НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО

Факультет інформатики та обчислювальної техніки
(назва факультету, інституту)

Кафедра автоматизованих систем обробки інформації і управління
(назва кафедри)

"На правах рукопису"
УДК *004.942*

«До захисту допущено»
В.о.завідувача кафедри

_____ О.А.Павлов
(підпис) (ініціали, прізвище)

“ ” 20 *19* р.

МАГІСТЕРСЬКА ДИСЕРТАЦІЯ

на здобуття ступеня магістра

за спеціальністю *126 Інформаційні системи та технології*
(код та назва спеціальності)

ОПП *Інформаційні управляючі системи та технології*
(код та назва спеціалізації)

на тему: *Оптимізація маршрутів групи безпілотних авіаційних систем*

Виконав: студент *VI* курсу групи ІС-82мп
(шифр групи)

Сторчевий Владислав Володимирович _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник *проф., д.т.н., с.н.с. Гуляницький Л.Ф.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант *доц.,к.т.н. Жданова О.Г.* _____
(науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент *д.ф.-м.н., с.н.с Стецюк П.І.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. ІГОРЯ СІКОРСЬКОГО»

Факультет *інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації та управління*
(повна назва)

Рівень вищої освіти *другий (магістерський) за освітньо-професійною програмою*

Спеціальність *126 Інформаційні системи та технології*
(код і назва)

ОПП *Інформаційні управляючі системи та технології*
(код і назва)

ЗАТВЕРДЖУЮ
В.о.завідувача кафедри
_____ О.А.Павлов
(підпис) (ініціали, прізвище)
«__» _____ 2019 р.

ЗАВДАННЯ

на магістерську дисертацію студенту
Сторчевому Владиславу Володимировичу
(прізвище, ім'я, по батькові)

1. Тема дисертації *Оптимізація маршрутів групи безпілотних авіаційних систем*

науковий керівник дисертації *Гуляницький Леонід Федорович, д.т.н., проф.*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від " *28* " *10* 20*19* р. № *3770-с*

2. Строк подання студентом дисертації " *2* " *12* 20*19* р.

3. Об'єкт дослідження *Функціонування групи БАС при виконанні поставлених завдань*

4. Перелік завдань, які потрібно розробити *1. Виконати формалізацію задачі планування маршруту групи БАС. 2. Розробити алгоритми комбінаторної оптимізації для планування маршруту. 3. Розробити підхід для налаштування параметрів розроблених алгоритмів. 4. Провести дослідження запропонованих алгоритмів шляхом проведення обчислювальних експериментів.*

5. Орієнтовний перелік ілюстративного матеріалу *1. Псевдокод алгоритму табуйованого пошуку. 2. Псевдокод макс-мін алгоритму мурашиних систем. 3. Блок-схема організації острівної моделі. 4. Псевдокод острівної моделі диверсифікованого макс-мін алгоритму мурашиних систем. 5. Псевдокод підходу до налаштування параметрів алгоритму. 6. Схема структурна класів програмного забезпечення. 7. Результати експериментів*

6. Орієнтовний перелік публікацій *Три публікації: дві статті у фахових журналах, тези доповіді на науковій конференції*

7. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

8. Дата видачі завдання "*2*" *вересня* 20*19* р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	*Систематизація результатів огляду літератури*	*17.09.2019*	
2	*Аналіз існуючих методів розв'язання задачі*	*01.10.2019*	
3	*Постановка та формалізація математичної моделі задачі*	*08.10.2019*	
4	*Розробка алгоритмів розв'язання задачі*	*15.10.2019*	
5	*Розробка інформаційного та програмного забезпечення*	*12.11.2019*	
7	*Проведення експериментальних досліджень розроблених алгоритмів*	*16.11.2019*	
8	*Оформлення документації*	*19.11.2019*	
9	*Подання роботи на попередній захист*	*20.11.2019*	
10	*Подання роботи на основний захист*	*02.12.2019*	

Студент _____ *В.В. Сторчевий*
(підпис) (ініціали, прізвище)

Науковий керівник _____ *Л.Ф. Гуляницький*
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Магістерська дисертація: 97 с., 17 рис., 35 табл., 1 додаток, 66 джерел.

Актуальність: Безпілотні авіаційні системи (БАС), зокрема безпілотні літальні апарати (БПЛА) набувають все більш широкого застосування. Широкий функціонал БАС і збільшення їх доступності, приводить до розширення сфери використання. Сьогодні БАС використовуються для аудиту земель, боротьби з браконьєрами, моніторингу і контролю сільськогосподарських угідь, за кордоном набуває популярності доставка товарів дронами. Особливо важливим є використання БАС у військовій сфері для проведення тактичної або стратегічної розвідки, оскільки дозволяє армії не ризикувати особовим складом.

Часто під час військової операції використовується група БАС. При використанні групи БАС важливо попередньо спланувати маршрут, оскільки противники навмисно можуть створювати перешкоди (приглушення радіозв'язку), що унеможливлюють керування у реальному часі.

У дисертації розглянуто планування маршруту групи БАС з використанням рухомих платформ для пуску БПЛА, це є особливо актуальним сьогодні, оскільки агентство передових оборонних дослідницьких проектів США (DARPA) запустило програму Gremlins, що передбачає запуск груп БПЛА з існуючих великих літальних апаратів, таких як бомбардувальники або транспортні літаки, а також з винищувачів, нерухомих платформ буксирування.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалась на філії кафедри автоматизованих систем обробки інформації та управління Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського» у рамках науково-дослідницької теми Інституту кібернетики ім. В. М. Глушкова НАН України ВФ.180.11 «Розробити математичний апарат, орієнтований на створення інтелектуальних інформаційних технологій розв'язування проблем комбінаторної оптимізації та інформаційної безпеки» (2017-2021 рр.), що виконується за Постановою бюро Відділення інформатики НАН України від 23.06.2016 р. № 2.

Мета дослідження – мінімізація затрат при виконанні поставлених завдань групою БАС, що діють як команда, шляхом оптимізації їх маршрутів.

Для досягнення цієї мети необхідно виконати наступні **завдання**:

- виконати огляд відомих задач маршрутизації БПЛА, а також алгоритмів комбінаторної оптимізації, які застосовуються для їх розв’язання;
- виконати формалізацію задачі планування маршруту групи БАС з використанням рухомих платформ для пуску БПЛА;
- розробити програмну реалізацію алгоритмів комбінаторної оптимізації для планування маршруту;
- провести дослідження запропонованих алгоритмів шляхом проведення обчислювальних експериментів.

Об’єкт дослідження – функціонування групи БАС при виконанні поставлених завдань.

Предмет дослідження – планування маршрутів груп БАС з можливістю використанням рухомих платформ для пуску та приземлення БАС, зокрема БПЛА.

