,j}$ не потраплять у розв'язок, а будуть нульовими.

Програма складається з декількох структурних елементів:

1) Введення швидкості і напряму вітру, швидкостей обох літальних апаратів (агентів), кількості і координат вузлів.

2) Розрахунок матриць часу перельоту між вузлами для кожного з агентів (відповідно матриці $\{a_{i,j}\}$ і $\{b_{i,j}\}$ часу перельоту між вузлами i та j), які розраховується за формулою (3.13).

3) Формування ключових процедур, таких як:

а) **num_Xij, num_Yij** , які повертають номер n змінної $x_{i,j}$ і $y_{i,j}$ за заданими індексами i та j в наскрізній нумерації змінних, враховуючи заборону на змінні з однаковими індексами i,i;

б) обернені процедури, які повертають індекси i та j змінних $x_{i,j}$ і $y_{i,j}$ за заданим n (**Xij_num**, **Yij_num**);

в) процедура розв’язання задачі бінарного програмування (**BINARYSOLVE**);

г) процедура пошуку підциклів за заданим розв’язком задачі бінарного програмування (**SUBCYCLES**);

д) процедура пошуку найменшого підциклу у списку підциклів (**MINSUBCYCLE**);

е) процедура, яка по заданими матрицями часу перельоту агентів між вузлами ($\{a_{i,j}\}$ і $\{b_{i,j}\}$) та заданим підциклам - спискам заборонених перельотів, повертає матриці $\{a_{i,j}\}$ та $\{b_{i,j}\}$ зі зміненими відстанями між вузлами відповідно до списків (**TRANSFORM_TATB**), змінюючи вказані відстані завеликим числом, щоб при дані перельоти не потрапили у розв’язок;

ж) Три процедури доповнення списку «задач» новими задачами, які формуються на основі вилучених з чергового розв’язку підциклів, які потрібно розірвати. Якщо черговий розв’язок містить підцикли для змінних $x_{i,j}$ і $y_{i,j}$, то використовується процедура **PLTRANSFORME_2**, якщо ж за однією зі змінних розв’язок вже є одним циклом і не містить підциклів, то список нових «задач» формується на основі чергової «задачі» та підциклу за однією зі змінних за допомогою процедур **PLTRANSFORME_1X, PLTRANSFORME_1Y.**

з) Процедура розв’язання задачі дискретного програмування за заданими матрицями відстаней для кожного з агентів, в якої автоматично формуються вектор цільової функції, обмеження і т.п., і видається розв’язок у вигляді значення цільової функції і набору підциклів для кожного агенту (**SOLVPROBLEM**).

4) Розв’язання першої задачі, формування перших елементів списку «задач»

5) Власно процедура пошуку розв’язку задачі комівояжера. Являє собою цикл, який виконується поки список задач не стане порожнім.

6) Блок візуалізації результатів, побудови графіку циклів.

Ключові елементи програми наведено нижче:

1) Процедура розв'язання задачі бінарного

```
> BINARYSOLVE:=proc(V, ANE, BNE, AEQ,BEQ)
> local flag, listnonbin, i, SLV, MSL, TF,num;
> listnonbin:=[]; flag:=0:
>
> SLV:=LPSolve(V,[ANE, BNE, AEQ,BEQ],assume=nonnegative):   решение
задачи линейного программирования
>    MSL:=SLV[2]:
>    TF:=SLV[2]:
>
>    for i from 1 to 2*N*(N-1) do:
>       if abs(MSL[i])<0.01 then MSL[i]:=0: end if:   округление
переменных
>       if abs(MSL[i])>0.99 then MSL[i]:=1: end if:
>    end do:
>
>        for i from 1 to 2*N*(N-1) do:
>              if abs(MSL[i])<0.99 then
>                    if abs(MSL[i])>0.01 then
>                        listnonbin:=[op(listnonbin), i]:
составление списка небинарных переменных
>                        flag:=flag+1:
>          end if:
>      end if:
> end do:
>
>         while flag>0 do:          цикл. решается до тех пор пока решение не
будет целочисленным
>                          num:=0:
>                          SLV:=LPSolve(V,[ANE, BNE,
AEQ,BEQ],assume=nonnegative, binaryvariables=listnonbin):
>                    MSL:=SLV[2]: TF:=SLV[1]:
>                                  for i from 1 to 2*N*(N-1) do:
>                                        if abs(MSL[i])<0.99 then
>                                           if abs(MSL[i])>0.01
then
>
listnonbin:=[op(listnonbin), i]:  пополнение списка небинарных переменных
>                                                num:=num+1:
>                                             end if:
>                                        end if:
>                                  end do:
>                          flag:=num:
>        end do:
> RETURN(TF,MSL):  возвращение значения целевой функции и вектора решения
> end proc:
```

2) Процедура пошуку підциклів за заданим розв'язком задачі бінарного програмування

```
> SUBCYCLES:=proc(NE)
> local SC, SSCA, startpoint, finpoint, c_ind:
> local i,NEL:
> SSCA:={}: NEL:=NE:
>
> while nops(NEL)>0 do:      пока основной список отрезков не пустой
>    SC:={NEL[1]}:
>    startpoint:=NEL[1][1]: finpoint:=NEL[1][2]:     стартовая и конечная
точка текущего подцикла
>    NEL:=NEL minus {NEL[1]}:                        исключение
выбранного отрезка из основного списка отрезков
>    c_ind:=0:                                       флаг, индикатор того
что подцикл разомкнут. Если принимает значение 1 - цикл замкнулся
>       while c_ind=0 do:                            формирование
подцикла, пока подцикл не замкнут
>          for i from 1 to nops(NEL) do              перебор по всему
текущему основному списку отрезков
>             if NEL[i][1]=finpoint  then
>                                finpoint:=NEL[i][2]:
>                                SC:=SC union {NEL[i]}:
>                                NEL:=NEL minus {NEL[i]}: break:
принудительный выход из цикла перебора. Так как уменьшилась величина списка
>             end if:
>          end do:
>    if finpoint=startpoint then c_ind:=1:end if:  если совпание
текущей конечной точки с начальной - замыкание подцикла
>  end do:
>  SSCA:=SSCA union {SC}:                           текущий подцикл
заносится в список подциклов
> end do:
> RETURN(SSCA):
> end proc:
>
```

3) Процедура пошуку найменшого підциклу у списку підциклів

```
> MINSUBCYCLE:=proc(SSCA)
> local i, SPL, MSC, MEL;
>
> SPL:=[]:
>
>    for i from 1 to nops(SSCA) do:
>       SPL:=[op(SPL),nops(SSCA[i])]:
>    end do:
> MEL:=min(SPL):
```

```
> for i from 1 to nops(SPL) do:
>   if SPL[i]=MEL then:
>                   MSC:=SSCA[i]:
>                   break:
>   end if:
>  end do:
>
> RETURN(MSC):
> end proc:
```

