МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет радіоелектроніки
Факультет Комп'ютерних наук
Кафедра Програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

Дослідження методів побудови енергоефективного маршруту для дронів.

Виконав:
студент 2 курсу групи ІПЗм-21-3
Матирін В.А.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення

Тип програми Освітньо-наукова

Керівник доц. Чуприна А. С.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ З.В.Дудар

2023 р.

Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерних наук_____
(повна назва)

Кафедра _____Програмної інженерії_____
(повна назва)

Рівень вищої освіти_____другий (магістерський)_____

Спеціальність _____121 – Інженерія програмного забезпечення_____
(код і повна назва спеціальності)

Тип програми _____освітньо-наукова програма_____
(освітньо-професійна або освітньо-наукова)

Освітня програма _____Інженерія програмного забезпечення_____
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____20 ____ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студента _____*Матиріна Вадима Андрійовича*_____
(прізвище, ім'я, по батькові)

1. Тема роботи «Дослідження методів побудови енергоефективного маршруту для дронів.» _____

затверджена наказом університету від «03» квітня 2023 р. № 83Стз

2. Термін подання роботи до екзаменаційної комісії «10» травня 2023 р.

3. Вихідні дані до роботи *методи обоходу графів, метрики, критерії їх оцінки, алгоритм Дейкстри, паралельна імплементація алгоритму, C++, C#, WinForms.*_____

4. Перелік питань, що потрібно опрацювати в роботі *вступ, аналіз предметної області, аналіз алгоритмів знаходження найефективнішого маршруту, критерії їх оцінки, оптимізація обраного алгоритму, проведення*

експериментів та аналіз їх результатів, програмна реалізація прототипу, формування подальших рекомендації.

5. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Спецчастина	доц. Чуприна А.С.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз предметної області	25.02.2023	виконано
2	Постановка задачі	10.03.2023	виконано
3	Проведення дослідження	01.04.2023	виконано
4	Підготовка пояснювальної записки	28.04.2023	виконано
5	Підготовка презентації та доповіді	02.05.2023	виконано
7	Перевірка на академічний плагіат	03.05.2023	виконано
8	Нормоконтроль	04.05.2023	виконано
9	Рецензування	05.05.2023	виконано
10	Знесення диплома в електронний архів	07.05.2023	виконано
11	Допуск до захисту у зав. кафедри	08.05.2023	виконано

Дата видачі завдання 17.01.2023 р.

Студент _____
(підпис)

Керівник роботи_____ доцент Чуприна А.С.
(підпис)

РЕФЕРАТ / ABSTRACT

Кваліфікаційна робота магістра містить: 63 стор., 17 рис., 2 табл., 17 джерел, 5 додатків.

ДРОН, МАРШРУТ, ЕНЕГРОЄФФЕКТИВНІСТЬ, ПОБУДОВА МАРШРУТУ, ГРАФИ, ОБХІД В ШИРИНУ, ДЕЙКСТРА, ОБХІД В ШИРИНУ, АЕРОРОЗВІДКА, ВИКОРИСТАННЯ В БОЙОВИХ УМОВАХ.

Об’єктом дослідження є методи побудови маршрутів на основі графів.

Метою дослідження є аналіз різних за складністю та структурою методів обходу графів для виявлення найбільш енергоефективного маршруту для дронів для виконання аеророзвідки.

Методи розробки базуються на основі архітектурних підходів для обходження графів, мові програмування C++, C# та їх бібліотек.

У результаті роботи було проведено аналіз існуючих методів обходу графів та виявлено найбільш енергоефективний для побудови маршрутів для дронів, результати, якого були продемонстровані на створеному прототипі.

DRONE, ROUTE, ENERGYEFFICIENCY, ROUTE BUILDING, GRAPHS, BREADTH-FIRST SEARCH, DIJKSTRA, AERO RECONNAISSANCE, BATTLEFIELD APPLICATIONS.

The object of research is building routes based on graphs.

The aim of the study is to analyze different in complexity and structure methods of traversing graphs for identifying the most energy efficient route for drones for conducting aero reconnaissance.

Development methods are based on architectural approaches to graphs traversal, C++, C# and their libraries.

As a result, a thorough analysis of existing graph traversal methods was conducted and the most energy efficient route building method for conducting aero reconnaissance using drones was found and the results were demonstrated on the created prototype.

Я, Матирін Вадим Андрійович, студент групи ІПЗм-21-3, здобувач вищої освіти на другому (магістерському) рівні, кафедра Програмної інженерії, заявляю: моя кваліфікаційна робота на тему «Дослідження методів побудови енергоефективного маршруту для дронів.», що буде представлена до ЕК для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElArKhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП 9
1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ 11
1.1 Загальний аналіз галузі 11
2 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ТА АЛГОРИТМІВ 18
2.1 Аналіз підходів обходження графів 18
2.2 Пошук в ширину 20
2.3 Пошук в глибину 21
2.4 Двосторонній пошук 23
2.5 Алгоритм Дейкстри 24
2.6 Дослідження графу 26
2.7 Вибір алгоритму 28
3 ОПТИМІЗАЦІЯ АЛГОРИТМУ ДЕЙКСТРИ 30
3.1 Актуальність використання алгоритму Дейкстри 30
3.1 Аналіз підходів оптимізації алгоритму Дейкстри 31
3.2 Вибір структури даних 32
3.3 Паралельна реалізація алгоритму 34
3.4 Генерування тестових даних 36
3.5 Тестування алгоритму 38
4 ПРОГРАМНА РЕАЛИЗАЦІЯ ПРОТОТИПУ 40
4.1 Проектування прототипу 40
4.2 Врахування впливу вітру на енерговитрати дрону 41
4.3 Побудова графу 42
4.4 Програмна реалізація 42
ВИСНОВКИ 47
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 49
ДОДАТОК А 52
ДОДАТОК Б 53

ДОДАТОК В.....54
ДОДАТОК Г.....62
ДОДАТОК Д.....63

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

UAV	Unmanned aerial vehicle, бпла
RPA	Remotely-piloted aircraft, бпла з можливісттю дістанійного керування
JPEG	Joint Photographic Experts Group
PNG	Portable Network Graphics
GIF	Graphics Interchange Format
BMP	Bitmap
DLL	Dynamic-link library

ВСТУП

У сучасному світі використання безпілотних апаратів (БПЛА) вже дуже багато років становить звичайну частину нашого життя. Вже майже 20 років комерційні дрони використовуються як звичайними людьми, так і професіоналами. Будь-хто може піти до звичайного магазину електроніки та придбати собі невеликий дрон, який він зможе використовувати для нескінченного списку задач – це може бути звичайне бажання погратися з чимось що літає, або використовувати їх для створення фільмів, або фотографій. У цей же час дрони використовуються великими бізнесами та корпораціями для розпилення удобреній на полях з різними агрокультурами, такі великі компанії як Амазон вивчають можливості їх використання для доставки товарів, а військові з усього світу для ведення бойових дій.

Завдання роботи – аналіз та надання можливості використовуючи задані маршрутні точки, на яких дрон має провести аеророзвідки, енергоефективного маршруту, для збільшення часу який дрон зможе перебувати у повітрі, та збільшення точок його можливого маршруту.

Метою роботи є дослідження використання існуючих алгоритмів теорії графів, ти вивчення можливості їх використання для створення програмної система за допомогою якої можливо прокладати енергоефективні маршрути для різних видів дронів.

З настанням нового етапу Російсько-Української війни, коли Російська Федерація почала широкомасштабне вторгнення на територію України 24го лютого 2022 року життя кожного українця змінилось на завжди. Від загарбницьких дій Росії вже загинули тисячі мирних жителів України, а мільйонам людей довелось шукати прихисток на території Європейського Союзу, або всередині України, змінюючи місце проживання з районів, де проходять бойові дії всередину країни. Вже у перші дня широкомасштабного вторгнення стало зрозуміло, що наміри захопити Київ за 3 дні провалилися, і що Україна чинить надзвичайний опір перед

окупантом, використовуючи всі наявні засоби ведення війни. Стало зрозуміло що дуже довгі лінії забезпечення військ загарбників всередині країни вразливі для атак, тому вони стали пріоритетними цілями ЗСУ з самих перших годин війни. Основним засобом ведення розвідки ліній забезпечення ворога стали звичайні комерційні дрони виробництва таких компаній як Dji, та безпілотники турецького виробництва «Байрактар». Для завдання ударів по лініям забезпечення та скупченням ворога спочатку окремі підрозділи аеророзвідки, або окремі бригади, запускали звичайні дрони, які вони використовували для знаходження пріоритетних цілей для атаки на противника, а потім завдавали удари ударними безпілотниками, або використовуючи артилерію, або інші засоби завдання ударів.

Тобто, від самого початку війни стало зрозуміло, що від використання дронів залежить успіх тих чи інших ударів по противнику, і та сторона яка має у себе більшу кількість безпілотних летальних апаратів має значну перевагу при нанесенні ударів, та має вирішальне значення для ходу бойових дій. Це також дай змогу зрозуміти чому існує велика кількість зборів засобів для тих чи інших бригад, коли люди пожертвуваннями збирають на дрони, або можливо навіть згадати державну ініціативу «Армія дронів», коли людей закликали жертвувати кошти на закупівлю великої кількості дронів для ЗСУ.

Отже, питання побудови енергоефективних маршрутів для дронів є досить актуальним, бо у бойових умовах кожен виліт дрону це ризик як для його пілота, який керує їм дистанційно, та для самого дрону, бо можливо проведення радіоелектронних атак, для перехоплення керування. Тому за один виліт дрону є велика необхідність мати можливість провести розвідку найбільшого набору певних точок, з врахування зовнішніх факторів таких як вітер і так далі, з врахуванням максимальної дальності та часу, який дрон може перебувати у повітрі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Загальний аналіз галузі

Використання дронів як під час мирного часу, так і під час військових дій з кожним роком лише зростає, та збільшує кількість людей та компаній, які проявляються цікавість у тому що щоб використовувати цей ринок у власних компаніях та для власних проблем, серед яких наприклад використання дронів для відстежування природних явищ для їх подальшого аналізу, та для використання цієї інформації для подальшого впровадження нових засобів перешкоджання тим чи іншим природним явищам. Вже багато років безпілотні летальні апарати використовуються збройними силами багатьох країн, їх можливо використовувати для ведення як аеророзвідки так і для нанесення ударів по цілям. Безпосередньо лідером країн які використовують ударні безпілотники можливо назвати Сполучені Штати.

Спроби використання дронів під час бойових дій почалися від початку створення летальних апаратів – тобто ще з Першої Світової війни, коли різні країни намагалися створити безпілотні літаки для ведення аеророзвідки, але їх широке використання стало лише можливо після впровадження та використання реактивних двигунів на летальних апаратах після Другою Світової війни, а першим їх великим використанням можливо назвати війну в В’єтнамі, коли армія США використовувала їх як приманку для ППО, та для розвідницьких цілей. Після цього Штати посили свої розробки БПЛА, для можливостей ведення як аеророзвідки, так і для нанесення ударів. Перші зразки новітніх ударних дронів «MQ-1 Predator» з’явились у середині 90-х, з подальшою еволюцією у ударний БПЛА «MQ-9 Reaper», який використовувався у великій кількості збройних конфліктах Сполученими Штатами, та уявляє собою високотехнологічний дрон, що може нести на собі навіть протитанкову зброю.