Наукова новизна одержаних результатів полягає в розробці математичної моделі та алгоритмів розв'язування проблем оптимізації маршрутів групи БАС, а також реалізації розробленого математичного апарату у вигляді спеціалізованого програмного комплексу для розв’язування досліджуваних задач маршрутизації.

Публікації. Матеріали роботи опубліковані у збірнику «Комп’ютерна математика» інституту кібернетики імені В.М. Глушкова НАН України та у фаховому журналі «Науковий вісник Ужгородського університету. Серія математика і інформатика» Ужгородського національного університету та на III всеукраїнській науково-практичній конференції молодих вчених та студентів «Інформаційні системи та технології управління» (ІСТУ-2019).

АЛГОРИТМИ КОМБІНАТОРНОЇ ОПТИМІЗАЦІЇ, ЗАДАЧА МАРШРУТИЗАЦІЇ БЕЗПІЛОТНИХ АВІАЦІЙНИХ СИСТЕМ, МУЛЬТИДЕПО, ПАРАМЕТРИ АЛГОРИТМІВ, НАЛАШТУВАННЯ ПАРАМЕТРІВ, ЛОКАЛЬНИЙ ПОШУК, ОСТРІВНА МОДЕЛЬ

ABSTRACT

Master's thesis: 97 p., 17 figures, 35 tables, 1 application, 66 sources.

Relevance: Unmanned aerial vehicle systems (UAS), in particular unmanned aerial vehicles (UAV), are becoming more widely used. The wide functionality of the UAS and their increased availability leads to an expansion of the scope. Today UAS is used for land audits, poaching, monitoring and control of agricultural land, the delivery of drones is becoming more popular abroad. Particularly important is the use of UAS in the military sphere for tactical or strategic reconnaissance, as it allows the army not to risk personnel.

Often during the military operation, the UAS group is used. When using the UAS group, it is important to plan the route beforehand, as opponents may intentionally create interference (radio jamming) that makes it impossible to control in real time.

The dissertation examines the route planning of the UAS group using mobile UAV launch platforms, which is especially relevant today as the United States Defense Advanced Research Projects Agency (DARPA) launched the Gremlins program, which involves launching UAV teams from existing large aircraft such as bombers, transport planes, etc.

Connection of the thesis with scientific programs, plans, topics. The thesis was written at the branch of The Department of Computer-aided management and data processing systems of the National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute» at the V. M. Glushkov Institute of Cybernetics of the National Academy of Sciences of Ukraine under the topic VF.180.11 «To develop a mathematical apparatus focused on the creation of intelligent information technologies for solving combinatorial optimization and information security problems» (2017-2021 biennium), which is executed by the Resolution of the Bureau of Informatics of the National Academy of Sciences of Ukraine from 23.06.2016 p. № 2.

The purpose of the study - to minimize the costs of fulfilling the tasks set by the UAS group acting as a team by optimizing their routes.

To achieve this goal, you must complete the following tasks:

- review the known UAV routing tasks and the combinatorial optimization algorithms used to solve them;
- formalize the task of planning the route of the UAS group using mobile platforms for launching UAV;
- develop software implementation of combinatorial optimization algorithms for route planning;
- carry out research of the offered algorithms by carrying out computational experiments.

The object of study is the functioning of the UAS group when completing the tasks.

The subject of study is the planning of routes of UAS groups with the possibility of using mobile platforms for launching and landing UAS, in particular UAV.

The scientific novelty of the results is the development of a mathematical model and algorithms for solving the problems of optimization of routes of the UAS group, as well as the implementation of the developed mathematical apparatus in the form of a specialized software complex for solving the studied routing problems.

Publications. The materials are published in the compilation "Computer Mathematics" of the Institute of Cybernetics of Glushkov NAS of Ukraine, in the professional journal «Scientific Bulletin of Uzhgorod University. Mathematics and Informatics Series» of Uzhgorod National University and at the III All-Ukrainian Scientific and Practical Conference of Young Scientists and Students "Information Systems and Technologies of Management" (ISTM-2019).

ALGORITHMS OF COMBINATORY OPTIMIZATION, THE UNMANAGED AIRCRAFT SYSTEMS ROUTING PROBLEM, MULTI-DEPOT, PARAMETERS OF ALGORITHMS, LOCAL SEARCH, ISLAND MODEL

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

БАС – безпілотна авіаційна система

БПЛА – безпілотний літальний апарат

МВС – метод вектору спаду

VRP – vehicle routing problem

TSP – travelling salesman problem

DARPA – defense advanced research projects agency

MDVRP – multi-depot vehicle routing problem

MMAS – max-min ant system

ВСТУП

Безпілотні авіаційні системи (БАС), зокрема безпілотні літальні апарати (БПЛА) набувають все більш широкого застосування. Через широкий функціонал БАС і збільшення простоти використання дедалі більше компаній починають впроваджувати їх у свою діяльність: житлово-комунальні господарства – для енергоаудиту будинків, лісництва – для боротьби з незаконною вирубкою, охоронні підприємства – для виявлення порушників, у сільському господарстві - для зрошення, захисту від замерзання, моніторингу структури площ для посівів і контролі за використанням угідь, тощо. Однією з найважливіших сфер застосування групи БАС є військова сфера, де БПЛА використовуються для ведення повітряної розвідки – як тактичної, так і стратегічної. Ефективним варіантом їх застосування є вирішення завдань у складі групи.

Сьогодні важливою є проблема автоматизації групи БАС для подальшого спрощення дій операторів, які керують ними. Однією з задач є побудова маршруту між заданим набором точок. Побудова маршруту для групи БАС відбувається заздалегідь, оскільки під час польоту можуть бути різного роду перешкоди для радіозв’язку і оператор втратить контроль над літальним апаратом, що призведе до провалу місії, тому маршрут повинен бути чітко спланований і запрограмований в БАС.

В дисертації досліджується задача математичного моделювання маршрутів групи БАС, для розв’язування якої пропонується спеціальний алгоритм мурашиних колоній, оскільки ефективність мурашиних алгоритмів підтверджується досвідом розв'язування різних задач маршрутизації транспорту.

1 ОГЛЯД ІСНУЮЧИХ ДОСЛІДЖЕНЬ

Було проведено багато досліджень на тему планування маршрутів безпілотних літальних апаратів (БПЛА). У 2004 році було сформовано загальну модель задачі оптимізації маршрутів БПЛА [1]. У наступних дослідженнях вводилися нові обмеження: вперше розглянуто обмеження на вантаж у роботі [2], поставлена умова наявності часових вікон [3], уведено обмеження польотного ресурсу [4]. Також були розглянуті різні цільові функції: мінімізація загального часу місії [5], мінімізація сумарної довжини маршрутів [6], мінімізація кількості БПЛА [7], мінімізація польотного ресурсу [8].