4) Процедура зміни матриць коефіцієнтів за списком елементів підциклів, які потрібно розірвати:

```
> TRANSFORM_TATB:=proc(TA,TB,SZ0)
> local TA0, TB0, i, NUM:
>   TA0:=Matrix(N,N,TA):  TB0:=Matrix(N,N,TB):
>   for i from 1 to nops(SZ0[1]) do
>        NUM:=Xij_num(SZ0[1][i]):
>        TA0[NUM[1],NUM[2]]:=RZ:
>   end do:
>
>   for i from 1 to nops(SZ0[2]) do
>        NUM:=Yij_num(SZ0[2][i]):
>        TB0[NUM[1],NUM[2]]:=RZ:
>   end do:
>
> RETURN([TA0,TB0]);
> end proc:
```

5) Процедура розв'язання задачі дискретного програмування за заданими матрицями відстаней для кожного з агентів, в якої автоматично формуються вектор цільової функції, обмеження і т.п., і видається розв'язок у вигляді значення цільової функції і набору підциклів для кожного агенту.

```
> SOLVPROBLEM:=proc(TA0,TB0)
> local V0, ANE0, BNE0, SLV0, MX0, MY0, TM0, i,j, MSL0, MXSL0,
MYSL0, NXE0,NYE0, SSCAX0, SSCAY0, num;
> V0:=Vector(2*N*(N-1)):    вектор целевой функции
> num:=1:
>
> for i from 1 to N do:
>  for j from 1 to N do:
>    if i<>j then V0[num]:=TA0[i,j]: num:=num+1: end if:
>  end do:
> end do:
>
```

```
> ANE0:=Matrix(1,2*N*(N-1));
```
матрица ограничений в виде неравенства (матрица-строка)

```
> BNE0:=Vector(1): BNE0[1]:=0:
>
> num:=1:
```
формирование матрицы-строки ограничения

```
> for i from 1 to N do:
>  for j from 1 to N do:
>    if i<>j then ANE0[1,num]:=-TA0[i,j]: num:=num+1: end if:
>  end do:
> end do:
>
> for i from 1 to N do:
>  for j from 1 to N do:
>    if i<>j then ANE0[1,num]:=TB0[i,j]: num:=num+1: end if:
>  end do:
> end do:
>
>    SLV0:=BINARYSOLVE(V0,ANE0, BNE0, AEQ,BEQ):
```
решение задачи лин.прогр. матрицы ограничений AEQ and BEQ не трансформируются!

```
>
>    MSL0:=SLV0[2]:
```
матрица решения

```
>    TM0:=SLV0[1]:
```
время полета (значение целевой функции

```
>
>       for i from 1 to 2*N*(N-1) do:
```
постообработка решения (округление до 0 и 1)

```
>          if abs(MSL0[i])<0.01 then MSL0[i]:=0: end if:
>          if abs(MSL0[i])>0.99 then MSL0[i]:=1: end if:
>       end do:
>
>    MXSL0:=Vector(N*(N-1)):   MYSL0:=Vector(N*(N-1)):
```
решения для X and Y

```
>
>     for i from 1 to N*(N-1) do:
>         MXSL0[i]:=MSL0[i]:   MYSL0[i]:=MSL0[i+N*(N-1)]:
>      end do:
>
>      NXE0:={}: NYE0:={}:
```
составление списков

```
>
> for i from 1 to N*(N-1) do:
>    if MXSL0[i]=1 then NXE0:=NXE0 union {Xij_num(i)}: end if:
>    if MYSL0[i]=1 then NYE0:=NYE0 union {Yij_num(i+N*(N-1))}:
     end if:
> end do:
>
> SSCAX0:=SUBCYCLES(NXE0);   SSCAY0:=SUBCYCLES(NYE0):
> RETURN([TM0, SSCAX0,SSCAY0, MSL0]):
> end proc:
```

6) Процедура трансформації списку задач за двома підциклами

```
> PLTRANSFORME_2:=proc(SZ,SZ0,CXL,CYL)
> local SZF,i,j,  SZX0,SZY0, SZXT,SZYT, SZT;
>
> SZF:=SZ minus {SZ0}:  из списка задач исключается выбранная ранее задача
> SZX0:=SZ0[1]:SZY0:=SZ0[2]:    списки запрещенных перемещений в дополняемой задаче
>
> for i from 1 to nops(CXL) do:
>    for j from 1 to nops(CYL) do:
>          SZXT:=[op(SZX0),num_Xij(CXL[i][1],CXL[i][2])];
>          SZYT:=[op(SZY0),num_Yij(CYL[j][1],CYL[j][2])];
>          SZT:=[SZXT,SZYT];    текущая задача
>          SZF:=SZF union {SZT};
>    end do:
> end do:
>
> RETURN(SZF);
>
> end proc:
```

7) Головна процедура розв’язання задачі

```
> while nops(SZ)>0 do:
>
>     SZ0:=SZ[1]:                 выбрали задачу для решения из списка задач
>     SZ:=SZ minus {SZ0}:         удалили выбранну задачу из основного списка задач
>
>
>     TM0:=TRANSFORM_TATB(TA,TB,SZ0):      преобразованные матрицы расстояний согласно списку запретов
>     TA0:=TM0[1]:
>     TB0:=TM0[2]:
>
>     SLV0:=SOLVPROBLEM(TA0,TB0):          решение задачи дискретного программирования с использованием трансформированных матриц расстояний
>
>     if SLV0[1]<LMIN then:                если значение целевой функции меньше текущего минимума, тогда....
>          if (nops(SLV0[2])+nops(SLV0[3]))=2 then LMIN:=SLV0[1]:
SBEST:=SLV0: end if:  если в решении всего два цикла - запомнить его.
>
>          if nops(SLV0[2])>1 then
>                    if nops(SLV0[3])>1 then
```

```
>
CXL0:=MINSUBCYCLE(SLV0[2]):        если циклов больше одного в каждом решении, то...
>
CYL0:=MINSUBCYCLE(SLV0[3]):        выбираем наименьшие подциклы, и...
>
SZ:=PLTRANSFORME_2(SZ,SZ0,CXL0,CYL0): трансформируем список задач по выбранной задаче и подциклам.
>                          end if:
>          end if:
>
>          if nops(SLV0[3])=1 then
>                          if nops(SLV0[2])>1 then
>
CXL0:=MINSUBCYCLE(SLV0[2]):        если циклов больше одного в ОДНОМ решении, то...
>
SZ:=PLTRANSFORME_1X(SZ,SZ0,CXL0): трансформируем список задач по выбранной задаче и подциклу
>                          end if:
>          end if:
>
>            if nops(SLV0[2])=1 then
>                             if nops(SLV0[3])>1 then
>
CYL0:=MINSUBCYCLE(SLV0[3]):        если циклов больше одного в ОДНОМ решении, то...
>
SZ:=PLTRANSFORME_1Y(SZ,SZ0,CYL0): трансформируем список задач по выбранной задаче и подциклу
>                             end if:
>            end if
>    end if:
>
>   #print(nops(SZ)): print(LMIN):
>
> end do:
```

5. МОДЕЛЬНИІ ПРИКЛАДИ СКЛАДАННЯ МАРШРУТІВ ПОЛЬОТУ

5.1 Простий приклад

Для наглядної демонстрації роботи алгоритму розглянемо найпростіший приклад складання маршруту обльоту восьми точок, координати яких (в км) наведено в таблиці 5.1. Точки підібрано таким чином, що оптимальний розв'язок задачі є очевидним і може буди отриманим лише міркуванням та аналізом умов.