Але використання таких дронів має свої недоліки, армія США розробляла та використовувала свої ударні БПЛА у збройних конфліктах, зазвичай проти

невисокорозвиненого супротивнику, що накладає певні обмеження у його використанні у Російсько-Українською війни. Армі РФ має на озброєні велику кількість ППО, що накладає обмеження у використанні таких великих дронів для проведення аеророзвідки, та збільшує можливість що таких дрон буде збитий, а технології, які використовувались під час його створення зможуть попасти до рук противнику. Тому у сучасному збройному конфлікту використання легких безпілотників, або дронів довело свою необхідність та ефективність

Легкі дрони які використовують підрозділи ЗСУ використовуються для трохи інших задач ніж ударні безпілотники, та можливо навіть сказати, що вони мають більш вирішальну роль під час ведення бойових дій ніж останні. Серед легких дронів які використовують ЗСУ можливо відокремити два типи, за типом використання – для розвідки та для завдання ударів. Прикладом останніх є використання дронів з підвищеною можливістю перевезення вантажів, для завдання ударів по супротивнику. Але найголовнішим типом використання безпілотників все ж є їх використання для аеророзвідки підрозділів супротивника, для подальшого їх знищення з використанням артилерії, РСЗО або інших засобів ураження. Також виконуючи дронів для проведення аеророзвідки місцевості, отриману таким чином інформацію використовують для планування подальших наступальних наземних операцій, та для прокладання маршрутів пересування.

Тобто дрони є чи не найголовнішою частиною завдання ударів підрозділами ЗСУ, та від їх можливості отримати найбільшу кількість інформації залежить можливість ці удари проводити. Самі ж дрони для аеророзвідки також можливо поділити на декілька видів, серед яких дрони які використовують тій же механізм для отримання під ’ємної сили, як і гелікоптер, тобто обертання лопастей, ті дрони які зазвичай мають в свої конструкції механізми створення під’ємної сили як і літаки – крила та двигун, зазвичай з пропелером. Можливості проведення аеророзвідки дронами з двигунами внутрішнього згорання зазвичай обмежені їх можливістю переносити в собі кількість палива. Їх використання також зазвичай робиться з великої висоти, для того, що їх ідентифікація супротивником не була такою легкою, та для зменшення залежності від погоди у землі. Використання

таких типів дронів зазвичай потребує спеціального комплексу, який складається з самих дронів, пускової установки, та з командного пункту для керування та отримання телеметрії у реальному часі. Необхідність наявності використання цілого комплексу для цих дронів, обмежує їх можливість швидкого використання на нових ділянках, де потрібно швидко проглянути цікаві точки. Саме тому велика кількість підрозділів має у себе легкі, комерційні дрони, які можливо використовувати у різних ситуаціях.

Використання легких дронів, від таких компаній як «Dji» надає можливість їх швидкого отримання підрозділами які їх потребують. Ці дрони мають розвинену систему стабілізації та керування зображенням, що дає можливість пілоту отримувати інформацію з камери, яка розташована на дроні у реальному часі на пульт, або навіть на свій телефон, та з додатковою можливістю отримання інформації після розвідки з внутрішнього накопичувача дрону у більшій роздільний якості та з більшою кількістю кадрів в секунду. Саме можливість запустити дрон з будь-якої поверхні, або навість з руки, зробила використання цих дронів серед лав ЗСУ такою поширеною, бо вона також не потребує попереднього навчання керування дроном, та дає можливість швидко отримати дані розвідки того чи іншого місця.

Отже використання легких дронів для розвідки ж надзвичайно важливим та впливає у першу чергу на усі бойові дії, які Україна веде проти Російської Федерації. Єдиною проблемою використання цих дронів можливо навести те що, через те що вони використовують вбудовані акумуляторі їх використання може бути обмежено погодними умовами, та тим наскільки дрон ефективного витрачає енергію на завдання аеророзвідки. Тому проблему побудування маршруту для таких дронів, з врахуванням погодних умов, підрозділів супротивника, які можуть вести радіоелектронну боротьбу з бажанням перехопити керування дроном є дуже важливою, бо маючи можливість отримати найоптимальніший маршрут за енергоефективністю для того щоб охопити найбільшу кількість точок на землі, для яких потрібно провести розвідку, підрозділи отримують можливість більш ефективного знаходити та ліквідувати цілі супротивника, бо менша кількість часу

буде витрачатися на повернення дронів для їх підзарядки, або на те щоб провести заміну акумуляторів. Використання непрокладеного маршруту нарікає дрони на небезпеку бути перехопленими супротивником їх засобами радіоелектронної боротьби, та на те що зв’язок з дроном може бути втрачено. Тобто можливо казати про те що наявність швидкого методу побудови маршруту найменших витрат енергії для дронів дозволить підрозділам працювати більше ефективно, зменшить їх навантаження на засоби підзарядки дронів, та дасть можливість отримувати розвіддані швидке на ефективніше, що у загальний перспективі зможе збільшити кількість цілей супротивника, які можливо буде ліквідувати за більш менший проміжок часу.

Оптимізуючи кожен з моментів використання дронів можливо отримати реальні можливості збільшення кількості разів, коли використання дронів для розвідки буде успішним, а кількість разів коли супротивник встигне покинути свої позиції зменшиться.

1.2 Виявлення проблем

У сучасній ситуації та сучасній війні можливість швидкого розгортання та використання дронів стала її невід’ємною частиною. Використання дронів у повсякденному житті також стає звичайною справою. Дрони використовують для доставки товарів, для відео нагляду над агрокультурами або для відстеження пересування скота. Усі ці операції потребують значних витрат енергії, та не є значно ефективними, бо наприклад у випадку доставки товарів, користувачам складно самім розробити маршрут, який дозволить максимально ефективно зробити обхід усіх існуючих точок доставки. Наявність різних застосувань однакових дронів створює проблему оптимізації їх енергоефективності. Так наприклад задача оптимізації доставки товарів використовуючи дрони з великою масою, та з великими двигунами може створювати інші умові задачі

енергоефективності ніж маленькі дроні, які можуть літати на великої швидкості, та можливо використовувати для цілей розвідки, або слідкування. Також обмеженість використання дронів у деяких місцях, наприклад з відсутністю інтернет з'єднання, може завадити використовувати методи для прокладання маршрутів які доступні через інтернет і так далі.

Тому проаналізувавши види дронів, методи їх використання у бойових діях та у громадянському житті, я прийшов висновку, що аналізувати предмет дослідження слід на легких дронах, а створення алгоритму побудови енергоефективного маршруту в умовах відсутності інтернет з'єднання може бути потім використано для використання на комерційних дронах, наприклад для аналізу землі, або природних явищ в віддалених кутках планети.

1.3 Аналіз існуючих систем

Наступним кроком слід проаналізувати вже існуючи засоби прокладання маршрутів для дронів, які можуть прокладати маршрути серед наведених точок в різних умовах. Зараз більшість програмних систем представлених на ринку, які надають можливості прокладання маршрутів створені для дронів компанії «Dji», бо їх дрони є найпоширенішими комерційними дронами, які використовуються по усьому світу, та мають найбільшу кількість користувачів, та документації, щодо їх використання.

Першою програмною системою, яка існує на ринку є система з назвою «Dji GS PRO»[2] (рис. 2.1). Система розроблена та рекламується компанією розробників дронів «Dji» для власних дронів на системі та для операційної системи iOS, та дає можливість створювати маршрути для дронів, для їх автоматизованого обходження з можливістю створення 3Д мап різних областей, або проведення дій на кожній з точок маршруту.

Система також дає можливість створювати проекти в яких можливо додавати

декілька користувачів, кожен з яких матиме свої ролі та доступи щодо змін даних у проекті. Також за наведеними даними продукт можливо використовувати з великою кількістю різних продуктів від «Dji», серед яких дрони, камери та наземні станції.

Рисунок 1.1 – Демо «DJI GS PRO» від «DJI»

Серед основних пунктів, які «Dji» пропонує щодо цього застосунку, є:

– можливість створювати різні місії, та додавати до них різних пілотів з різними ролями та правами;

– можливість сканування дроном спеціально обраної зони за автоматично згенерованим маршрутом;

– автоматичне виконання дій на деяких точках маршруту;

– перегляд логів під час та після виконання польоту за маршрутом;

– створення 3Д мапи місцевості за зробленими під час польоту знімкам;

– створення аеро фотографій сканованої місцевості;

– створення 3Д моделей великих структур, таких як статуї, або будівлі.

Іншим застосуванням яке надає можливість прокладання маршрутів є «DjiFlightPlanner»[3] (рис. 1.2), який можливо використовувати для створення маршрутів для повного обльоту тієї чи іншої зони для дронів «Dji». Програмне застосування потребує використання ПК на операційній системі Windows та

мобільного застосування для iOS або Android.

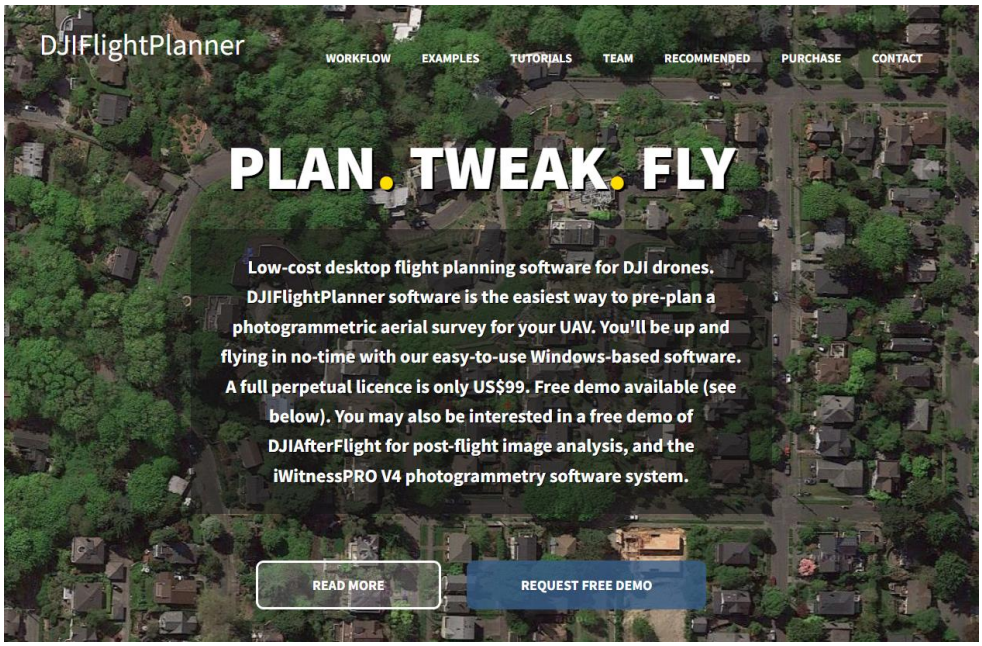


Рисунок 1.2 – Демо застосунку «DjiFlightPlanner»

Серед функціоналу та перевагами програмного застосування вказано наступний функціонал:

– невелика ціна в порівнянні з іншими застосунками;

– автоматична побудова маршрутів для певної області;

– редагування автоматично створених точок маршруту з можливістю їх редагування або видалення;

– збереження та експорт створеного маршруту у мобільний додаток керування польотами.