Однією з найважливіших сфер застосування БПЛА є військова сфера, де БПЛА використовуються для ведення повітряної розвідки – як тактичної, так і стратегічної [9,10]. У [2] бойові літальні апарати застосовуються для знищення заздалегідь визначених цілей з умовою обмеження боєприпасів. Також у [11] представлений алгоритм розв'язання задачі збору інформації за мінімальний час з використанням групи безпілотних літальних апаратів. Задача планування маршруту з умовою уникнення загроз (радіолокаційне виявлення, перешкоди, зіткнення з іншими БПЛА) розглянуто у [12].

З кожним роком розширюється сфера застосування БПЛА, що призводить до активного створення нових постановок задач. У роботі Ines Khouf “A Survey of Recent Extended Variants of the Traveling Salesman and Vehicle Routing Problems for Unmanned Aerial Vehicles” [13] проведено детальний аналіз проблем оптимізації шляху БПЛА у рамках двох класів задач: задачі комівояжера (TSP) та задачі маршрутизації транспортних засобів (VRP). У статті розглядається задача кооперації БПЛА та вантажівок. Вантажні автомобілі слугують платформою для перевозки дронів з вантажем, а ті в свою чергу доставляють товари клієнтам. Для даної задачі було виділено п’ять можливих конфігурацій:

- кількість БПЛА = 1, використовується тільки один БПЛА;
- кількість БПЛА = m, розгорнутий парк дронів;

- кількість БПЛА = 1, кількість вантажівок = 1, один БПЛА працює в тандемі з однією вантажівкою;
- кількість БПЛА = m, кількість вантажівок = 1, парк БПЛА працює в тандемі з однією вантажівкою;
- кількість БПЛА = m, кількість вантажівок = n, парк БПЛА працює в тандемі з багатьма вантажними автомобілями.

Усі існуючі проблеми маршрутизації БПЛА, опубліковані протягом останніх кількох років, є варіантами задачі комівояжера або задачі маршрутизації транспортних засобів. У [13] наведена наступна класифікація.

Розширені варіанти задачі комівояжера

TSP-D - задача комівояжера з використанням БПЛА. Дано набір позицій, одне депо, один БПЛА і показник витрат. Метою TSP-D є визначення такого маршруту, за якого кожна ціль є відвідана лише один раз і вкінці БПЛА повертається до депо, при цьому загальна вартість маршруту (наприклад, пройдена відстань або час завершення) зводиться до мінімуму.

mTSP-D - це узагальнення TSP-D у якому використовується більше ніж один дрон. Завдання mTSP-D полягає у визначенні набору m маршрутів таким чином, що загальна вартість усіх маршрутів зведена до мінімуму, а кожну ціль відвідує лише один БПЛА.

ATSP-D - також є узагальненням TSP-D, відрізняється тим, що вартість між вершинами може бути різна в залежності від напряму.

CETSP-D - є узагальненням TSP-D, де кожна ціль ідентифікується за її сусідством. Мета – визначити маршрут БПЛА, який відвідує околиці кожної цілі лише один раз і повертається в депо, при цьому загальна вартість маршруту зведена до мінімуму.

Розширені варіанти задачі маршрутизації транспортних засобів

VRP-D - задача маршрутизації транспортних засобів з БПЛА. VRP-D має на меті визначити набір маршрутів для парку дронів, які відвідують набір цілей, при цьому щоб загальна вартість маршрутів була мінімізована.

VRPTW-D - це узагальнення VRP-D, де обслуговування кожного клієнта починається в заданий час, який називається часовим вікном. Вікно задається для кожного клієнта і визначається інтервалом.

CVRPTW-D - є узагальненням VRPTW-D, коли кожен БПЛА має певну ємність (наприклад, вантажопідйомність), а загальний попит, що обслуговується кожним дроном, не перевищує його потужність.

GVRP-D - це варіант VRP-D, у якому за мету поставлено зменшення кількості використовуваних транспортних засобів за рахунок дозаправки БПЛА.

MTVRP-D - це варіант GVRP-D, спрямований на подолання проблеми обмеженої вантажопідйомності, БПЛА можуть повернутися в депо і здійснювати багаторазові польоти.

БПЛА можуть використовуватися наступним чином:

- транспортування та доставка: доставка посилок, товарів, реклами, тощо;
- зв'язок: БПЛА можуть використовуватися як літаючі ретрансляційні пункти для відновлення зв'язку;
- спостереження, моніторинг, відстеження: БПЛА, обладнані датчиками, можна використовувати для моніторингу території;
- логістичні процеси: БПЛА можна використовувати для інвентаризації запасів на складах;
- катастрофи: БПЛА можуть використовуватися для ліквідації пожеж або для пошуку людей чи тварин.

Також, багато уваги приділено алгоритмам оптимізації маршруту.

Для розв'язання задач високої складності, таких як планування маршруту БПЛА, використовується окремий клас оптимізаційних методів розв'язання задач комбінаторної оптимізації - метаевристики. Вони покладаються на вдосконалення локальних розв’язків протягом кількох ітерацій для отримання квазіоптимальних розв’язків. Останніми прикладами застосування метаевристик, що використовуються для планування шляху БПЛА, є алгоритм імітаційного відпалу [14], генетичний алгоритм [15], метод диференціальної

еволюції [16], оптимізація потоком частинок [17], оптимізація мурашиними колоніями [18], бджолині алгоритми [19], алгоритм хижак-жертва [20], алгоритм сірого вовка [21] та багато інших. Кожен з цих підходів забезпечує унікальну стратегію модифікації та вдосконалення розв'язків, їх ефективність часто залежить від проблеми, яка розглядається.

Задача, що розглянута у даній дисертації, відрізняється від вищеописаних наступними аспектами:

- альтернативний вибір депо;
- використовуються рухомі платформи для пуску БПЛА;
- депо представляє собою обмежену ділянку, яка може прийняти певну кількість БПЛА.

Використання рухомих платформ для пуску БПЛА є особливо актуальним сьогодні, оскільки агентство передових оборонних дослідницьких проектів США (DARPA) запустило програму Gremlins. Програма передбачає запуск груп БПЛА з існуючих великих літальних апаратів, таких як бомбардувальники або транспортні літаки, а також з винищувачів, нерухомих платформ-буксирування. Коли гремліни завершать свою місію, транспортний літак поверне їх у повітря та віднесе додому, де наземні екіпажі підготують їх до наступного використання. Гремліни покликані не замінити поточні багатоцільові літаки, а доповнити їх більш простими і дешевими системами для спеціалізованих завдань [22].

Висновки до розділу

Задача планування маршрутів безпілотних літальних апаратів є особливо актуальною сьогодні. Особливий інтерес до неї проявляється у військовому секторі різних країн. Створюються різні наукові програми, які шукають нові методи і розробки способів запуску і посадки дронів, їх інтеграції з літаком, автоматизованої заправки і безпілотної посадки. Також увага приділяється створенню маршрутів для групи БПЛА, оскільки керування дронами у реальному часі може бути ускладнене різними перешкодами, наприклад, блокування радіозв'язку. Отже, весь маршрут операції повинен бути визначений заздалегідь і БПЛА в автоматичному режимі здійснюватимуть політ.