Таблиця 5.1.

n	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
x	0	0	3	3	10	10	13	13
y	0	2	0	1	0	1,7	0	1

На рис. 5.1 показане положення точок.

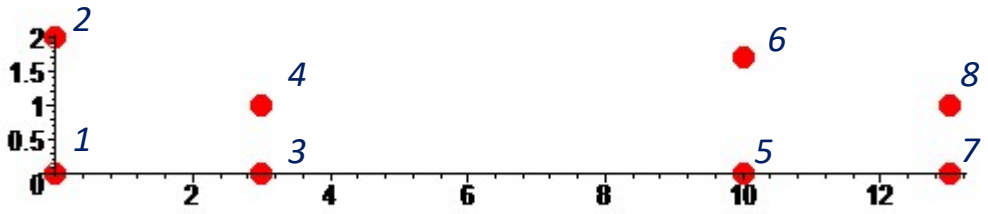

Рисунок 5.1 - Положення точок

Відмітимо, що система містить дві групи точок по 4 точки в кожній. Ці групи віддалені одна від одної на відносно значну відстань. При цьому групи загалом подібні, однак положення аналогічних точок 2 та 6 незначним чином відрізняються. А саме – координата *y* для точки 2 дорівнює 2 км, а для точки 6 дорівнює 1,7 км. Тобто цикл 5-6-8-7 буде коротшим, ніж аналогічний цикл 1-2-4-3.

Призначимо швидкості агентів (ПБЛА) також різними, а саме швидкість першого БПЛА призначимо 61 км/год, а другого БПЛА призначимо 70 км/год. Швидкість вітру 5 км/год, а його напрям $\varphi = \frac{\pi}{4}$.

Розв'язання задачі комбінаторного програмування без врахування умови про циклічність розв'язку призводить до наступних маршрутів: $x_{5,6} = x_{6,5} = 1$, $x_{7,8} = x_{8,7} = 1$, $y_{1,2} = y_{2,1} = 1$, $y_{3,4} = y_{4,3} = 1$. Всі інші змінні дорівнюють нулю. Ці маршрути показано на рис. 5.2.

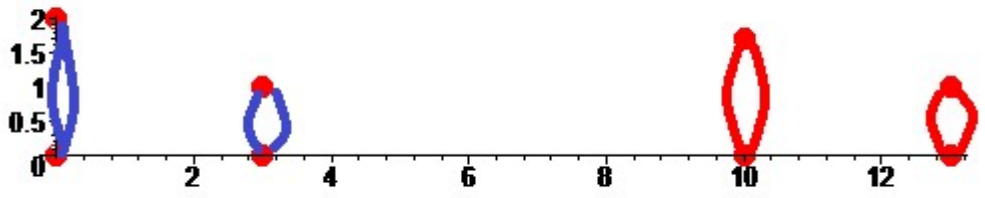

Рисунок 5.2 - Маршрути першого розв'язку

Але вочевидь цей розв'язок не задовольняє умові про наявність лише одного циклу для кожного з агентів. Тому за наведеним алгоритмом обираються по одному підциклу кожного з агентів і створюються декілька нових задач, які заносяться до списку задач.

Так, обираються підцикли $x_{5,6} = x_{6,5} = 1$ і $y_{1,2} = y_{2,1} = 1$ і розриваються. У наскрізній одноіндексовій нумерації відповідні чотири задачі мають вигляд

$$[[33],[57]],[[33],[64]],[[40],[57]],[[40],[64]]\,.$$

Виконавши декілька ітерацій, отримуємо кінцевий розв'язок. Маршрути показано на рис. 5.3.

Напрям вітру вказано на рисунку пунктирними стрілками.

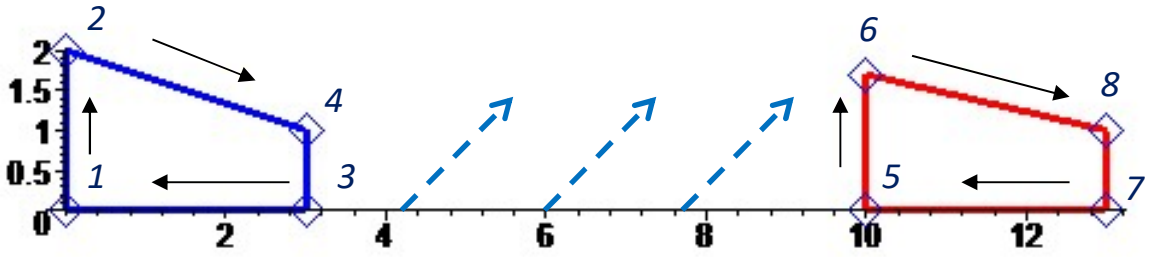

Рисунок 5.3 – Оптимальні маршрути

Розв’язок має вигляд: $x_{1,2} = x_{2,4} = x_{4,3} = x_{3,1} = 1$ і $y_{5,6} = y_{6,8} = y_{8,7} = y_{7,5} = 1$.

Можна відзначити два факти, які є у розв’язку

1) Менший цикл, якій розташовано лівіше, обрав менш швидкий перший агент, тоді які більший за протяжністю цикл обрано більш швидким другим агентом.

2) Напрям обходу вузлів агентами узгоджується з напрямом вітру. А саме, найдовші вертикальні переходи 1-2 та 5-6 направлені за напрямом вітру, і у найдовших переміщеннях 2-4 та 6-8 вітер також є попутним.

5.2 Приклад розв’язання більш складної задачі

Розглянемо задачу маршрутизації для дев’яти, розташованих так, як показано на рис. 5.4. Кожній точки для визначеності призначено свій номер з 1 до 9. Координати точок маршруту приведено в таблиці 5.2.