Проаналізувавши та переглянувши приклади програмних систем які дозволяють автоматизувати прокладання маршрутів для дронів можливо зробити висновок, що усі вони мають деякі недоліки, з яких такі як – необхідність інтернет підключення для створення маршруту, відсутність можливість задати певні обмеження серед яких зони які потрібно уникати та так далі. Та усі вони не мають можливість побудови енергоефективного маршруту, який можливо використати для усіх дронів, а не лише для дронів виробництва «Dji»

2 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ТА АЛГОРИТМІВ

2.1 Аналіз підходів обходження графів

Вивчення графів та робота з ними називається теорія графів. Графи являється з себе математичні структури, які можливо використовувати під час моделювання відносин між деякими об'єктами (рис 2.1). Загалом будь-який граф буде складатися з деякого набору об’єктів – вершин, які також часто кличуть точками або вузлами, між якими є деякі пов’язання, тобто ребра, або посиланнями. Існують декілька видів графів серед яких: неорієнтований граф, в якому кожне ребро симетрично пов’язує дві інші вершини, та орієнтований граф, я якому посилання пов'язують дві вузли несиметрично. Ці та інші графи грають основу роль при вивченні дискретної математики [5].

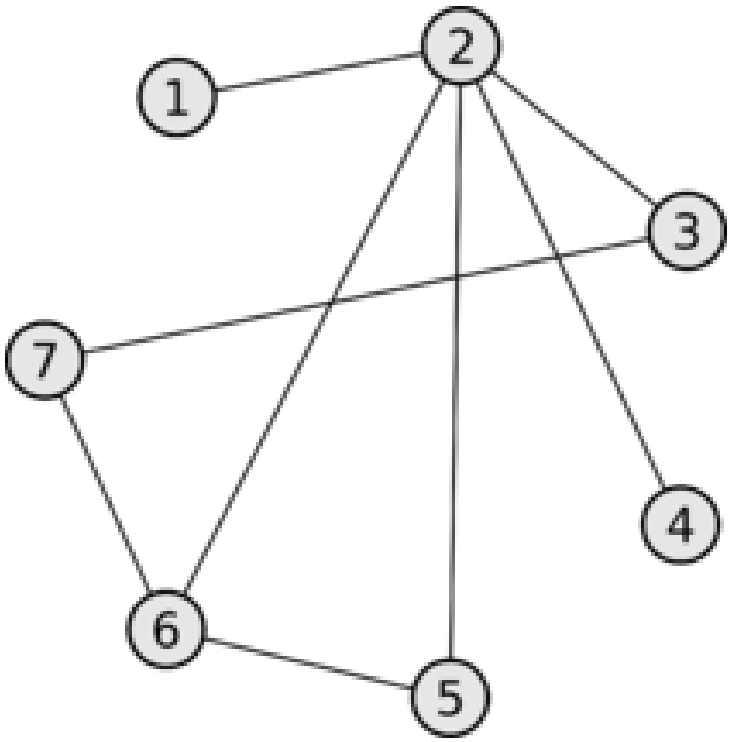


Рисунок 2.1 – Схематичне зображення графу з 6 вершинами

Вже дуже великий час графи перебували та продовжують перебувати основними конструкціями технологій та алгоритмів, але також їх використовують

для моделювання взаємодії між об’єктами у таких областях як біологія, фізика та хімія. Багато алгоритмів які ми використовуємо у звичайному житті, такі як побудова маршруту у Google Maps можливо представити як графи, що широко використовується різними компаніями.

Є велика кількість алгоритмів які можливо використовувати для знаходження самого енергоефективного шляху, але усі вони під собою використовують базові алгоритми, серед яких:

– пошук в ширину (BFS, Breadth-First Search);
– пошук в глибину (DFS, Depth-First Search);
– двосторонній пошук;
– алгоритм Дейкстри.

Усі наведені алгоритми мають певні властивості які притаманні лише їм. Тому для вирішення задачі знаходження алгоритму який можливо використовувати для знаходження енергоефективного маршруту потрібно зробити аналіз кожного з них, та зрозуміти який буде найвлучніше використати для наведеної задачі.

Для коректного аналізу алгоритмів та їх використання потрібно також зазначити, що серед графів існую також додаткова кваліфікація. Графи можуть складатися з ребер, на кожному з яких буде так звана вага, а також бути орієнтованими, або не орієнтованими, мати в складі цикли – путь всередині графа, в якому перший та останні елементи однакові.

Використовуючи вхідні дані задачі можливо припустити, що у нашому випадку самою головною характеристикою моделі графу буде граф, в якому ребра матимуть певну вагу, що буде відображувати реальні значення витрат енергії дроном на цьому путі муж двома точками маршруту. Також теоретично можливо уявити ситуацію коли ребро може бути одно направленим, тобто пересування дроном між двома точками маршруту можливо лише у одному напряму, протилежна ж ситуація буде у випадку ребра, по якому можливо пересуватися в обидві стороні, тобто при створені маршруту пересування цим шляхом доступно як при проходженні в глибину графу, та і у зворотному напрямку при повертання

до початкової вершини, що загалом є характерним для звичайного, не направленого графу, саме для такого будуть розглянуті алгоритми в цьому розділі.

2.2 Пошук в ширину

Пошук в ширину – алгоритм обходження графу, який можливо використовувати для знаходження найкоротшого шляху (рис. 2.2). Алгоритм відносно легкий у використанні та програмній реалізації, та його можливо застосовувати для усі графів, які не мають ваги на ребрах:

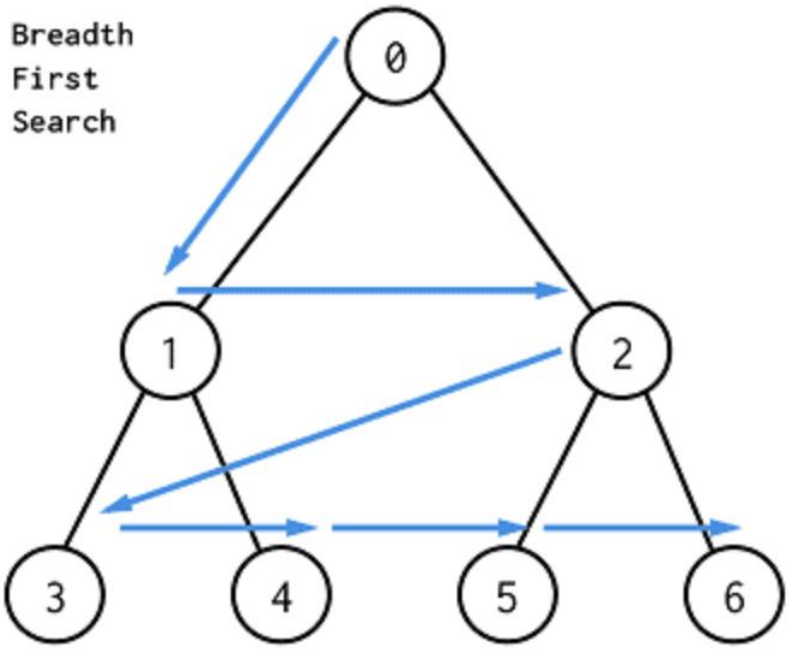


Рисунок 2.2 – Робота BFS

Схематично зображено робота алгоритму пошуку в ширину. Обирається початкова вершина, і починаються обхід інших вершин, з якими вона зв’язана. Після того як алгоритм «завітав» в усі вершини з якими була зв’язана початкова вершина, переходимо до обходу вершин які зв’язані з першою вершиною, з якою була зв’язана початкова вершина. Алгоритм продовжується поки ми не «завітаємо»

до усіх вузлів графу, або поки ми не дійдемо до вершини, яку ми позначили як фінальну, тобто ту до якої ми шукаємо найкоротший шлях[7].

Алгоритм обходу в ширину має певну специфіку, так наприклад сусід сусіда вершини не буде «відвідано» до того, як будуть «відвідані» прямі сусіди вершини, які мають з нею прямий зв'язок. Це призводить до того, що всі вершини, які мають деяку відстань від початкової вершини, будуть відвідані лише після відвідування усіх вершин відстань яких менше цієї відстані. Тобто алгоритм пошуку в ширину спочатку відвідає усі вершили з умовною відстанню нуль, а лише після цього всі вузли з відстанню один і так далі. Саме ця особливість надає нам можливість використовувати цей алгоритм для пошуку ефективного шляху, навіть у графах які мають в собі цикли.

Також можливо оцінити загальну складність алгоритму: хай k – найбільша кількість вершин-сусідів певного графу, а d – теоретично довжина найкоротшого путі між початковою вершиною, та вершиною до якої ми будемо намагатися знайти найкоротший шлях. Можливо сказати, що алгоритм має складність $O(k^d)$.

Пошук в ширину робить обхід графу на різних рівнях. На кожному рівні однакова відстань до початкової вершини, тобто щоб добратися до першого рівня кількість кроків складатиме $O(k^2)$. Через це для того щоб дістатися певного рівня d потрібна саме $O(k^d)$ кількість кроків.

2.3 Пошук в глибину

Пошук в глибину – чи не найвідоміший алгоритм для обходження графу, який можливо використати для знаходження найкоротшого маршруту від однієї вершини у графі до іншої (рис. 2.3). Але його використання обмежено деякими обмеженнями його використання та недоліками, серед яких умова того, що граф на якому буде використано алгоритм має не мати циклів, тобто бути ациклічним, а також усі ребра цьому графі повинні бути без ваги. При дотримані цих умов

можливо використовувати алгоритм пошуку в глибину для знаходження найкоротшого шляху[6].

У випадку відсутності маршруту між вершиною яку буде обрано як стартову, та кінцевої вершини, найкоротший шлях, який алгоритм покаже для цього випадку буде дорівнювати -1.

Схематично алгоритм можливо описати так – обирається початкова вершина, потім «відвідується» перша інша вершина до якої початкова має сполучення. Після цього така ж дія відбувається для цієї іншої вершини. Це повторюється для поки не буде «відвідано» вершину до якої ми шукаємо шлях, або доки у вершини не буде вершин з якими вона має зв'язок, які ще не були «відвідані». Далі алгоритм підіймається на рівень вище, та повторює всі ці дії доки фінальна вершина не буде відвідано, або доки не закінчиться усі не відвідані вершини.