Усі існуючі проблеми маршрутизації БПЛА, опубліковані протягом останніх кількох років, є варіантами задачі комівояжера або задачі маршрутизації транспортних засобів.

Тому для розв'язку задачі маршрутизації безпілотних авіаційних систем використовуються ті ж методи, що цих двох класів задач, а саме генетичний алгоритм, оптимізація потоком частинок, оптимізація мурашиними колоніями, бджолині алгоритми, тощо.

Задача, що розглядається у даній дисертації, є актуальною, оскільки враховує потреби використання рухомих платформи для пуску БПЛА, особливу увагу до яких приділяє DARPA.

2 ЗАДАЧА МАРШРУТИЗАЦІЇ БЕЗПІЛОТНИХ АВІАЦІЙНИХ СИСТЕМ, АЛГОРИТМИ ЇЇ РОЗВ'ЯЗАННЯ ТА ЇХ ОБЧИСЛЮВАЛЬНА СХЕМА

2.1 Задача маршрутизації безпілотних авіаційних систем

Задачу маршрутизації БПЛА можна звести до задачі маршрутизації транспортних засобів (VRP), яка вперше була представлена в літературі у 1959 році Данцігом і Рамсером [23]. Задачі VRP лежать на перетині двох добре відомих задач. Це задача комівояжера (якщо вантажопідйомність кожного транспортного засобу вважається достатньою, то VRP зводиться до набору задач комівояжера) та задача про ранець (розв'язок даної задачі, по суті, еквівалентний розв'язанню задачі VRP за умови, що всі відстані приймаються рівними нулю). Класична задача маршрутизації транспортних засобів може бути представлена наступним чином: маємо граф $G(E,V)$, де $V = \{0,1,\ldots,n\}$ – множина вершин (депо, що має індекс $j = 0$; споживачі, що мають індекс $j = 1,2,\ldots,n$), а E – множина ребер. Кожен споживач має попит $d_j > 0$. Кожне ребро являє собою маршрут від вузла i до вузла j. Вага кожної дуги $C_{ij} > 0$ відповідає вартості (часу або навіть відстані) переходу від вузла i до вузла j. Ціль - мінімізація довжини маршруту. Існує два варіанти: задача є симетричною, якщо $C_{ij} = C_{ji}$, та асиметричною у іншому випадку. Існує багато варіацій задач VRP, які утворюються за допомогою введення нових обмежень або комбінації старих. Також можливе комбінування декількох критеріїв, що значно ускладнює задачу. Одними з найбільш поширених є обмеження на вантажопідйомність (CVRP) [24] та врахування часових вікон VRP [25]. На рисунку 2.1 відображені основні класи задач VRP [26,27].

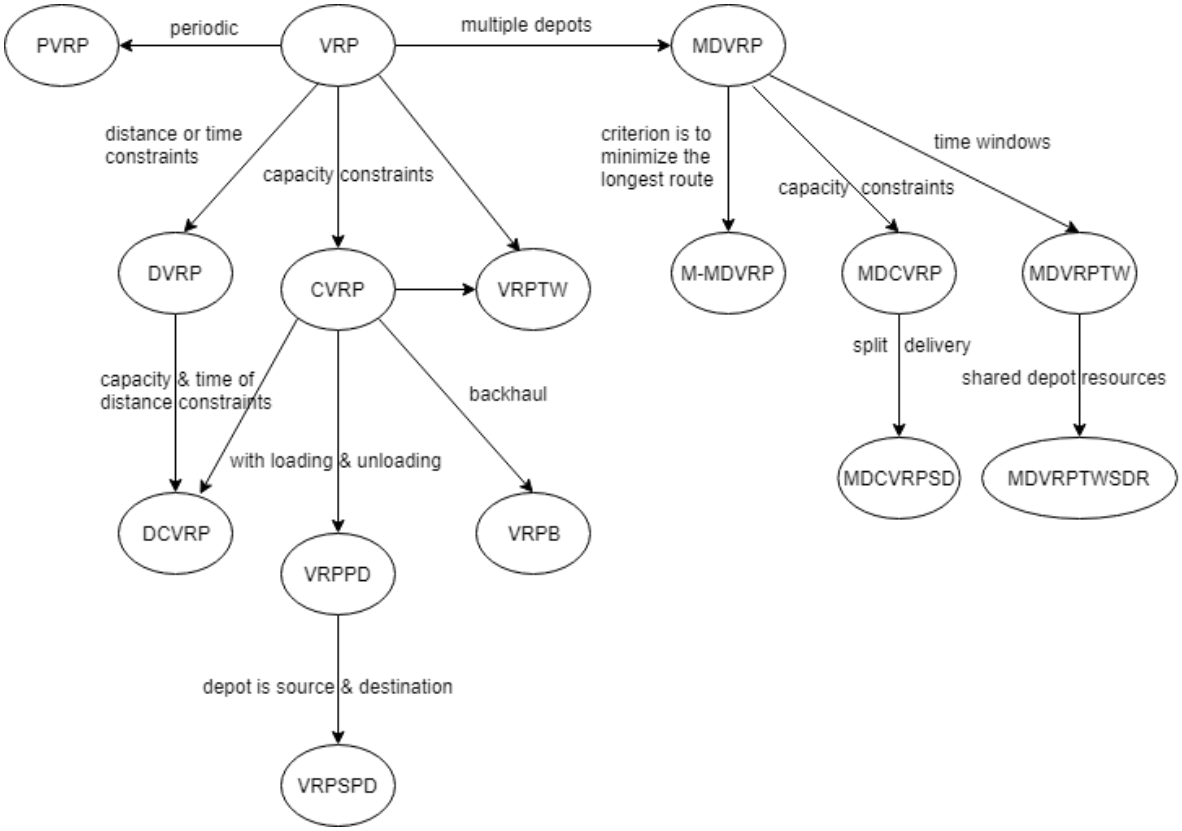


Рисунок 2.1 - Основні класи задач VRP

Задача маршрутизації БПЛА у військовій сфері часто відноситься до класу multi depot vehicle routing problem (MDVRP). MDVRP може бути представлена наступним чином: маємо граф $G(V,E)$, де $V = \{0,1,\ldots,n\}$ – множина вершин (депо, споживачі), а E – множина ребер. Кожне ребро являє собою маршрут від вузла i до вузла j. Множина V розбита на дві підмножини: $V_c = \{ v_1, v_2,\ldots, v_N \}$, яка містить споживачів та множину $V_d = \{ v_{N+1}, v_{N+2},\ldots, v_M \}$, яка містить депо. Кожен споживач $v_i \in V_c$ має невід'ємний попит d_i. Кожне ребро з E має вартість, відстань або час в дорозі c_{ij}. Маємо K транспортних засобів, кожен з яких має ємкість P_k. Задача полягає в побудові маршруту таким чином, щоб: кожен маршрут починався і закінчувався в одному і тому ж депо; кожен клієнт обслуговувався лише один раз; загальний попит на маршруті не перевищував ємкість транспортного засобу; загальна довжина маршруту була мінімальною [26]. На рисунку 2.2 зображено відмінність MDVRP від VRP [27].