Таблиця 5.2

Координати точок

n	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
x	7,91	0,44	8,55	8,25	7,72	15,46	2,13	18,9	15,0
y	3,86	16,0	16,85	19,92	13,89	14,61	7,93	4,22	9,1

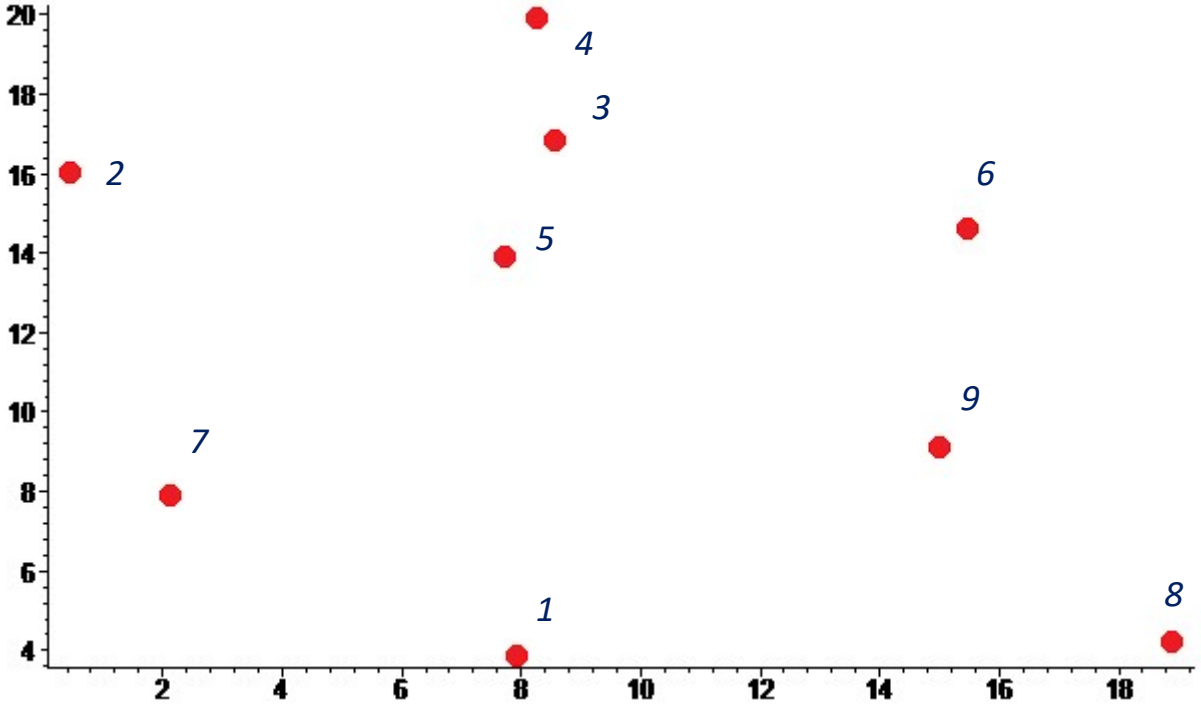

Рисунок 5.4 - Точки маршруту

При розрахунках швидкість першого БПЛА приймалася рівною 70 км/год, а швидкість другого БПЛА – 61 км/год, швидкість вітру складала 5 км/год. Напрям $\varphi = \frac{\pi}{4}$. При постановці вважалось що спільної точки немає.

Результат першої ітерації показано на рис. 5.5.

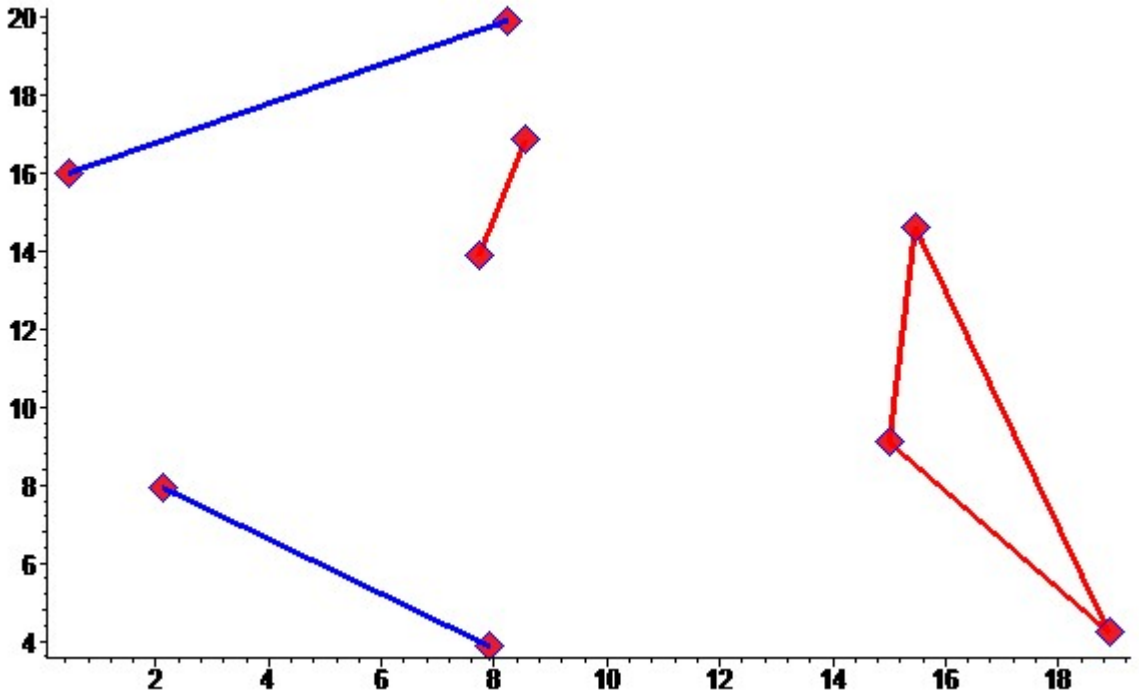

Рисунок 5.5 - Перша ітерація розрахунку маршрутів

Як бачимо, у кожного з агентів оптимальній маршрут складається з двох циклів. Причому у першого агента до одного з циклів входять три вузла.

Оптимальний маршрут показано за рис. 5.6.