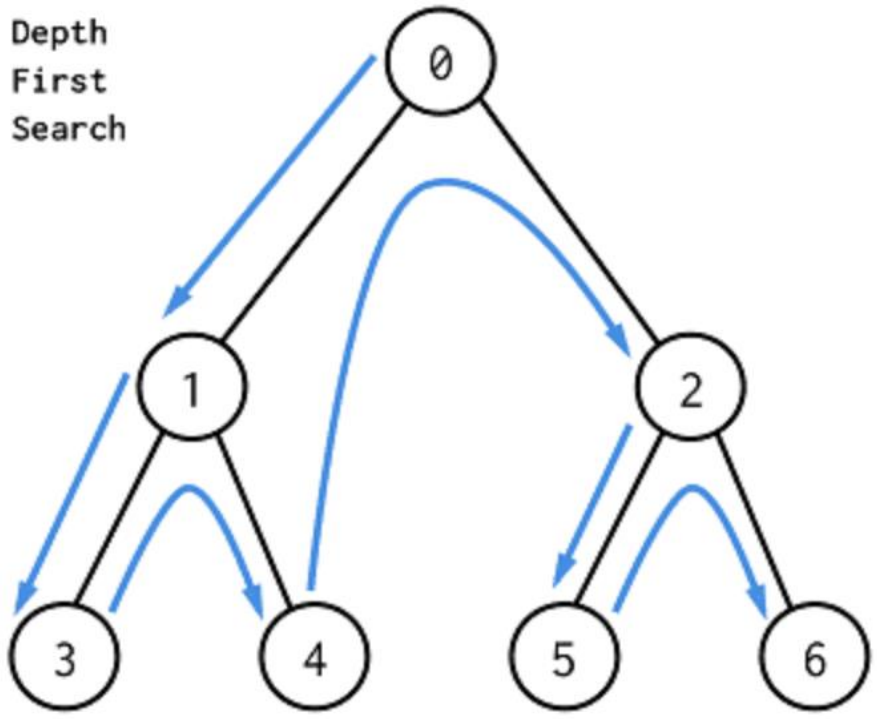


Рисунок 2.3 – Робота DFS

Серед обмежень цього алгоритму є неможливість його використання для графів які мають в собі цикли, це обумовлено тим що, коли ми знаходимо маршрут

немає чіткого розуміння що саме цей маршрут є найкоротшим. Саме цьому щоб мати можливість використовувати цей алгоритм на графах потрібно впевнитись, що він не має у собі циклів, тобто що цей граф є деревом, або переробити його структуру у таку, яка буде деревовидною.

Також можливо оцінити загальну складність алгоритму: хай k – кількість вершин графу, а d – кількість ребер певного графу. Загальна складність алгоритму дорівнює $O(k)$, а через рекурсивну особливість реалізації алгоритму, то його споживання асимптотичної пам'яті також дорівнюватиме $O(k)$.

2.4 Двосторонній пошук

Двосторонній пошук - ще одним алгоритм для знаходження найкоротшого маршруту і графу, який можливо як і пошук в ширину використовувати лише не неорієнтованих графах, у яких вершини не мають ваги.

Алгоритм представляє собою пошук в ширину, але замість того щоб починати пошук з початкової вершини, алгоритм працює паралельно – запускається один пошук в ширину на початковій вершині, а інший з вершини, до якої потрібно знайти найкоротший маршрут. Припинення алгоритму відбувається коли пошуки з цих двох вершин зустрічаються, тоді можливо говорити, що найкоротший шлях було знайдено.

Алгоритм не є найкоротшим під час виконання, але через те що він уявляє собою два алгоритми пошуку в ширину які працюють одночасно, його програмна реалізація буде досить простою (рис 2.4). Основна перевага цього алгоритму над звичайним пошуком в ширину, є то що він працює значно швидше, в середньому наближенні алгоритм дозволяє знайти найкоротший маршрут певною довжиною, то двосторонній пошук дозволить знайти маршрут з довжиною у двічі більшою [8].

Оцінюючи загальну складність алгоритму, можливо взяти за основу оцінку складності для звичайного пошуку в ширину, але заміть того щоб закінчуватися на

рівні d, двосторонній пошук буде закінчуватися після досягнення рівня d/2, бо цей рівень буде центром шляху від вершину початку роботи одного пошуку в ширини, та з іншого алгоритму з вершини кінця.

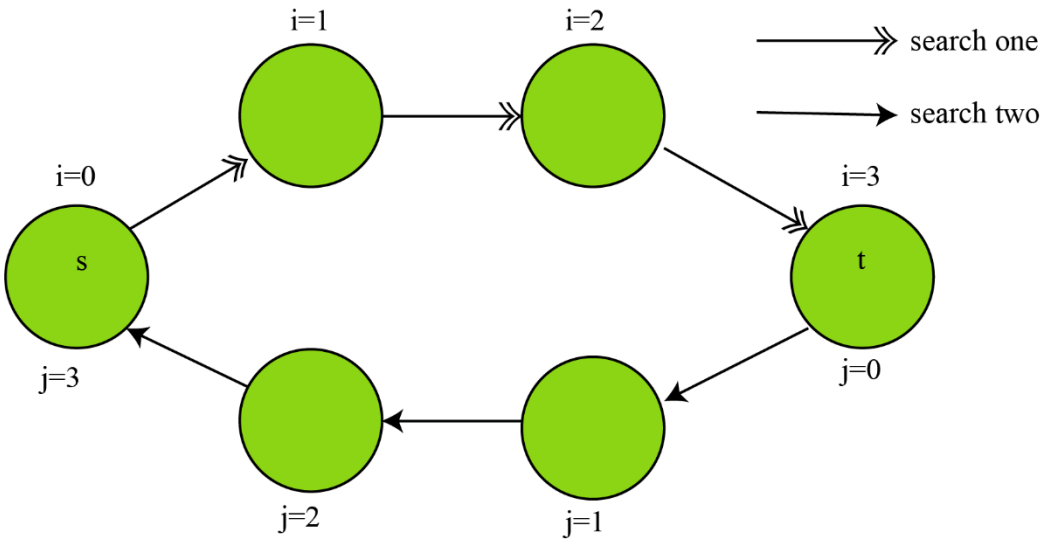


Рисунок 2.4 – Робота двостороннього пошуку

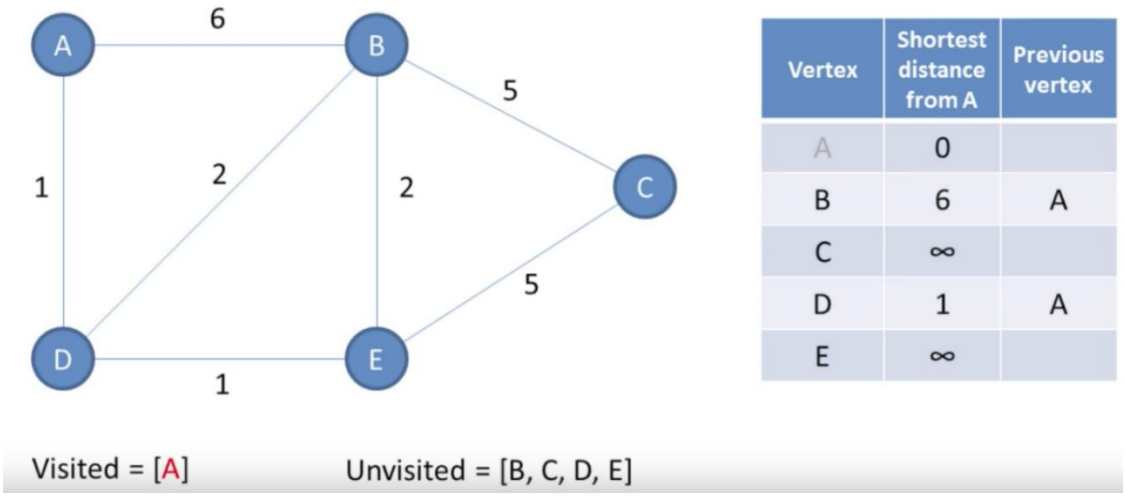
Тобто обидва пошуки в ширину будуть одночасно відвідувати $k^{d/2}$ вершин, що у загальній складності буде становити $2k^{d/2}$. Це призводить до того, що загальна складність алгоритму буде $O(k^{d/2})$, що говорить про те що двосторонній алгоритм пошуку швидший звичайного пошуку в ширину на коефіцієнт $k^{d/2}$.

2.5 Алгоритм Дейкстри

Алгоритм Дейкстри було винайдено Єдсгером Дейкстрой у 1959му році, і він також використовується для знаходження найкоротшого маршруту у графі, але на відміну від пошуку в ширину, в глибину та двостороннього його можливо використовувати з усіма графами, які мають ребра з певною вагою, але ще залишається обмеження, що через особливості його реалізації його використання можливо лише у випадку, коли ваги ребр, позитивні, тобто не мають у собі

негативних значень. Це обмеження використовується, бо у випадку негативної ваги припинення алгоритму для цього графу не можу гарантуватися, що унеможливлює його практичне використання [9].

Роботи алгоритму на прикладі вагового графу полягає у наступному: спочатку знаходиться найкоротший маршрут від початкової вершини до кожної іншої вершини, з якою в неї є зв’язок (рис 2.5).



Vertex	Shortest distance from A	Previous vertex
A	0	
B	6	A
C	∞	
D	1	A
E	∞	

Рисунок 2.5 – Проміжкові етапи алгоритму Дейкстри

Початкові значення маршрутів будуть дорівнювати нескінченності для усіх вершин, окрім початковою, для якої він буде дорівнювати 0, бо це буде значення відстані до самого себе. Для кожної з вершин з якою є зв’язок проставляється значення маршруту, яке дорівнює значенню ваги на ребрі. Після обходження усіх поточних вершин, алгоритм переходить до вершини з якою початкова вершина зв’язана ребром з найменшою вагою, а початкова вершина відмічається як пройдена. Після цього алгоритм проходить по усім вершинам які пов’язані з поточною вершиною, окрім тих що відмічені як ті, які вже були пройдені. Якщо значення маршруту для поточної вершини в сумі зі значенням ребра до наступної вершини менше її значення маршруту, то її значення зменшується на це. Після обходження усіх вершин, поточна помічається як пройдена і алгоритм помічає її як пройдену. Дії повторюються, доки усі вершини не будуть помічені як пройдені.

Загальна складність алгоритму в великій кількості залежить від методів програмної реалізації, та організування способу зберігання проміжних етапів роботи алгоритму, що може зумовлювати різні рівні складності його використання, від песимістичних $O(n^2)$, де n – кількість вузлів, до оптимістичних $O(nlog(n) + e)$, де e – кількість ребер у графі.

2.6 Дослідження графу

Для вирішення поставленої задачі для отримання енергоефективного маршруту для дронів, алгоритм який буде використано має мати можливість працювати з графом в якого ребра неорієнтовані, але мають вагу. Для коректного моделювання маршруту дронів у створюваному графі, кожне ребро матиме дві числові характеристики – одна це відстань у кілометрах між точками, інша це кількість використаної енергії на цей шлях (рис 2.7).