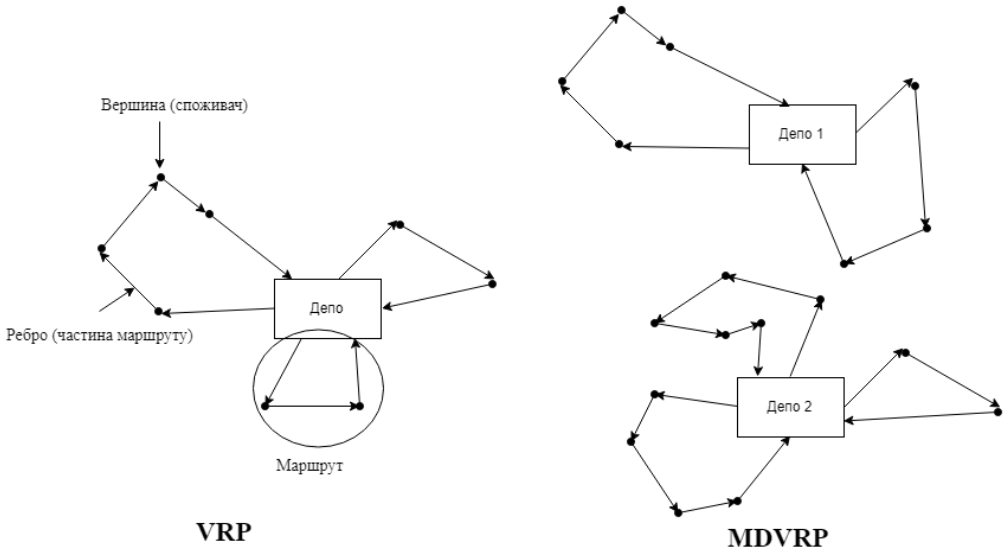


Рисунок 2.2 - Порівняння VRP та MDVRP

2.2 Постановка задачі пошуку маршрутів

Розглядається проблема пошуку оптимальних маршрутів для групи спеціальних літальних апаратів, в першу чергу, БПЛА. Вона полягає у тому, що перед заданою групою БПЛА, які можуть стартувати з різних точок пуску та мати можливість закінчувати маршрут в різних місцях (депо), стоїть завдання облетіти низку заданих об’єктів (точок на місцевості) з мінімізацією сумарної довжини маршрутів чи тривалості польотів за умов, що кожен об’єкт відвідується одним і лише одним БПЛА і всі об’єкти повинні бути відвіданими. При цьому часто слід враховувати ще й ряд додаткових обмежуючих умов (дальність польоту без підзаряджання чи дозаправлення, вантажопідйомність, погодні умови, тощо). Отже, приземлення БПЛА може здійснюватися в декількох заданих територіально зонах. Припускається також, що характеристики кожної зони вибираються так, що у конкретній зоні можливе приймання заданої кількості БПЛА: у такій зоні можуть завершувати свій маршрут всі чи частина БПЛА. Кожну зону будемо ідентифікувати її центроїдою. Отже, маршрут кожного із БПЛА повинен закінчуватися в одній із таких зон, але конкретна зона для конкретного БПЛА не вказується. Цю проблему назвемо задачею маршрутизації з альтернативними депо. Кількість міст старту БПЛА може перевищувати кількість самих БПЛА – це створює додаткові можливості для вибору місць (чи моментів) старту. Також відмітимо,

що в процесі розв'язування задачі маршрутизації може оптимізуватися також і кількість задіяних БПЛА.

Задача. За умови наявності декількох можливих зон приймання БПЛА і місць старту БПЛА, а також завершення маршрутів БПЛА лише в одній із виділених зон слід визначити як маршрути БПЛА з оптимізацією сумарної довжини (тривалості польотів) та їх задіяної кількості, так і місця (зони) їх приймання. Зауважимо, що в одній зоні може завершуватися декілька маршрутів БПЛА.

Змістовна постановка задачі маршрутизації груп БПЛА для пошуку об’єктів може бути подана так. Задано координати місць старту, кількість БПЛА, координати точок, які необхідно відвідати, координати зон приймання, польотний ресурс кожного БПЛА, ємність зон приймання.

Критерієм виступатиме мінімум сумарної довжини (часу) польоту для виконання завдання; мінімум БПЛА [28].

2.3 Математична модель

Вважаємо заданими:

m – загальна кількість БПЛА,

b – кількість місць старту ($b \geq m$);

e – кількість зон приймання БПЛА ($e \geq 1$);

n – кількість об'єктів, які слід відвідати;

R^k – польотний ресурс k-го БПЛА, $k = 1,...,m$.

Підкреслимо, що кількість місць старту БПЛА не може бути меншою m, а в разі, коли $m > n$, частина із них не буде задіяною.

Розглянемо зважений граф $G = (Y,U)$, де вершини Y відповідають об'єктам, місцям старту та зонам приймання, тобто $\| Y \| = n + b + e$, задавши таку їх нумерацію: від 1 до n – об'єкти, від $n+1$ до $n+b$ – місця старту, а від $n+b+1$ до $n+b+e$ – зони (їх центроїди).

Позначимо $V = \{1,...,n\}$ – множина вершин, які відповідають об'єктам, $B = \{n+1,...,n+b\}$ – множина вершин, які відповідають місцям старту,

$E = \{n+b+1,...,n+b+e\}$ – вершини, що відповідають зонам приймання. Зрозуміло, що для вершин із множини B матимемо лише вихідні дуги, а для вершин із E – лише вхідні.

Для подання розв'язку введемо матрицю $X = (x_{ij})$ розмірності $m \times (n+b+e)$, де $x_{ij} \in \{0,1\}$, яка складається із трьох блоків. Перший блок включає стовпчики від 1 до n, другий – від $n+1$ до $n+b$, третій – від $n+b+1$ до $n+b+e$, а рядки будуть відповідати номерам БПЛА. Ця матриця буде описувати фактично етап розбиття, на якому кожному БПЛА приписуються ті об'єкти, які він має відвідати, разом із одним із місць старту та місцем завершення маршруту.

Позначимо δ_j максимальну кількість БПЛА, які можуть бути прийняті в зоні j, $j = 1,...,e$.

Задача розбиття (як окремий етап чи складова загального процесу розв'язування) може бути подана так.