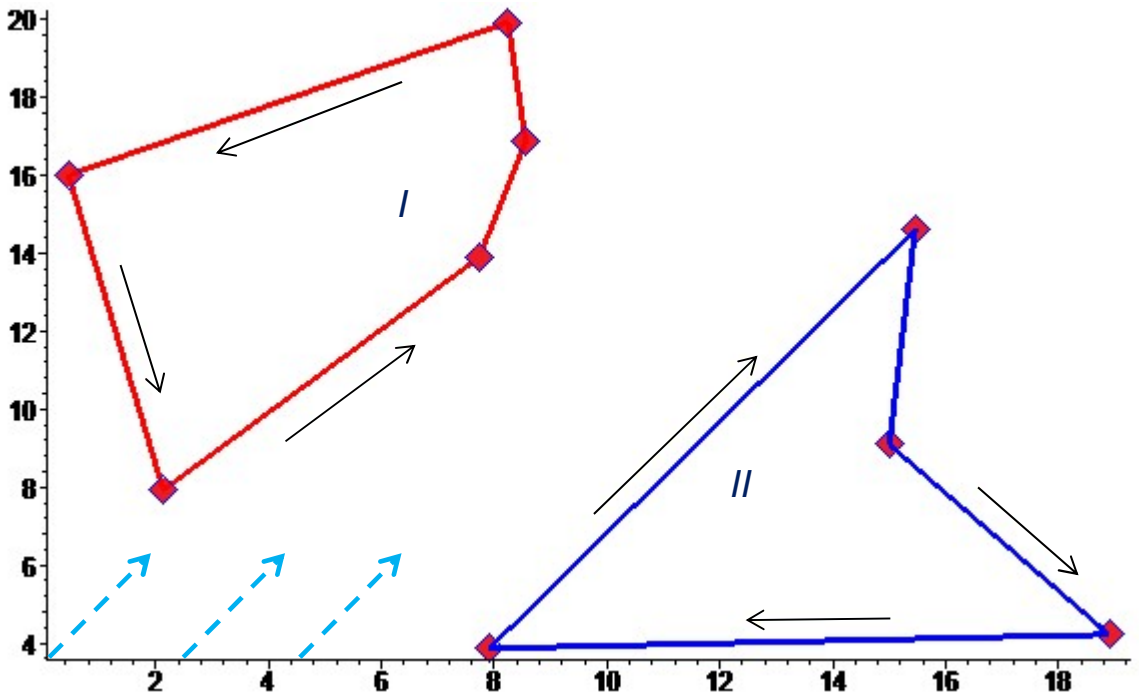

Рисунок 5.6 - Оптимальні маршрути

Як бачимо, оптимальні маршрути для кожного агента мають вигляд непересічних циклів. Напрям вітру показано на рисунку штриховою стрілкою.

5.3 Задача з наявністю однієї спільної точки (гараж чи депо)

Будемо розглядати ту ж саму геометрію точок (рис. 5.4), але будемо вважати що одна з точок є спільною для обох агентів. Оберемо в якості спільної точки точку за номером 5, задав їй координати (8; 11) км.

На першій ітерації отримуємо наступний розв’язок, який показано на рис. 5.7

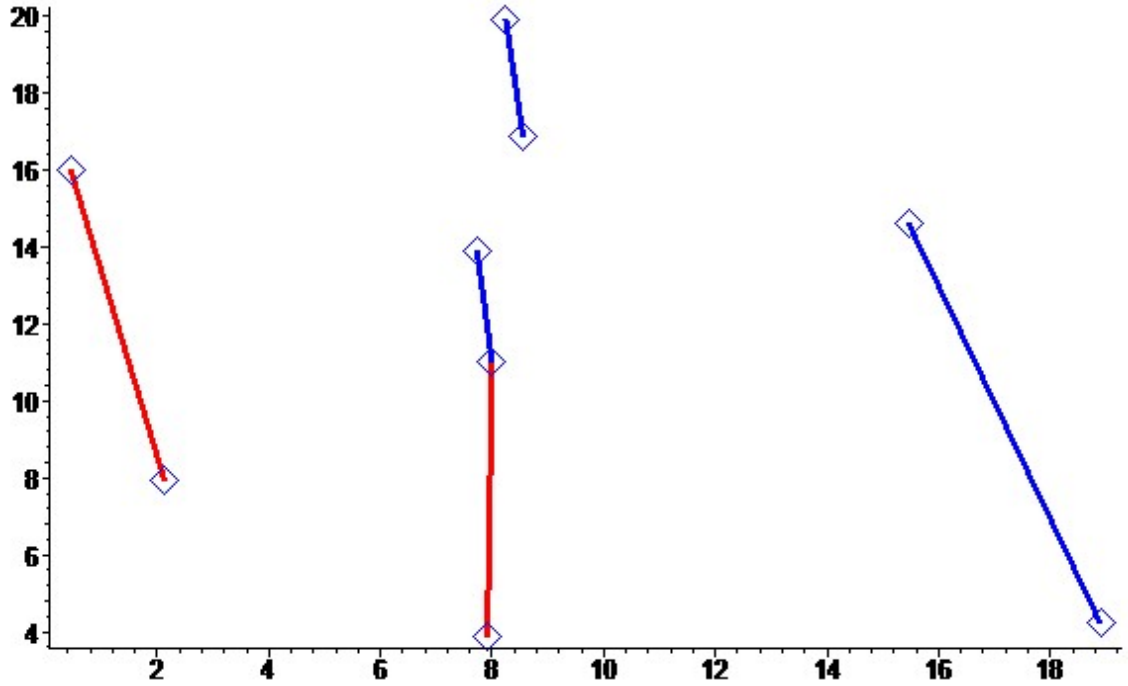

Рисунок 5.7 - Перша ітерація розрахунку маршрутів зі спільною точкою

Після виконання алгоритму отримуємо розв’язок, який містить лише по одному циклу для кожного з агентів, див. рис. 5.8.

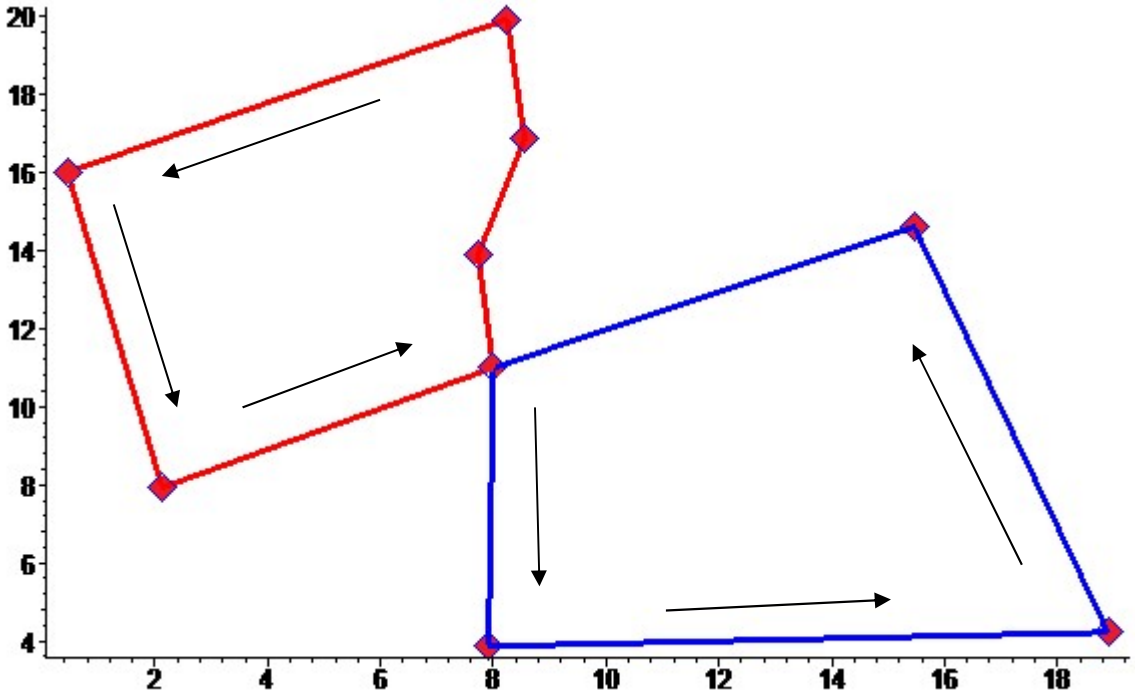

Рисунок 5.8 - Фінальний вигляд маршрутів

6. ЕКОНОМІЧНА ЧАСТИНА

6.1 Опис програмного продукту

Ця глава присвячена кількісній оцінці матеріальних витрат для створення представленої в дипломній роботі аналітичної, наукової роботі і програмного продукту, який будує розв’язок двохагентної задачі комівояжера, тобто формує оптимальний маршрут групи вузлів за заданої швидкості та напрямку вітру, власної швидкості двох БПЛА, який мінімізує час виконання всієї роботи.