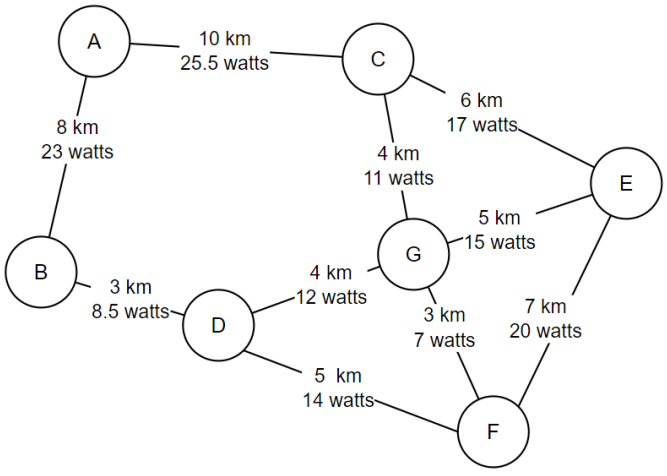


Рисунок 2.7 – Граф відстаней та витрат енергії

Використовуючи емпіричні методи знаходження найкоротшого маршруту за кількістю кілометрів для наведеного на рисунку 2.7 графу можливо отримати

наступі можливі маршрути між початковою вершиною A, який йде до фінальної вершини F:

– A-C-G-F (17 км);

– A-B-D-F (16 км);

– A-B-D-G-F (18 км);

– A-C-E-F (23 км);

– A-C-G-D-F (23 км).

Можливо побачити що маршрутом з найкоротшою відстанню від вершини A до F у кілометрах буде маршрут ABDF, який зображено на рисунку 2.8

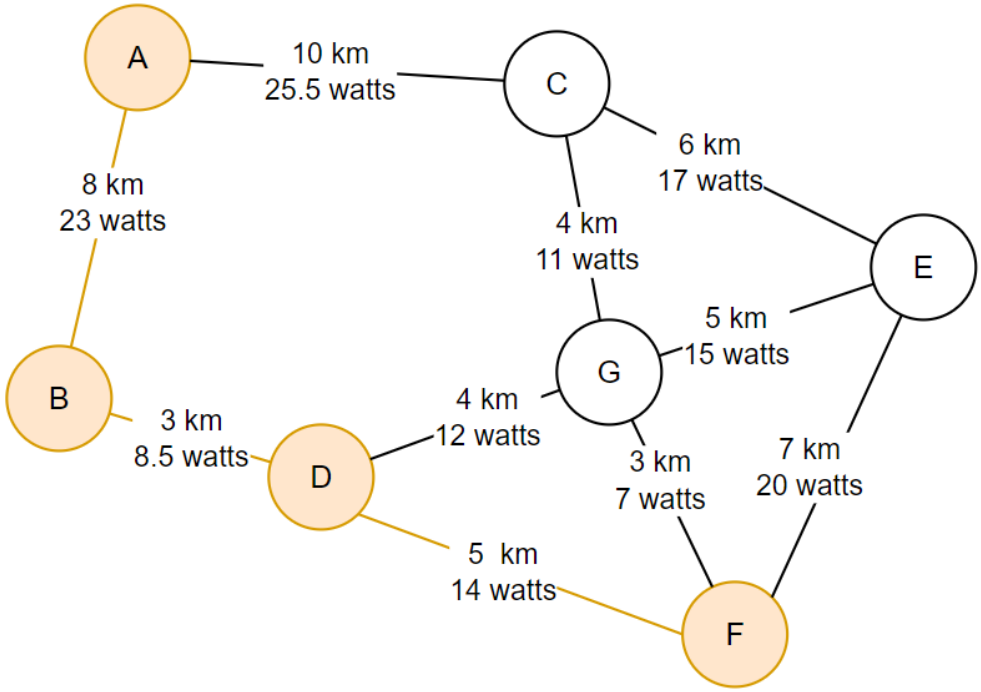


Рисунок 2.9 – Найкоротший маршрут

Далі розглянемо усі наступні маршрути, але будемо підраховувати не мінімальну відстань, а значення витрат електроенергії.

– A-C-G-F (43.5 ватт);

– A-B-D-F (45.5 ватт);

– A-B-D-G-F (50.5 ватт);

– A-C-E-F (62.5 ватт);

– A-C-G-D-F (62.5 ватт).

Тобто можливо зробити висновок, що самим енергоефективним є маршрут A-C-G-F, який зображено на рисунку 2.9.

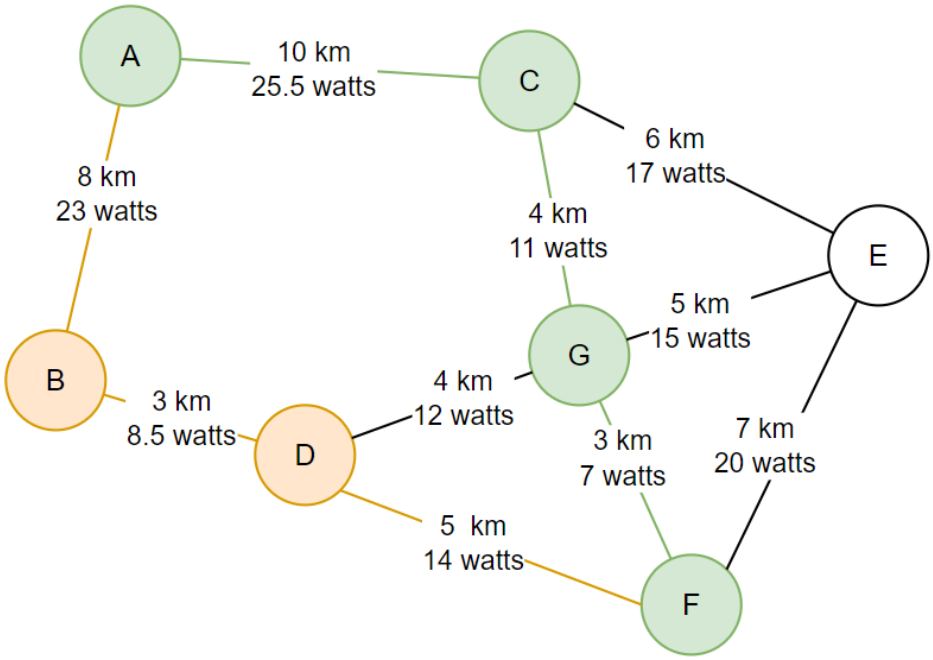


Рисунок 2.9 – Маршрут з найменшим споживанням електроенергії

Проаналізувавши наведену ситуації можливо зробити висновок, що для різних вхідних умов можливо отримати різні значення самого короткого та самого енергоефективного маршрутів. До можливих чинників цього ефекти можливо віднести ситуації коли на деяких ділянках дрону доводилось летіти проти вітру, або збільшувати свої висоту, на що витрачалась електроенергія, ніж на ділянках де цього робити не доводилось.

2.7 Вибір алгоритму

Проаналізувавши на практиці які проблеми потрібно вирішувати для знаходження енергоефективного маршруту від однієї точки до іншої для дронів

мною було зроблено висновок, що для цієї задачі підходить саме алгоритм Дейкстри, через його можливість використання на графах з вагами, у нашому випадку значенням ваг графів буде кількість електроенергії витраченої на проходження даного ребра. Також використання цього алгоритму можливо для графів які маються в собі цикли, томі це дозволить збільшити область його використання

В додаток до цього алгоритм дає можливість працювати з орієнтованими графами, без необхідності зміни його імплементації, що дає можливість використовувати цей алгоритм, для моделювання ситуацій, коли переліт дрону з одного місця в інше має різни витрати в залежності від напрямку руху. Така ситуація може відповідати погодним явищам, таким як вітер, напрям якого може прямо впливати на швидкість на енерговитрати на переміщення деяким маршрутом. Так наприклад попутній вітер може зменшити кількість витраченої електроенергії, але збільшити при подорожі цім же маршрутом, але у іншому напрямі.

Також для більшої наглядового використання алгоритму для побудови енергоефективного маршруту, та для отримання інформації щодо тих чи інших умов які впливають на енергоефективність маршруту, можливо одночасно використовувати алгоритм Дейкстри для обходження графу для знаходження як найкоротшого так і найбільш енергоефективних маршрутів. Це можливо використати для побудування мапи енерговитрат для досягнення тих чи інших точок маршруту, або неможливість їх досягнення, коли ваги графу мають дуже великі коефіцієнти.

Враховуючи це, та поширеність та відносну простоту реалізації цього алгоритму, можливо сказати, що саме його використання буде найбільш доцільним для вирішення задачі побудови найбільш енергоефективного маршруту для дронів.

3 ОПТИМІЗАЦІЯ АЛГОРИТМУ ДЕЙКСТРИ

3.1 Актуальність використання алгоритму Дейкстри

У певних умовах алгоритм Дейкстри є ефективним засобом розв'язання великої кількості оптимізаційних завдань на графах. Тому в даний час досить велика кількість робіт присвячена поліпшенню цього алгоритму для можливості його застосування в складних умовах [1-3].

У цьому плані[4] представлена вдосконалена версія алгоритму маршрутизації Дейкстри, що дозволяє оптимізувати не тривіальні метрики маршрутизації, ефективно виконувати багатопараметричну оптимізацію. У роботі [5] представлені результати моделювання та вирішення проблеми багатопроменевих шляхів, що не перетинаються, з метою забезпечення ефективної масштабованості. У роботі [6] представлені моделі та алгоритми для побудови оптимального шляху, що базуються на застосуванні алгоритму Дейкстри, для оптимізації шляху AGV. Основна мета дослідження – розробка алгоритму автономного планування маршруту наземного міського транспорту з використанням мап надійності сигналів та стільникового зв'язку. У роботі [7] представлено реалізацію модифікованого алгоритму Дейкстри для побудови двоетапної схеми оптимальної стратегії зарядки електромобіля; стратегія спрямована на мінімізацію витрат та деградації акумуляторної батареї автомобіля. У роботі [8] представлені модель та алгоритм для планування траєкторій на основі інтеграції пошуку по решітці та опуклій оптимізації. У роботі [9] представлено модель та алгоритми роботи ієрархічного планувальника швидкості для реалізації енергозберігаючих технологій для автоматизованих транспортних засобів. У роботі [10] запропоновано ефективне рішення для пошуку даних на основі протоколу системи доменних імен (DNS) з використанням деревоподібного підходу. У запропонованому протоколі сервери шукають інформацію, послідовно запитуючи предків у дереві. Дерево будується з урахуванням застосування модифікованої версії алгоритму Дейкстри. У роботі [11] вирішуються завдання апроксимації та

спрощення оптимального розподілу каналів із застосуванням реалізації алгоритму Дейкстри, жадібного алгоритму та пакетного розподілу. У роботі [12] представлено пошукову модель, яка є об'єднанням класичних пошукових підходів. У роботі реалізовано пошук оптимальних шляхів із застосуванням мереж без навчання. У роботі [13] представлено реалізацію алгоритму на основі комбінації алгоритмів Дейкстри та Хієрхольцера для моделювання маршруту із супутньою статистикою для оптимального планування маршрутів руху спеціалізованих транспортних засобів.

Аналіз сучасного стану питання свідчить про ефективність використання алгоритму Дейкстри та його модифікації для вирішення широкого кола задач з дотриманням певних обмежень. У цьому сенсі алгоритм Дейкстри також може бути ефективно використаний для оптимізації пошуку за структурами даних у задачах обробки зображень [14, 15], включаючи оптимізацію пошуку за структурами після використання нейронних мереж [16].