Обмежуючі умови:

$$\sum_{j=n+1}^{n+b} x_{ij} \le 1, \quad i = 1,...,m; \tag{2.1}$$

$$\sum_{j=n+b+1}^{n+b+e} x_{ij} \le 1, \quad i = 1,...,m; \tag{2.2}$$

$$\sum_{i=1}^{m} x_{ij} = 1, \quad i = 1,...,n; \tag{2.3}$$

$$\sum_{i=1}^{m} x_{ij} \le 1, \quad j = n+1,...,n+b; \tag{2.4}$$

$$\sum_{j=n+1}^{n+b} \sum_{i=1}^{m} x_{ij} = m; \tag{2.5}$$

$$\sum_{i=1}^{m} x_{ij} \le \delta_{j-n-b}, \quad j = n+b+1,...,n+b+e; \tag{2.6}$$

$$L^i(\overline{x_i}) \le R^i, \quad i = 1,...,m. \tag{2.7}$$

Задача полягає в пошуку такого розв'язку X, для якого

$$F(X) \equiv \sum_{i=1}^{m} L^i(\overline{x_i}) \to \min, \qquad (2.8)$$

за умов (1) – (7), де $\overline{x_i}$ – це i-й вектор-рядок матриці X, а $L^i(\overline{x_i})$ – довжина маршруту i-го задіяного БПЛА, як розв'язку задачі комівояжера, який визначається вершинами, що відповідають ненульовим компонентам вектора $\overline{x_i}$ починаючи з певного стартового місця $s \in B : x_{is} = 1$ і закінчуючи зоною $f \in E : x_{if} = 1.$

Зрозуміло, що для незадіяних БПЛА відповідний вектор-рядок має всі нульові координати.

Тобто матриця X, як елемент простору розв'язків задачі (2.1)–(2.8), у компактному вигляді може бути подана так:

$$X = \begin{pmatrix} \overline{x_1} \\ \vdots \\ \overline{x_m} \end{pmatrix},$$

де $\overline{x_i} = (x_{i1}, x_{i2}, ..., x_{i,n+b+e})$.

Умова (2.1) визначає, що кожен і-й БПЛА стартує лише з одного із можливих місць старту (чи момент часу, який і визначає таке місце), якщо ж деякий БПЛА не задіяний, то відповідний рядок стає нульовим. Аналогічно, формула (2.2) описує ситуацію, що до конкретного центроїда від об'єктів або веде одне і лише одне ребро графа G, або жодного. Умова (2.3) визначає, що кожен об'єкт $j \in V$ відвідується лише одним БПЛА. Умова (2.4) визначає, що з кожного місця або стартує один БПЛА, або ж це місце не задіяне, а разом із умовою (2.5) – що кількість задіяних місць співпадає із кількістю БПЛА. Можливість приймати в кожній зоні $j \in E$ (в іншому трактуванні – у момент часу) не більше, ніж δ_j БПЛА, відображена у формулі (2.6). Обмеження на польотний ресурс кожного літального апарату визначено умовою (2.7).

Зауважимо, що не виключаються випадки, коли в одній зоні (чи в момент часу) може прийматися лише один БПЛА, тоді $\delta_j = 1$ для відповідної зони, чи всі БПЛА можуть бути прийняті в одній зоні – в цьому разі $e = 1$, а $\delta_1 = m$.

Науковою новизною у даній постановці задачі є модифікація [29] за рахунок введення зон приймання з певними ємностями.

Таким чином, ця задача подібна до задачі маршрутизації транспортних засобів із декількома депо (Multiple Depot Vehicle Routing Problem, MDVRP) з додатковими обмеженнями.

MDVRP є більш складною задачею, ніж класична задача VRP. Крім того, MDVRP є NP-складною задачею, це означає, що не існує ефективного алгоритму для отримання оптимального її розв’язку. Точні методи, такі як метод гілок і меж, метод гілок і відсікання, є неефективними для розв’язку MDVRP через надмірну трудомісткість. Для того, щоб ефективно розв'язувати подібні задачі, використовуються наближені методи [28].

2.4 Алгоритми розв’язання та їх класифікація

Найуживаніші на практиці наближені алгоритми можна поділити на сім класів [30], які зображені на рисунку 2.3.

Досить потужними є алгоритми, які основані на природних явищах, найбільш широко вживаними є:

- оптимізація мурашиними колоніями;
- генетичний алгоритм;
- оптимізація потоком частинок;
- бджолині алгоритми.

Мурашине суспільство - це добре організована та ієрархічна структура з конкретними ролями та визначеною поведінкою. Хоча існує багато видів мурашок і кожен з них має свої особливості, ми можемо виділити спільну організаційну структуру: королеву та робітників. Робітники виконують різні завдання колонії: вигодовування, утримання гнізд, догляд за личинками, захист тощо. Саме завдяки спостереженню за поведінкою мурашок у 1992 році Марко Доріго запропонував алгоритм оптимізації мурашиними колоніями [31].