Під програмним продуктом (ПП) розуміється програмне забезпечення (ПЗ) як результат людської діяльності, виставлений на ринку масового покупця в якості товару і має ненульову споживчу вартість. Продукт розроблений в програмному середовищі Maple 13.

6.2 Розрахунок заробітної платні та трудомісткість робіт

Метою даного розділу дипломної роботи є розрахунок собівартості і вартості програмного продукту. Для оцінки розроблюваного програмного продукту необхідно виконати наступні пункти:

- скласти перелік робіт, необхідних для виконання, розрахувати трудовитрати на їх виконання;
- розрахувати заробітну плату розробників;
- розрахувати витрати на комплектуючі, матеріали і машинний час.

До витрат на розробку програмного продукту також входять:

- вартість оренди комп'ютера;
- відрахування з заробітної плати;
- вартість малоцінних і швидкозношуваних предметів і т.д.

У список робіт, які необхідно виконати входять:

- формулювання і постановка задачі;
- проектування програмного продукту;
- розробка програмного продукту;

- тестування і впровадження програмного продукту і супровід.

6.3 Виконавці роботи

Для керівництва ходом робіт і ведення всього проекту в цілому необхідна посада керівника. Для проектування підсистеми, її подальшої налагодження і введення в експлуатацію необхідна участь програміста. Для тестування програми потрібен тестувальник. Дані про посадові окладах і склад виконавців робіт занесені в таблицю 6.1. Тривалість робочого місяця будемо в середньому вважати 22 дні. По-перше розглянемо скільки буде коштувати розробка ПП з невеликою командою, яка проте має розподілені ролі.

Таблиця 6.1 – Склад виконавців робіт

Виконавці	Посадові оклади, грн	
	Місячні	Щоденні
Керівник	11000	500
Програміст	8800	400
Тестувальник	6600	300

6.4 Перелік робіт для створення програмного продукту

Розрахуємо тривалість розробки продукту за видами робіт. Результати розрахунків занесені в таблицю 6.2.

Таблиця 6.2 – Перелік робіт

з/п	Найменування стадій і етапів	Тривалість, дні	Трудомісткість, дні		
			Керівник	Програміст	Тестувальник
Розробка технічного завдання					
1.1	Організаційна підготовка до створення ПП	1	1	-	-
1.2	Розробка ТЗ на постанову	2	2	-	-

	задачі				
	Разом	3	3	0	0
Постановка завдання					
2.1	Розробка алгоритмів	8	3	5	-
2.2	Технічне забезпечення	2	-	2	-
2.3	Розробка схем програм	3	-	3	-
2.4	Розробка опису ТЗ	2	-	2	-
	Разом	15	3	12	0
Розробка програмного продукту					
3.1	Аналіз даних в ПП	3	-	3	-
3.2	Розробка ПП і реалізація її в середовищі Maple	12	-	12	-
3.3	Тестування ПП	2	-	-	2
3.4	Відладка ПП	4	-	2	2
3.5	Розробка документації	5	1	3	1
3.6	Розробка технологічної документації	2	2	-	-
3.7	Випуск комплект. раб. документ.	2	-	1	-
	Разом	29	3	21	5
	Усього	47	9	33	5

Розрахуємо основну заробітну плату (ОЗП) виконавцям :

$$ОЗП = 9 * 500 + 33 * 400 + 5 * 300 = 19200 \text{ грн} \quad (6.1)$$

Розрахуємо додаткову заробітну плату (ДЗП), яка визначена в розмірі 10% ОЗП та єдиний соціальний внесок ЄСВ.

$$ДЗП = 0,1 * 19200 = 1920 \text{ грн} \quad (6.2)$$

$$ЄСВ = 0,22 * (ОЗП + ДЗП) = 0,22 * (19200 + 1920) = 4646 \text{ грн} \quad (6.3)$$

6.5 Розрахунок витрат на матеріали і комплектуючі

Розрахуємо витрати на матеріали і комплектуючі, необхідні для написання програми. Результати занесені в таблицю 6.3.

Амортизацію не розраховуємо, тому що орендуємо приміщення з усім необхідним обладнанням.

Таблиця 6.3 – Вартість матеріалів і комплектуючих та поточні витрати

Матеріали	Кількість	Ціна, грн	Сума, грн	Призначення
Папір	400 листів	0.5	200	Документація, друк
Оренда офісу та обладнання	2 тижні	2500	10000	Розробка ПП
Електроенергія	220 Вт/год	1.92	1.92*47*8*3=1128	Живлення комп'ютера
Разом, грн			11328	

6.6 Розрахунок витрат на заробітну плату

Схема витрат на створення і впровадження з угрупованням витрат за статтями калькуляції наведена в таблиці 6.4.

Таблиця 6.4 - Витрати на розробку програмного продукту

	Види витрат	Позначення, формула для розрахунку	Сума, грн
1.	Основна заробітна плата	ОЗП	19200
2.	Додаткова заробітна плата	ДЗП	1920
3.	Єдиний соц. внесок 22%	ЄСВ	4646
4.	Матеріали і комплектуючі		11328
5.	Собівартість ПП		37094

Отже повна ціна ПП, що розробляється складе 37094 грн.

6.7 Альтернативний процес розробки програмного продукту

В цілому для керівництва ходом робіт необхідна розподілена команда. Але якщо як у нашому випадку, проект невеликий і не дуже складний, одна і та ж людина може виконувати декілька ролей. Тому для керівництва ходом робіт і ведення всього проекту в цілому, тестування і розробки складемо команду з двох висококваліфікованих програмістів. Дані про посадові окладах і склад виконавців робіт занесені в таблицю 6.5. Тривалість робочого місяця будемо в середньому вважати 22 дні. По-перше розглянемо скільки буде коштувати розробка ПП з невеликою командою, яка проте має розподілені ролі.