Тобто враховуючи велику кількість робот, які беруть за свою основу алгоритм Дейкстри можливо казати, що цей алгоритм є досить дуже актуальним і його використання та оптимізація можуть піти на користь у різних сферах застосування.

3.1 Аналіз підходів оптимізації алгоритму Дейкстри

Алгоритм Дейкстри дозволяє швидко знаходити найкоротший шлях від однієї вершини у графі до іншої, але для використання алгоритму у системах які потребують швидкого опрацьовування інформації є необхідним проведення його відповідної оптимізації, в залежності від конкретних задач, та на яких саме системах він буде виконуватись.

Є декілька основних методів за якими може бути оптимізований наведений алгоритм – оптимізація за допомогою структур даних, та оптимізація виконання за

допомогою різних засобів. Для оптимізації за допомогою структур даних зазвичай можуть використовуватися різні способи роботи з графом у програмному застосунку, серед яких: черга з пріоритетом, зв’язаний лист, або матриця суміжності.

Оптимізація за допомогою інших способів може включати в себе використання багато потокової технології сучасних процесорів. Використання декількох потоків одночасно дозволить значно зменшити час обробки великих графів, які можуть складатися з тисяч вершин, що значно покращить можливості використання алгоритму під час бойового зіткнення, для прокладання маршрутів дронів.

Майже всі серверні та клієнтські комп’ютери підтримують багато потокову обробку даних та запитів і є однією з найефективніших частин сучасних методів для обробки інформації.

Таким чином, оптимізація алгоритму Дейкстри скрадатиметься з обрання оптимальної структури даних, та з використанням багато потокової реалізації алгоритму. Щоб переконатися в ефективності та застосовності цих оптимізацій для пошуку найкоротшого шляху між початковим і кінцевим вузлом зваженого графа при використанні у сучасних серверних мовам програмування, а також клієнтських додатках буде обрано відповідну структуру даних, та проведено експеримент з порівнянна скорості виконання звичайної реалізації алгоритму, з її оптимізованою версією.

3.2 Вибір структури даних

Далі ми попитаємось встановити можливих способи збереження даних графа в пам’яті програми, яка буде використовуватися для виконання алгоритму. У нашому випадку розглянемо два можливі способи зберігання зваженого неорієнтованого графа для програм – список суміжності та матрицю суміжності.

Загалом ці два представлення графа можна використовувати як основу для інших різних методів роботи з графами, наприклад пріоритетної черги, яка також спираються на представлення списку суміжності з використанням черги, яка зберігає вершин із найкоротшим відстанями для того, щоб обробити у порядку їх ваги. Те саме можна застосувати до інших структур даних, таких як куча, а також багатьох інших.

Представлення графа за допомогою списку суміжності здійснюється шляхом визначення об’єкта (вузла) і створення списку або масиву всіх інших вузлів або вершин, до яких він має зв’язок. В кожній пов’язаній вершині також зберігається вага цього вузла. Застосовуючи це до графа, зв’язок між деякими точками в просторі з їх відстанями на основі графа можливо побачити на рисунку 3.1.

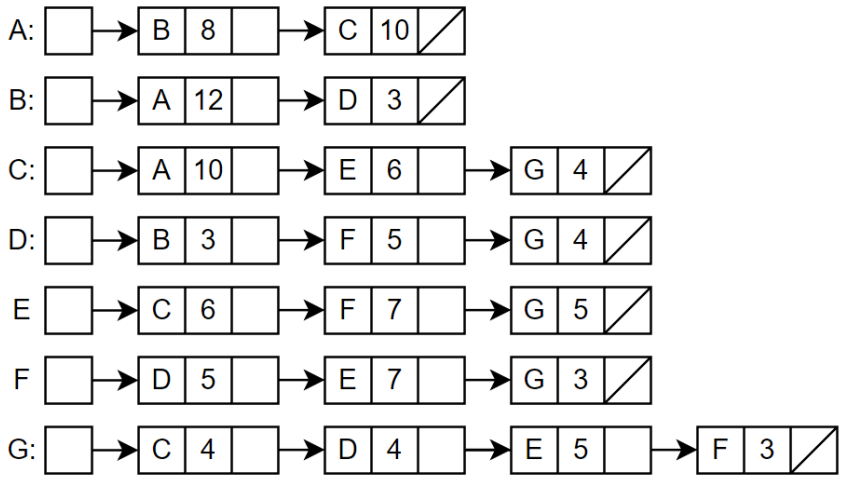


Рисунок 3.1 – Зважене неорієнтоване представлення графа з використанням списку суміжності

Другий спосіб полягає у використанні матриці суміжності. Матриця суміжності — це метод представлення графа у вигляді двовимірної матриці або масиву. У матриці суміжності кожен рядок і кожен стовпець представляють вершину в графі, а значення на перетині рядка і стовпця представляє ребро, що з’єднує вершини, представлені рядком і стовпцем. У цьому випадку незв'язність можна представити у вигляді нескінченно великого числа (рис. 3.2).

Використовуючи вище зазначені дані, можливо зробити висновок, що використання списку суміжності є кращим для загальної оптимізації, через можливість її більш використання у майбутньому для використання разом з пріоритетними чергами та купами для подальшої роботи.

Але матриця суміжності має своє місце завдяки своїй простоті та легкості представлення, та через легкість представлення у пам’яті, та через можливість виконання алгоритму у паралельному режимі. Саме завдяки використанню матриці суміжності можливо реалізувати паралельну версію алгоритму.

	A	B	C	D	E	F	G
A	0	8	10	∞	∞	∞	∞
B	8	0	∞	3	∞	∞	∞
C	10	∞	0	∞	6	∞	4
D	∞	3	∞	0	∞	5	4
E	∞	∞	6	∞	0	7	5
F	∞	∞	∞	5	7	0	3
G	∞	∞	4	4	5	3	0

Рисунок 3.2 – Зважене неорієнтоване представлення графа з використанням матриці суміжності для графа

Також слід зауважити той факт, що результуюча матриця суміжності для графу з рисунку 3.2 для неорієнтованого графу є симетричною над головною діагоналлю і використовуючи це, може бути зменшена в розмірі для зберігання на диску.

3.3 Паралельна реалізація алгоритму

Звичайна реалізація алгоритму Дейкстри працює в серії ітерацій, де кожна ітерація додає нову вершину до набору обчислених вершин. Однак обирання

декількох вершин одночасно може бути складною операцією, оскільки значення $d[v]$ для вершини v може змінюватися кожного разу, коли нова вершина u додається до набору S усіх вершин. Розпаралелювання ітерацій циклу while для обробки усього графу - непроста задача. Незважаючи на це, ітерації можна запускати паралельно, розділивши набір вершин V на p підмножин за допомогою блокового представлення даних. Кожна підмножина, що складається з $\frac{n}{p}$ послідовних вершин, призначається окремому процесу для обробки.

Для $i = 0, 1, \ldots, p-1$, підмножина вершин V_i призначається процесу p_i. Кожен процес p_i містить частину масиву d, яка відповідає V_i (як показано на малюнку 3.3).

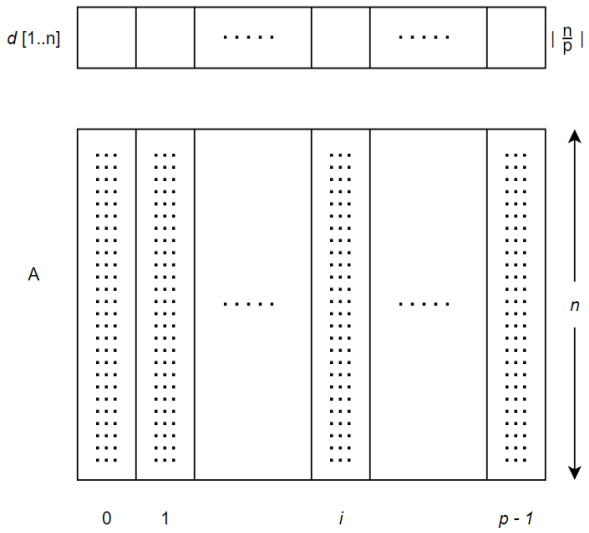


Рисунок 3.3 – Зважене неорієнтоване представлення графа з використанням списку суміжності

Під час кожної ітерації циклу кожен процес p_i обчислює $d_i[u]$, знаходячи мінімальне значення серед $d_i[v]$. Глобальний мінімум обчислюється з усіх $d_i[u]$ за допомогою операції зменшення «все до одного» та зберігається в процесі p_0. Нова вершина u тепер утримується процесом p_0 і буде вставлена в набір S. Потім процес p_0 передає нове значення u усім іншим процесам за допомогою трансляції один до

всіх, а процес, відповідальний за вершину u, p_i, позначає її як член набору S. Нарешті, кожен процес оновлює значення $d[v]$ для своїх локальних вершин.

Коли нова вершина u додається до набору S, значення $d[v]$ для вершин v повинні бути оновлені. Процес, відповідальний за v, повинен знати про вагу ребра, що з'єднує u і v. Таким чином, кожен процес p_i повинен зберігати стовпці зваженої матриці суміжності, які відповідають набору S вершин, призначених йому, що виконується через блокове відображення матриці. Простір, необхідний у кожному процесі для зберігання необхідної частини матриці суміжності, становить $\theta(\frac{n^2}{p})$.

Складність обчислення, яке виконується процесом для мінімізації та оновлення значень $d[v]$ під час кожної ітерації, становить $\theta(\frac{n}{p})$. Зв'язок між операціями, що виконується в кожній ітерації, відбувається завдяки операції "всі до одного" та оператору трансляції "один до всіх". Для паралельного комп'ютера з передачею повідомлень p-процесу трансляція «один до всіх» до одного слова потребує часу $log\ p$, де p – час, який займає виконання одного процесу. Час паралельного виконання алгоритму визначається як $T_p = \theta\left(\frac{n^2}{p}\right) + \theta(n\ log\ log\ p\ )$.

3.4 Генерування тестових даних

Як згадувалося раніше, існує два основних способи роботи з графами в пам'яті – список-масив як список суміжності та матриця суміжності. У рамках тестування запропонованого алгоритму згенеровані тестові дані будуть записані в 4 різні файли з відповідними розмірами графів - 100, 250, 500, 1000 і 1500 вершин. Після цього тестові дані будуть передаватися до реалізації алгоритмі на мові C++ на у звичайній та паралельних реалізаціях. Граф, створений і збережений у файлі, використовуватиме матрицю суміжності через особливості обраного підходу розпаралелювання, використання матриці суміжності як у пам'яті, так і у файлі

дозволяє нам використовувати підхід розпаралелювання на основі блоків. Використовуючи матрицю, ми зможемо надати кожному з під процесів у наших програмах власну підмножину вершин для обходу.

Також використовуючи той факт, що матриця суміжності є симетричною відносно головної діагоналі для незважених графів, ми можемо створити лише одну половину графа для збереження у файлі, тому що в одно направленому графі вершина [3;4] є такою самою, як вершина [4;3] .