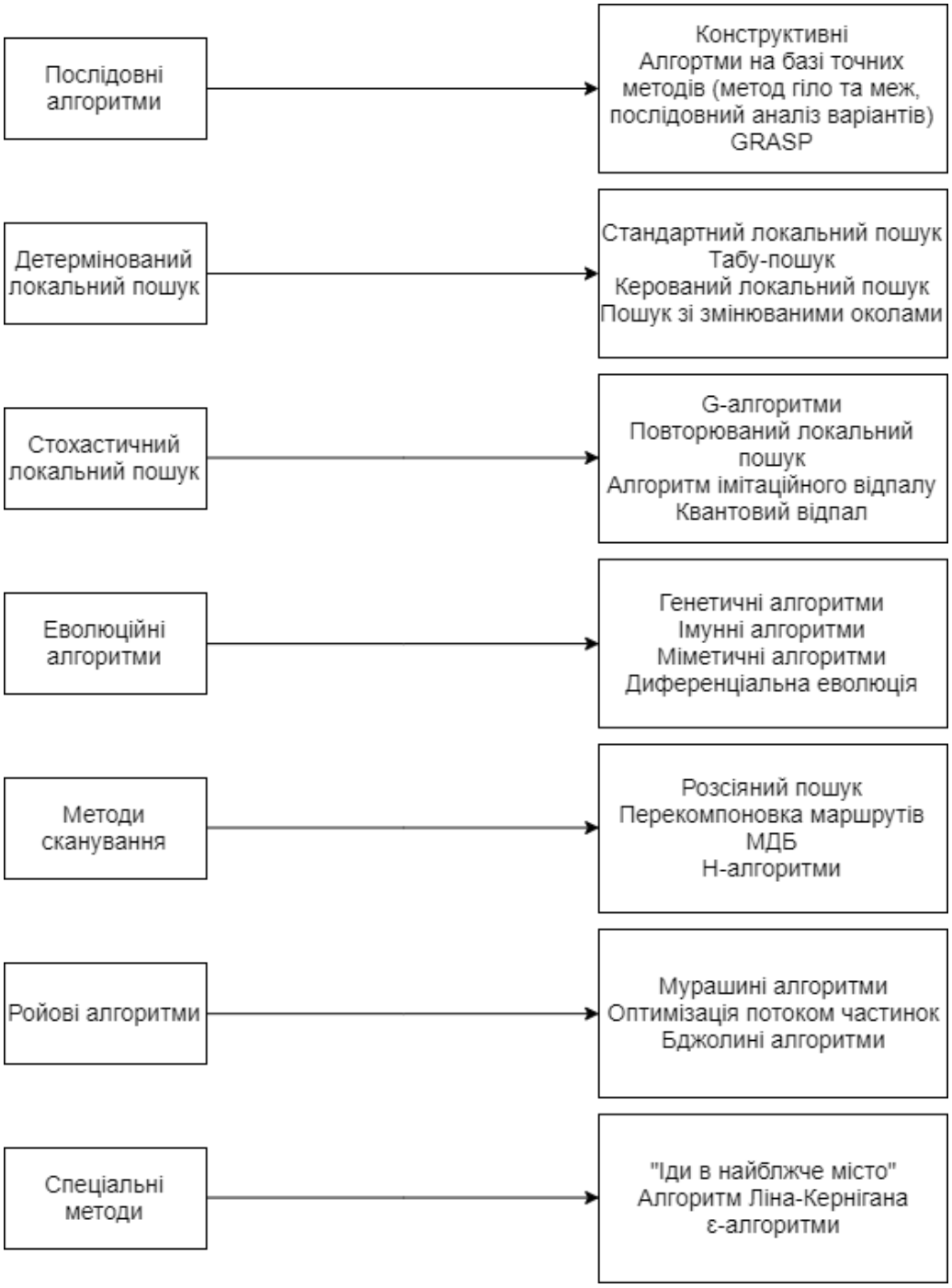


Рисунок 2.3 - Класифікація наближених методів комбінаторної оптимізації

Ідея виникла зі спостереження за поведінкою мурашника: коли мураха покидає гніздо в пошуках їжі, вона закладе феромонний слід, за яким інші мурахи здатні йти; якщо вона знайде їжу, то повернеться до гнізда - використовуючи той самий шлях - залишивши ще більше феромону на шляху назад. Якщо вона не знайде їжі, вона не відкладе жодного феромону на шляху до гнізда, і попередній слід випарується. Якщо ми застосуємо таку поведінку на великій колонії, ми побачимо, що з'являються сильні феромонні сліди, що зв'язують гніздо з джерелами їжі. Цей непрямий механізм комунікації на основі

феромона лежить в основі обміну знаннями в колонії і дозволяє мурахам знайти кращі шляхи до їжі. Однією із найсильніших переваг оптимізації мурашиними колоніями є її здатність швидко знаходити хороше рішення; ця перевага врівноважується можливою передчасною збіжністю.

Генетичний алгоритм використовує методи з еволюційної біології, такі як відбір, мутація, успадкування та рекомбінація для вирішення задач комбінаторної оптимізації. Метод полягає у створенні групи осіб випадковим чином із популяції. Їм надається оцінка, яка підкреслює їх пристосованість. Кращі дві особини потім використовуються для створення одного або декількох потомства за допомогою оператора схрещення, після чого на потомстві робляться випадкові мутації. Залежно від потреб, ця процедура триває до тих пір, поки не буде знайдено прийнятне рішення або поки не пройде певна кількість поколінь. Генетичний алгоритм відрізняється від класичних алгоритмів оптимізації тим, що генерує сукупність точок на кожній ітерації, тоді як класичний алгоритм генерує одну точку на кожній ітерації. Також генетичний алгоритм вибирає наступну сукупність шляхом обчислень з використанням генератору випадкових чисел, тоді як класичний алгоритм вибирає наступну точку шляхом детермінованих обчислень. У порівнянні з іншими методами генетичний алгоритм є більш надійним і менш схильним до «поломки» через незначні зміни у вхідних даних.

Оптимізація потоком частинок - це методика стохастичної оптимізації, розроблена Еберхартом та Кеннеді в 1995 році [32], натхненна соціальною поведінкою зграї птахів. Оптимізація потоком частинок має багато подібного з еволюційними методами обчислення, такими як генетичні алгоритми. Система ініціалізується з сукупністю випадкових розв’язків та шукає оптимуму з використанням оновлення поколінь. Однак, на відміну від генетичного алгоритму, немає таких операторів еволюції як кросовер та мутація. У цьому методі потенційні розв’язки, звані частинками, рухаються у просторі розв’язків, слідуючи за поточними найкращими частинкам. Кожна частинка відслідковує свої координати в просторі розв’язків, які пов'язані з найкращим розв’язком (придатністю), якого вона досягла - це значення називається pbest. Ще одне

"найкраще" значення, яке відстежується оптимізатором потоку частинок, - це найкраще значення, отримане серед сусідніх частинок – позначається lbest. Коли частинка розглядає всю популяцію як своїх топологічних сусідів, найкращий розв’зок - gbest.

Концепція оптимізації потоком частинок заснована на використанні зміни швидкості на кожному етапі (прискорення) кожної частинки у напрямку до її pbest та lbest координат. Прискорення визначається випадковим чином. Одна з причин привабливості даного алгоритму - це те, що для налаштування є мало параметрів. Одна й та ж сама версія, з незначними варіаціями, досить добре працює для широкого спектру задач.

Бджолині алгоритми - це сім'я алгоритмів, яка створена на основі спостереження за поведінкою бджіл при пошуках їжі. Спостереження за бджолами показало наступну послідовність дій:

КРОК 1. З вулика вилітають бджоли-розвідники і випадковим чином обирають напрям руху.

КРОК 2. Бджоли-розвідники намагаються відшукати місця з їжею, таким чином формуючи розв'язки.

КРОК 3. Після повернення до вулику бджоли-розвідники за допомогою бджолиного танцю повідомляють іншим бджолам, де знаходиться їжа.

КРОК 4. Далі бджоли-робітники летять для збору нектару.

Характерно те, що бджоли передають інформацію про кількість їжі і чим більше її, тим більше бджіл-робітників полетить до того джерела. Бджоли-розвідники ж вирушають на пошуки інших місць з їжею, цей процес циклічно повторюється. Середня кількість бджіл-розвідників оцінюється у 10-15 відсотків від усіх наявних. Особливістю бджолиних алгоритмів є те, що бджоли-робітники прямують не саме в те місце, де були бджоли-розвідника, а в їх околи, уточнюючи координати місць випадковим чином [30].

Також за останні роки було запропоновано багато інших підходів до стохастичної оптимізації, які основані на природних явищах. Більшість цих підходів можна віднести до однієї з трьох категорій [33]:

- підходи, що засновані на поведінці живих організмів;
- підходи, що використовують поведінкові шаблони фізичних, хімічних чи інших природних явищ;
- підходи, що засновані на поведінці людей в різних аспектах життя.

До першої категорії належать алгоритми, які використовують поведінку таких біологічних організмів:

- павуки [34];
- кити [35];
- сірі вовки [36];
- птахи [37];
- бактерії [38];
- світлячки [39];
- інвазійний бур’ян [40].