Таблиця 6.5– Склад виконавців робіт

Виконавці	Посадові оклади, грн	
	Місячні	Щоденні
Програміст 1	11000	500
Програміст 2	11000	500

6.8 Перелік робіт для створення ПП (однорідна команда)

Розрахуємо тривалість розробки продукту за видами робіт. Результати розрахунків занесені в таблицю 6.6.

Таблиця 6.6 – Перелік робіт

з/п	Найменування стадій і етапів	Тривалість, дні	Трудомісткість, дні	
			Програміст 1	Програміст 2
Розробка технічного завдання				
1.1	Розробка ТЗ на постанову задачі	2	1	1
	Разом	2	1	1
Постановка завдання				

2.1	Розробка алгоритмів	7	3.5	3.5
2.2	Розробка схем програм	1	0.5	0.5
2.3	Розробка опису ТЗ	2	1	1
	Разом	10	5	5
Розробка програмного продукту				
3.1	Аналіз даних в ПП	2	1	1
3.2	Розробка ПП і реалізація її в середовищі Maple	14	7	7
3.3	Відладка ПП	5	2.5	2.5
3.4	Розробка технологічної супровідної документації	2	1	1
3.5	Випуск комплект. раб. документ.	2	1	1
	Разом	25	12.5	12.5
	Усього	36	18	18

Розрахуємо основну заробітну плату (ОЗП) виконавцям :

$$ОЗП = 18 * 2 * 500 = 18000 \text{ грн} \quad (6.4)$$

Розрахуємо додаткову заробітну плату (ДЗП), яка визначена в розмірі 10% ОЗП та єдиний соціальний внесок ЄСВ.

$$ДЗП = 0{,}1 * 18000 = 1800 \text{ грн} \quad (6.5)$$

$$ЕСВ = 0{,}22 * (ОЗП + ДЗП) = 0{,}22 * (18000 + 1800) = 4356 \text{ грн} \quad (6.6)$$

6.9 Розрахунок витрат на матеріали і комплектуючі (однорідна команда)

Розрахуємо витрати на матеріали і комплектуючі, необхідні для написання програми. Результати занесені в таблицю 6.7.

Таблиця 6.7 – Вартість матеріалів і комплектуючих та поточні витрати

Матеріали	Кількість	Ціна, грн	Сума, грн	Призначення
Папір	400 листів	0.5	200	Документація, друк
Оренда офісу та обладнання	2 тижні	3500	7000	Розробка ПП
Електроенергія	220 Вт/год	1.92	1.92*36*8*2=1106	Живлення комп'ютера
Разом			**8306**	

Таблиця 6.8 – Вартість основних засобів

п/п	Найменування	Кількість, шт	Ціна за одиницю, грн	Сума, грн
1	Ноутбук ASUS N75SF	2	21000	42000
Разом				42000

Амортизаційні відрахування (АМО) - частина вартості основних засобів, що входять у вартість готової продукції.

Річна норма амортизаційних відрахувань (АМО) розраховується як 25% від вартості одного комп'ютера і його комплектуючих.

$$АМО = 23700{,}00 * 0{,}25 = 59 \quad (6.7)$$

Норма амортизаційних відрахувань на період роботи проекту становить 15 робочий день, розраховується за такою формулою:

$$АМО_{\text{на раб.период.}} = \frac{АМО}{264} * 20 \qquad АМО_{\text{на раб.период.}} = \frac{АМ}{26} \quad (6.8)$$

$$АМО_{\text{на раб.период.}} = \frac{10500}{264} * 20 = 795\text{грн}$$

$$АМО_{\text{на раб.период.}} = \frac{5925}{264} * 15 = 33($$ (6.9)

де 264 – кількість робочих днів на рік.

6.10 Розрахунок витрат на заробітну плату (однорідна команда)

Схема витрат на створення і впровадження з угрупованням витрат за статтями калькуляції наведена в таблиці 6.9.

Таблиця 6.9 - Витрати на розробку програмного продукту

Види витрат	Позначення, формула для розрахунку	Сума, грн
1. Основна заробітна плата	ОЗП	18000
2 Додаткова заробітна плата	ДЗП	1800
3. Єдиний соц. внесок 22%	$ЄСВ = 0{,}22 * (ОЗП + ДЗП)$	4356
4. Матеріали і комплектуючі		8306
5. Амортизація	АМО	795
6. Собівартість ПП		33257

Отже повна ціна ПП, що розробляється складе 33257 грн.

Висновок:

На сьогоднішній день в умовах високої конкуренції на ІТ-ринку дуже важливо розробити програмний продукт за прийнятною для клієнта ціною. Для цього, до моменту завершення розробки програмного продукту, необхідно знати собівартість і ціну готової програми, а також мати інформацію про потенційних покупців і кількості одиниць програмного продукту, який буде проданий їм, тобто досліджувати ринок збуту. Завчасна обізнаність про потенційних покупців допоможе регулювати ціну в умовах ринкової конкуренції та при цьому залишати її оптимальною. Також, якщо на виході ми хочемо отримати програмний продукт високої якості, слід наймати досвідчених, а отже високооплачуваних фахівців. Слід знати приблизну

вартість готового продукту, щоб не найняти зайвих фахівців і не перевищити заплановану вартість програми, також необхідно скласти перелік і графік робіт, для того щоб укластися в терміни і не збільшити вартість продукту, шляхом збільшення терміну його розробки.

Внаслідок зазначених вище причин в даному розділі було проведено порівняльний аналіз ефективності роботи розподільної та однорідної команди :

• розрахунок собівартості розробленого програмного продукту розподільної командою склав 37094 грн, а однорідною 33257 грн;

• розрахунок щоденної заробітної плати, для фахівців, що беруть участь в розробці ПП: керівник - 600 грн, програміст - 500 грн, тестувальник - 400 грн;

• розрахунок суми витрат на комплектуючі матеріали для розподільної команди 11328 грн, а для однорідної 8306 грн;

• перелік робіт на створення продукту.

В вище наведених розрахунках не бралася до уваги амортизація, це обумовлено тим, що все обладнання та приміщення орендувалися. Скоротивши для однорідної команди час виконання деяких робіт пов’язаних з документацією та обговоренням завдання строк розробки зменшився з 47 до 36 днів, порівнюючи з розподільною командою. Враховуючи, що розробка ПП однорідною командою виявилася дешевша і швидша, приходимо до висновку, що на невеликих проектах краще використовувати однорідну невелику команду, так як вона економніша та швидша.

ВИСНОВКИ

У роботі проведено аналіз проблеми побудови маршруту двохагентної задачі комівояжера. При постановці задача була розглянута як система, визначені її елементи, запропоноване декілька відповідних математичних моделей, обрана найпростіша з них, розроблено алгоритм її розв’язання, створена програма з розв’язання двохагентної задачі комівояжера та проведено аналіз розв’язків декількох задач.