Головна діагональ матриці суміжності буде представлена як 0, тому що ми не будемо генерувати жодних циклів у межах графа, а 0 буде індикатором того, що це координати в матриці суміжності до тих самих початкового та кінцевого вузлів.

Незв’язані вузли будуть позначені спеціальним символом «__», для зручності читання даних із файлу для позначення різних вершин використовується спеціальний роздільник «|». Оскільки ми працюємо лише з вагами від 10 до 99, ми могли б пропустити використання роздільників, але їх використання дає нам змогу легше зрозуміти згенеровані дані.

Використовуючи цю інформацію, ми пишемо невелику програму на C#, яка генерує файли .txt із графом, кількість вузлів якого було задано як параметр. Отримані файли називаються за кількістю вузлів у цьому графі.

Згенерований файл для графа розміром 64 важить близько 7 Кб, тоді як файл для графа з 1000 вузлів важить 1467 Кб і 3299 Кб для графа з 1500 вузлів. Це показує нелінійне зростання розміру, оскільки граф із 1500 вузлами має не лише в 1,5 рази більше вершин, ніж 1000 вузлів, але навіть у 1,5x1,5 рази більше, що дорівнює зростанню у 2,25 рази.

Ці тестові дані графу будуть використовуватися для порівняння продуктивності та скорості роботи послідовної реалізацій Дейкстри з її запропонованою паралельною реалізацією, що дасть можливість точно виявити, та оцінити можливі переваги однієї імплементації над іншою.

3.5 Тестування алгоритму

Як було зазначено у минулому пункті для тестування використовуватиметься послідовна та запропонована паралельна реалізація алгоритму Дейкстри на мові C++, з використанням бібліотеки OpenMP, яка дозволяє писати багато потокові програми та використовувати спеціально блоки для синхронізації даних.

Кожен тестовий запуск для кожного з обраних розмірів графів буде потворено не менше ніж 5 разів для послідовної та паралельної імплементації алгоритму, щоб отримати середній час, щоб переконатися, що ми отримуємо узгоджені результати, і виключити, що якийсь інший процес ОС міг вплинути на наші результати.

Потім ми порівняємо результати між собою, щоб переконатися, що найкоротші шляхи правильні у двох реалізаціях, і порівняємо середній час, який було витрачено на виконання алгоритму Дейкстри між послідовною та паралельною імплементаціями, щоб зробити висновок, щодо кореляція між розміром графу та приросту часу виконання між послідовною та паралельними імплементаціями.

У таблиці 3.1 показано час виконання пошуку найкоротших шляхів від початкового вузла 0 до всіх інших вузлів у межах графа для графів різного розміру з використанням послідовного алгоритму, які запускалися 5 разів, а також середній час виконання цих запусків. У таблиці 3.2 наведено середній час виконання для паралельної імплементації алгоритму.

Таблиця 3.1 – Виконання послідовного алгоритму

	Граф з 100 вершинами	Граф з 250 вершинами	Граф з 500 вершинами	Граф з 1000 вершинами	Граф з 1500 вершинами
Перший	2.13 мс	2.36 мс	2.58 мс	4.49 мс	6.9 мс
Другий	2.31 мс	2.43 мс	2.94 мс	4.41 мс	6.95 мс

Кінець таблиці 3.1

Третій	2.32 мс	2.41 мс	2.87 мс	4.48 мс	6.86 мс
Четвертий	2.32 мс	2.4 мс	2.89 мс	4.52 мс	6.94 мс
П'ятий	2.49 мс	2.9 мс	2.81 мс	4.44 мс	6.88 мс
Середній	2.314 мс	2,5 мс	2,818 мс	4.468 мс	6.906 мс

Таблиця 3.2 – Виконання паралельного алгоритму

	Граф з 100 вершинами	Граф з 250 вершинами	Граф з 500 вершинами	Граф з 1000 вершинами	Граф з 1500 вершинами
Перший	0.94мс	1.04 мс	1.87 мс	3.19 мс	4.93 мс
Другий	0.71мс	1.07 мс	1.63 мс	3.05 мс	5.02 мс
Третій	0.67мс	1.08 мс	1.73 мс	3.09 мс	4.92 мс
Четвертий	0.67мс	1.14 мс	1.68 мс	3.11 мс	4.78 мс
П'ятий	0.68мс	1.02 мс	1.75 мс	3.03 мс	4.71 мс
Середній	0.734мс	1.07 мс	1.732 мс	3.094 мс	4.872 мс

Як можливо побачити з отриманих даних паралельна реалізація алгоритму має відчутній приріст у виконанні у порівнянні з послідовною реалізацією. Так наприклад виконання для графу зі 100 вершинами демонструє приріст майже у 3 рази, в цей же час виконання для графу з 1500 вершинами складає 4.8 мс проти 6.9 мс при послідовному варіанту. Все це говорить про досить швидкий час виконання алгоритму, навіть при послідовному варіанті, а при використанні запропонованого паралельного алгоритму, можливо отримати досить суттєвій приріст, від 3х до 1.5 разів.

Отже, можливо зробити висновок, що використання паралельного алгоритму буде дуже доцільним для створення додатку для побудову енергоефективного маршруту для дронів.

4 ПРОГРАМНА РЕАЛИЗАЦІЯ ПРОТОТИПУ

4.1 Проектування прототипу

Запропонований варіант алгоритму дозволяє швидко знаходити найкоротший шлях у графі від однієї вершини до іншої, завдяки його багато потоковій природи його використання можливо майже на будь-якому сучасному пристрої – телефоні, комп’ютері та сервер.

Для вирішення задачі знаходження енергоефективного маршруту у складних умовах бойових дій актуальним є створення прототипу, який використовуватиме реализацію алгоритму, та який можливо буде використовувати на переносних девайсах, таких як ноутбуки або планшети. Тому для програмної реалізації прототипу було обрано створення десктопного додатку для комп’ьютерів, які працюють з системою Windows – WinForms, на платформі .NET Core, з мовою програмування C#. Для реалізації алгоритму пошуку маршруту було створенно DLL за допомогою мови програмування C++, до якої входить набір функцій які можливо визивати з різних місць, таких як проект на WinForms, додатків на мові C++ та інших, що дозволяє у разі необхідності швидко застосувати створену реалізацію у різних додатках, без необхідності змінювати цю реалізацію під конкретні випадки.

Створений прототип матиме можливість завантажувати у себе поточну мапу місцевості, далі користувач матиме можливість ввести точки на мапі, до яких він хоче отримати значення витрати електроенергії, яку витратять дрони для її досягнення. Особливості використання програми також мають давати користувачам отримувати точні значення витрат електроенергії, навіть у різних погодних умовах, тому прототип матиме можливість також враховувати максимальну швидкість дрону та поточку швидкість, та напряму вітру, для врахування під час обчислень витрат електроенергії. Користувач також матиме можливість вказувати яку з введених точок обирати як початкову та масштаб обраної мапи.

4.2 Врахування впливу вітру на енерговитрати дрону

Для правильного урахування як вітер впливає на дрон та його витрати, потрібно врахувати, яка швидкість вітру є максимальною, за якою дрон ще може безпечно літати. Деякі джерела [17] наводять значення, що максимальною швидкістю вітру для безпечного польоту є значення у $\frac{2}{3}$ від максимальної швидкості дрону. Будемо вважати, що якщо швидкість вітру перевищує це значення, то дрон не може летіти цим маршрутом.

Для обчислення впливу напрямку вітру на маршрут дрону потрібно спочатку розрахувати під яким кутом відносно напрямку польоту дрону буде дути вітер. Для цього є можливість використовуючи значення напряму вітру, знаючи кут відхилу напряму вітру від умовних осей ординат, так як напрям вітру буде вважатися лінійною функцією, яка задається формулою $y = kx + b$, де k – кут відхилу функції від осей ординат. Так як значення напряму вітру буде вводитись юзером, використовуючи тангенс цього куту, ми маємо можливість отримати коефіцієнт k.

Наступним кроком буде отримання значення коефіцієнту k, для лінійної функції напряму польоту дрону, так як дрон буде пересуватися між точками прямою лінією, тому можливо вважати формулу цього маршруту лінійною функцією. Знаючи координати точки від якої дрон буде здійснювати політ, до наступної точки, можливо отримати значення коефіцієнту за формулою $k = \frac{y_2 - y_1}{x_2 - x_1}$.

Знаючи значення двох коефіцієнтів k для двох лінійних функцій, можливо використовуючи формулу $\tan\theta = \left|\frac{k_2 - k_1}{1 + k_2 * k_1}\right|$, де θ – кут між двома лінійними функціями. Використовуючи зворотню функцію від тангенсу, в нас є можливість отримати фактичне значення куту, який потім можливо використати, для обчислення впливу вітру на поточний маршрут дрону.

4.3 Побудова графу

Так як у користувача буде можливість обирати на мапі деяку кількість точок, до яких має бути розраховано енерговитрати дрону – загальні кількість цих точок була обмежена кількість 100 штук. Через можливість користувача вводити різний напрям вітру, то потрібно враховувати ситуацію, що маршрут з однієї точки до іншої, за назад – це різни маршруту з різною кількістю енерговитрат, через можливість того, що вітер може бути попутний, або дути у спину.

Враховуючи це, граф який буде побудовано всередині прототипу буде – звішеним та направленим. Але через особливості алгоритму Дейкстри його реалізація як для направлених, так не для направлених графів не відрізняється, тому робити якісь зміни у алгоритм не потрібно.

Для коректної побудови енерговитрат від однієї точки до іншої з урахуванням напряму вітру, потрібно з кожної точки графи побудувати грані до кожної іншої точки, окрім цієї ж самої, через те що маршрут від точки до цієї ж точки не має сенсу. Враховуючи ці дані можливо розрахувати що для побудови мапи енерговитрат для польоту до точок, кількість яких складатиме 40, потрібно побудувати граф, в якому буде 1600 граней, і з врахування попередньо проведеного експерименту має давати приблизне значення у 4.872мс для обробки цього графу, що є дуже коротким часом і може використовуватися в умовах бойових дій.

4.4 Програмна реалізація

Як було попередньо зазначення додаток скрадатиметься з DLL написаної мовою програмування C++, яка скрадатиметься з функцій, які були створені під час проведення експерименту, та десктопного додатку, написаного з технологією WinForms, .NET Core та мовою програмування C#. Створений прототип має

можливість обирати мапу місцевості для завантаження у нього, після чого користувач має можливість створити точки на зображенні місцевості, до яких він хотів би отримати значення енерговитрат, швидкість вітру у метрах на секунду, напрям вітру у градусах, енерговитрати дрона на метр у ватах, максимальну швидкість дрону у метрах на секунду, обирати з якої точки на мапі починати розрахунки, та масштабування завантаженого зображення у пікселях на кілометр (рис 4.1).

Рисунок 4.1 – Головна форма додатку.

Натиснувши на кнопку Add Map у низу головної форми додатку, користувач має можливість обрати зображення місцевості в майже усіх можливих форматах зображень – JPEG, PNG, GIF та BMP.