У першому розділі проведено аналіз наукових джерел зі вказаної проблематики, визначені цілі та задачі дослідження, визначена актуальність задачі за визначено метод розв’язання.

У другому розділі проблема оптимальної маршрутизації розглянута з точки зору системного аналізу, визначені складові системи.

У третьому розділі наведено опис математичної моделі задачі маршрутизації. Дано опис задачі комівояжера та наведені різні варіанти постановки двохагентної задачі комівояжера. Обрано одну з моделей яка здається найбільш адекватною поставленій задачі, наведено шукані змінні, цільову функцію та декілька обмежень на відвідування вузлів. Надано аналітичні залежності зі знаходження елементів матриць комівояжера для кожного з агентів, які враховують швидкість и напрямок вітру та власну швидкість БПЛА. Описано постановку задачі зі спільною для обох агентів «базою» та наведено відповідні корективи, які потрібно внести до системи обмежень на змінні.

У четвертому розділі дано опис створеної програми та наведено її ключові елементи – підпрограми та функції.

У п’ятому розділі наведено розв’язок трьох модельних задач – простої, більш складної, та задачі зі спільною базою. Проведено дослідження впливу на топології оптимального маршруту та мінімальний час польоту вхідних параметрів задачі – швидкості вітру та напрямку вітру.

У шостому розділі проведено розрахунок собівартості проведеної роботи та створеного ПО.

Подальші перспективи:

1. Створення інтерфейсу ПО.
2. Побудова розв’язків більш складних задач з маршрутизації: включення в модель зміни швидкості вітру у часі, різних вузлів старту та фінішу, обліт максимальної кількості вузлів при обмеженій довжині маршруту, обліт максимальної сумарної кількості вузлів за наявності обмеження на час виконання роботи, тощо.
3. Вдосконалення моделі, створення більш швидкого та менш затратного до ресурсів алгоритму.
4. Параметричне дослідження отриманих розв’язків. Винесення висновків, рекомендацій та схем до побудови маршрутів у багатоагентних логістичних системах.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. *Германчук М.С.* Многоагентный подход к задачам типа многих коммивояжеров на сложных сетях // Дистанционные образовательные технологии, - 2018, с. 208-216.

2. *Poli R.* Analysis of the publications on the applications of particle swarm optimization // Hindawi Publishing Corporation Journal of Artificial Evolution and Applications Volume. - 2008, p. 10. DOI: 10.1155/2008/685175

3. *Dorigo M.,* Ant Colony Optimization // [Электронный ресурс] – Режим доступа: http://www.scholarpedia.org/article/Ant_colony_optimization

4. Ю. П. Титов, Модификации метода муравьиных колоний для решения задач разработки авиационных маршрутов // Автомат. и телемех., - 2015, Выпуск 3, с. 108–124.

5. *Li L., Cheng Y., Tan L., Niu B.,* A discrete artificial bee colony algorithm for TSP problem // D.-S. Huang et al. (Eds.): ICIC 2011, LNBI 6840, pp. 566–573, 2012. DOI: 10.1007/978-3-642-24553-4_75

6. *Закиян Х.С., Частикова В.А.,* Адаптация алгоритма интеллектуальных капель воды для решения задач комбинаторики // Научные труды КубГТУ, 2015, № 12. Режим доступа: https://ntk.kubstu.ru/file/692

7. *Смирнова О.С., Богорадникова А.В., Блинов М.Ю.* Описание роевых алгоритмов, инспирированных неживой природой и бактериями, для использования в онтологической модели. – International Journal of Open Information Technologies ISSN: 2307-8162 vol.3, no. 12, 2015. – С. 28-37.

8. *Колесников А.В., Кириков И.А., Листопад С.В., Румовская С.Б., Доманицкий А.А.* Решение сложных задач коммивояжера методами функциональных гибридных интеллектуальных систем / Под ред. А.В. Колесникова. — М.: ИПИ РАН, 2011. — 295 с., ил. — ISBN 978-5-902030-88-1

9. *Фам С.К.* Методика планирования полета легкого беспилотного летательного аппарата: Дис. … канд. техн. наук. М., 2013. – 155 с.

10. *Моисеев Д.В.* Анализ устойчивости оптимальных маршрутов полета БПЛА с учетом прогноза ветра // Труды XVII Международного научно-технического семинара «Современные технологии в задачах управления, автоматики и обработки информации» -СПб: РИЦ ГУАП, 2008. – С. 172.

11. *Фам С.К., Моисеев Д.В.* Свойства оптимальных замкнутых маршрутов полета легкого самолета с учетом прогноза ветра // Электронный журнал ―Труды МАИ‖, 2012, № 52, www.mai.ru/science/trudy/.

12. *Чинь В.М.* и др. Маршрутизация полета легкого беспилотного летательного аппарата в поле постоянного ветра на основе решения разновидностей задачи коммивояжера // Электронный журнал ―Труды МАИ‖, 2015, № 79, www.mai.ru/science/trudy/.

13. *Ceccarelli Nicola, Enright John J., Frazzoli Emilio, Rasmussen Steven J. and Schumacher Corey J.* Micro UAV Path Planning for Reconnaissance in Wind. Proceedings of the 2007 American Control Conference. New York City, USA, July 11-13, 2007.

14. *Таргамадзе Р.Ч., Моисеев Д.В., Фам С.К.* О рациональном выборе замкнутого маршрута полета легкого летательного аппарата с учетом прогноза ветра // Вестник ФГУП НПО им. С.А.Лавочкина. 2012. № 3. С. 76-83.

15. *Моисеев Д. В.* Вычислительные аспекты и прикладное программное обеспечение оптимальной маршрутизации полета легкого беспилотного летательного аппарата в поле постоянного ветра // Интернет-журнал «Науковедение» - 2017, Том 9, №3.

16. *Сигал И.Х., Иванова А.П.* Введение в прикладное дискретное программирование: модели и вычислительные алгоритмы. - М.: ФИЗМАТЛИТ, 2003, - 240 с.

17. *Козлов М.В., Костюк Ф.В., Сорокин С.В., Тюленев А.В.* Решение задачи коммивояжера методом целочисленного линейного программирования с последовательным исключением подциклов: описание и алгоритмическая реализация // Advanced Science. 2012. №2. С. 124-141.

18. *Козлов М.В., Костюк Ф.В., Сорокин С.В., Тюленев А.В.* Решение задачи коммивояжера методом целочисленного линейного программирования с последовательным исключением подциклов: обоснование, тестовые испытания, применение // Advanced Science. 2012. №2. С. 142-159.

19. *Иглин С.П.* Математические расчеты на базе MATLAB. – СПб.: БХВ – Петербург, 2005. – 640 с.

20. *Вагнер Г.* Основы исследования операций, Том 2. - М.: «Мир», 1973, 488 с.

21. *Раскин Л.Г., Кириченко И.О.* Многоиндексные задачи линейного программирования. М.: Радио и связь, - 1982, -242 с.