Після того як користувач обирає зображення замість іконки мапи, яку зображено на рисунку 4.1 на головній формі з’явиться обране зображення (рис. 4.2),

а у користувача з’явиться можливість натискаючи на різних місцях обраного зображення створювати точки.

Рисунок 4.2 – Головна форма після обирання зображення.

Створені точки на мапі, будуть потім використовуватися для обраховування енерговитрат для їх досягнення, але для точного обрахування у поточних умов на фронті, є можливість справа від зображення ввести значення швидкості вітру, та якого напряму. Після введення даних для вітру справа внизу від зображення чорна стрілка буде показувати відносне значення напряму вітру, що дозволить знаючи орієнтацію зображення легше ввести правильні значення.

Після додавання значень для вітру важливо заповнити значення характеристик дрону, бо завдяки значень його максимальної швидкості та енерговитрат буде розраховуватися точки до яких можливо добратися у поточних погодних умовах, то розраховувати наскільки сильно вітер буде впливати на тий чи інший маршрут.

Після введення усіх коректних даних для дрону, та введення значення масштабу завантаженого зображення у користувача з’явиться можливість натиснути кнопку «Calculate Path» (рис. 4.3), яка призведе до розрахування енерговитрат.

Рисунок 4.3 – Головна форма після ведення усіх значень.

Отриманий маршрут для наступних значень:

– швидкість вітру – 5 М/с;
– напрям вітру – -122 градуси;
– витрати дрону – 10 Вт/м;
– максимальна швидкість дрону – 15 М/с;
– початкова точка – 1;
– масштаб мапи – 234 пікселі на кілометр.

На рисунку 4.4 можливо побачити що для досягнення деяких точок, таких як точка 16 та точка 15 потрібно різна кількість витраченої електроенергії, майже у 2

рази більша для точки 15 ніж 16, що обумовлено напрямом вітру, який впливає на дрон при його досягненні точки 15 більше ніж для точки 16, через різні кути напряму руху дрону відносно вітру для цих двох точок.

Рисунок 4.4 – Значення енерговитрат.

Порахувавши загальні витрати енергії для того, щоб дрон міг дістатися усіх точок можливо отримати значення у 52488 ват, що є досить великим значенням, і знаючи енергоємність акумуляторів або баків дрону, можливо прибрати не такі цікаві точки для обходження заради економії енергії.

ВИСНОВКИ

Під час виконання науково-дослідницької роботи було проаналізовано та досліджено існуючи додатки для побудови маршрутів для дронів, а також застосування дронів під час Російсько-Української війни. Було з’ясовано як саме використовуються дрони під час війни, та якими дронами користуються ти чи інші бригади та для яких задач.

Після цього було проведено аналіз існуючих методів для обходження графу для побудови маршрутів, та який з алгоритмів буде найточніше підходити під поставлену задачу побудови енергоефективних маршрутів для дронів. Алгоритмом для подальшої роботи був обраний алгоритм Дейкстри.

Був проведений аналіз алгоритму Дейкстри з урахуванням його особливостей, та запропоновано метод паралельної оптимізації цього алгоритму за допомогою мови програмування C++ та бібліотеки OpenMP. Було проведено експеримент та порівняння паралельної реалізації цього алгоритму з її послідовної варіацією, були отримані результати, які показують що використання запропонованої паралельної імплементації є перспективним, та що вона може бути використана для побудови маршрутів для дронів.

Для демонстрації актуальності та перспектив використання цього алгоритму було створено програмну систему прототип, за допомогою мови програмування C++, для створення DLL з реалізацією алгоритму, WinForms, .NET Core та C# для реалізації десктопної частини прототипу, яка дає можливість користувачу отримувати значення енерговитрат для досягнення дроном для точок на мапі, з можливістю корегувати розрахунки в залежності від напряму та швидкості вітру. Цей прототип показав можливість створення додатку для використання як для компаній, які використовують дрону у комерційних цілях – доставка вантажів, аналіз місцевості та інше, так і у військових цілях – для розуміння можливостей дрону у поточних умовах, що дає можливість точніше робити розрахунки та планувати наступальні операції.

Враховуючи створений прототип та запропонований алгоритм, можливо сказати, що в рамках науково-дослідницької роботи було представило паралельну реалізацію відносно послідовного алгоритму Дейкстри, та що експерименти з цим алгоритмом показали нам деякі особливості використання його з сучасними мовами програмування, але залишили багато місця для вдосконалення.

Так створений прототип має у собі дуже спростовану методику розрахування впливу напрямку вітру на те як поводить себе дрон, так наприклад не враховані можливість того, що під час перельоту до наступної точки швидкість вітру може змінитися, що може сильно вплинути на загальні витрати дрону. Також не враховується масу дрону, що також впливає на максимальну швидкість вітру, під час якої цей дрон може здійснювати безпечний політ, так наприклад дрони з однаковою максимальною швидкістю, але різною масою можуть витримувати різні максимальні швидкості вітру, що у створеному прототипі не було враховано, а було враховано лише вказану максимальну швидкість дрону.

Алгоритм Дейкстри є дуже послідовним, і роботи з його подальшим розпаралелюванням обмежені через його природу. Для подальших удосконалень цього алгоритму пропонується або використовувати пошук, який починається з двох різних вузлів і виконується паралельно, або спробувати використати інших доступний алгоритм для пошуку шляху у графі, наприклад A*. Оптимізувати запропонований алгоритм паралельної імплементації Дейкстри та A* можна також спробою використовувати інші методи збереження графі у пам’яті через те що матриця суміжності не є найефективнішим методом для цього, що призводить до того, що вона займає багато пам'яті як в пам'яті, так і на диску. Але для найбільшої економії ресурсів пам’яті краще використовувати список суміжності, який займає менше місця.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. X. Bai, L. Wang, Y. Hu, P. Li and Y. Zu, "Optimal Path Planning Method for IMU System-Level Calibration Based on Improved Dijkstra’s Algorithm," in IEEE Access, vol. 11, pp. 11364-11376, 2023, doi: 10.1109/ACCESS.2023.3240518.
2. M. C. Brindise, B. A. Meyers, S. Kutty and P. P. Vlachos, "Automated Peak Prominence-Based Iterative Dijkstra's Algorithm for Segmentation of B-Mode Echocardiograms," in IEEE Transactions on Biomedical Engineering, vol. 69, no. 5, pp. 1595-1607, May 2022, doi: 10.1109/TBME.2021.3123612.
3. Q. Jin, W. Sheng, C. Gao and Y. Han, "A Cable Harness Routing Method Based on the Expected Maximum Crosstalk," in IEEE Transactions on Electromagnetic Compatibility, vol. 64, no. 3, pp. 856-864, June 2022, doi: 10.1109/TEMC.2021.3131492.
4. A. Buzachis, A. Celesti, A. Galletta, J. Wan and M. Fazio, "Evaluating an Application Aware Distributed Dijkstra Shortest Path Algorithm in Hybrid Cloud/Edge Environments," in IEEE Transactions on Sustainable Computing, vol. 7, no. 2, pp. 289-298, 1 April-June 2022, doi: 10.1109/TSUSC.2021.3071476.
5. D. Lopez-Pajares, E. Rojas, J. A. Carral, I. Martinez-Yelmo and J. Alvarez-Horcajo, "The Disjoint Multipath Challenge: Multiple Disjoint Paths Guaranteeing Scalability," in IEEE Access, vol. 9, pp. 74422-74436, 2021, doi: 10.1109/ACCESS.2021.3080931.
6. S. Ragothaman, M. Maaref and Z. M. Kassas, "Autonomous Ground Vehicle Path Planning in Urban Environments Using GNSS and Cellular Signals Reliability Maps: Models and Algorithms," in IEEE Transactions on Aerospace and Electronic Systems, vol. 57, no. 3, pp. 1562-1580, June 2021, doi: 10.1109/TAES.2021.3054690.
7. Y. Yu, O. S. Nduka and B. C. Pal, "Smart Control of an Electric Vehicle for Ancillary Service in DC Microgrid," in IEEE Access, vol. 8, pp. 197222-197235, 2020, doi: 10.1109/ACCESS.2020.3034496.

8. Y. Meng, Y. Wu, Q. Gu and L. Liu, "A Decoupled Trajectory Planning Framework Based on the Integration of Lattice Searching and Convex Optimization," in IEEE Access, vol. 7, pp. 130530-130551, 2019, doi: 10.1109/ACCESS.2019.2940271.
9. J. Han, D. Shen, D. Karbowski and A. Rousseau, "Leveraging Multiple Connected Traffic Light Signals in an Energy-Efficient Speed Planner," in IEEE Control Systems Letters, vol. 5, no. 6, pp. 2078-2083, Dec. 2021, doi: 10.1109/LCSYS.2020.3047605.
10. B. Confais, B. Parrein and A. Lebre, "Data Location Management Protocol for Object Stores in a Fog Computing Infrastructure," in IEEE Transactions on Network and Service Management, vol. 16, no. 4, pp. 1624-1637, Dec. 2019, doi: 10.1109/TNSM.2019.2929823.
11. F. Zhang, X. Zhou and M. Sun, "On-Demand Receiver-Centric Channel Allocation via Constrained VCG Auction for Spatial Spectrum Reuse," in IEEE Systems Journal, vol. 13, no. 3, pp. 2519-2530, Sept. 2019, doi: 10.1109/JSYST.2019.2912757.
12. T. Kulvicius, S. Herzog, M. Tamosiunaite and F. Wörgötter, "Finding Optimal Paths Using Networks Without Learning—Unifying Classical Approaches," in IEEE Transactions on Neural Networks and Learning Systems, vol. 33, no. 12, pp. 7877-7887, Dec. 2022, doi: 10.1109/TNNLS.2021.3089023.
13. T. Parsons, J. Seo and D. Livesey, "Waste Collection Area Generation Using a 2 Stage Cluster Optimization Process and GIS Data," in IEEE Access, vol. 11, pp. 11849-11859, 2023, doi: 10.1109/ACCESS.2023.3241626.
14. K. Smelyakov, A. Chupryna, M. Hvozdiev and D. Sandrkin, "Gradational Correction Models Efficiency Analysis of Low-Light Digital Image," 2019 Open Conference of Electrical, Electronic and Information Sciences (eStream), 2019, pp. 1-6, doi: 10.1109/eStream.2019.8732174.
15. K. Smelyakov, A. Datsenko, V. Skrypka and A. Akhundov, "The Efficiency of Images Reduction Algorithms with Small-Sized and Linear Details," 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications,

Science and Technology (PIC S&T), 2019, pp. 745-750, doi: 10.1109/PICST47496.2019.9061250.

16. K. Smelyakov, A. Chupryna, O. Bohomolov and N. Hunko, "The Neural Network Models Effectiveness for Face Detection and Face Recognition," 2021 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream), 2021, pp. 1-7, doi: 10.1109/eStream53087.2021.9431476.
17. Can Drones Fly in Strong Winds? (Drone Wind Resistance Comparison) [Електронний ресурс] – Режим доступу до ресурсу: https://dronesgator.com/can-drones-fly-in-strong-winds/ (дата звернення: 18.04.2023).