До другої категорії належать алгоритми, які використовують поведінкові шаблони наступних явищ:

- електромагнетизм [41];
- чорні діри [42];
- ерозія річок [43];
- великий вибух [44];
- електрика [45];
- кругообіг води [46].

До третьої категорії належать алгоритми, які пов'язані з наступними аспектами:

- музиканти [47];
- політичний імперіалізм [48];
- викладачі [49];
- соціальна поведінка [50].

У даній роботі у якості основного алгоритму буде досліджуватися оптимізація мурашиними колоніями, оскільки ефективність її застосування підтверджується досвідом розв'язування різних задач маршрутизації транспорту [25, 51-53].

2.5 Алгоритм прямого перебору

Найпростіший точний метод, який можна застосувати – це прямий перебір всіх можливих варіантів. Алгоритм повного перебору всіх можливих розв’язків задачі складається з чотирьох етапів.

Перший етап – формування всіх можливих перестановок початкових депо. Для визначення кількості перестановок з b елементів маємо формулу $P_b = b!$.

Другий етап – розбиття множини з n об'єктів, які необхідно відвідати. Загальна кількість поділів довільної n-елементної множини дорівнює числу Белла, яке можна обчислити як суму чисел Стірлінга другого роду, $B_n = \sum_{m=0}^{n} S(n,m)$.

Третій етап – формування всіх можливих розміщень для e зон приймання. Загальна кількість розміщень визначається за наступною формулою $A_e^b = \frac{b!}{(b-e)!}$.

Останній етап – композиція всіх маршрутів отриманих на перших трьох кроках, загальна кількість варіантів $k = b! \sum_{m=0}^{n} S(n,m) \frac{b!}{(b-e)!}$.

Через таку надзвичайно велику кількість варіантів можна зробити висновок, що методом повного перебору за прийнятний час неможливо знайти оптимальний розв'язок. Його застосування виправдане лише для задач невеликої розмірності для отримання точного розв’язку [26].

2.6 Алгоритм локального пошуку

Алгоритми локального пошуку використовують поняття околу в просторі розв’язків задачі X. Варіант розв’язку складається з множини маршрутів – послідовності номерів клієнтів, які відвідуються кожним БПЛА. Кожен маршрут починається з стартового депо та закінчується зоною приймання. Наприклад, маємо два БПЛА. Перший стартує з місця старту №1, відвідує клієнта №4, потім клієнта №3 та летить у зону приймання №6. Другий починає

рух з місця старту №2, відвідує клієнтів №5, №7, №8 та летить у зону приймання №9 [6].

$$\begin{Bmatrix} 1,4,3,6 \\ 2,5,7,8,9 \end{Bmatrix}$$

Для генерації околів $O(x)$ довільної точки $x \in X$ застосовується оператор переміщення, який є одним з найбільш ефективних способів модифікації маршруту [54].

Для розв'язування цієї задачі пропонується алгоритм табуйованого пошуку, який довів свою ефективність при розв'язанні задач класу MDVRP [55].

Пвсевдокод алгоритму табуйованого пошуку наведений у графічних матеріалах.

На етапі ініціалізації формується початковий розв'язкок отриманий за допомогою конструктивного алгоритма, який реалізує принцип жадібного вибору. Критерієм завершення роботи алгоритму є кількість можливих ітерацій, на яких не відбувається покращення розв'язку. Задається розмір списку заборон, а також кількість ітерацій, під час яких кожне переміщення може знаходитися у списку заборон.

2.7 Диверсифікований макс-мін алгоритм мурашиних систем

Більш інтенсивне використання кращих розв'язків у ході пошуку призводить до підвищення продуктивності алгоритмів оптимізації мурашиною колонією [56], але при цьому підвищується ризик передчасної збіжності. Тому доречно комбінувати використання кращих розв'язків та методи запобігання передчасній збіжності. Для задоволення цих двох потреб розроблений спеціальний максі-мінний алгоритм мурашиних систем як покращення алгоритму мурашиних систем [57].

Модель задачі подається у вигляді повного зваженого графа $G\ V,E$, де $v_i \in V,\ i = 1,...,n$ – вершини, що відповідають компонентам розв'язку задачі, а $e_{ij} \in E,\ e_{ij} = (v_i, v_j),\ v_i, v_j \in V$ – множина можливих переходів між відповідними

вершинами. Також маємо набір обмежень $\Pi = \Pi(V, E, t)$ для елементів V та E, де t – параметр часу. Обмеження визначають припустимість зв'язків між компонентами та переходами і побудованого з них розв'язку. Також введемо додаткові поняття.

Евристична інформація η_{ij} – це числове значення, що не залежить від знайдених на попередніх кроках розв'язків і відображає ступінь бажаності включення в побудований фрагмент розв'язку того чи іншого нового ребра графа моделі $e_{ij} \in E$. Евристичні значення η_{ij} базуються на апріорній інформації, що відображає умови конкретної задачі та надається джерелом, відмінним від мурах [58].

Рівень феромону τ_{ij}, що відповідає ребру $e_{ij} \in E$, – це додатне число, яке показує, наскільки часто мурахами використовувалося це ребро на попередніх кроках чи при формуванні повного розв'язку. Феромонні сліди є для мурах довготривалою пам'яттю щодо всього процесу пошуку. Залежно від вибраного способу подання задачі, феромонні сліди можуть відповідати всім дугам задачі або тільки деяким з них [58].

Розглянемо обчислювальну схему алгоритму.

На першому кроці відбувається визначення кількості мурах (у нашому випадку кількість мурах рівна кількості БПЛА) та відбувається початкова ініціалізація матриці феромонів. Для уникнення стагнації розв’язку вводиться інтервал значень для феромону, тобто вводиться поняття нижньої та верхньої межі. На початку роботи алгоритму матриця феромонів ініціалізується значеннями нижньої межі феромону. Розв’язок будується покроково: поточна мураха вибирає наступну вершину графа задачі, після переходу в яку ця вершина стає недоступною для відвідання мурахами, які роблять свій крок пізніше [28]. Для всіх мурах на кожному кроці відбувається наступна послідовність дій [30]:

- формування підмножини припустимих вершин, які може відвідати мураха;

- обчислення ймовірності переходів від поточної вершини і до всіх допустимих вершин;
- вибір допустимої вершини і перехід в неї.

Кожна мураха формує лише частковий розв’язок задачі, а їх сукупність дає повний розв’язок. Через це проводиться заздалегідь визначена кількість ітерацій, формується набір часткових розв’язків і з них обирається найкращий, на ньому відкладається феромон. На наступних етапах відбуються такі дії:

- випаровування феромонів;
- оновлення допустимих нижньої та верхньої межі феромону;
- оновлення матриці феромонів відповідно нижньої та верхньої межі;
- перевірка стагнації.

Детальніше розглянемо правила переходу до наступної вершини та процес обчислення ймовірностей переходів. Стани задачі визначаються в термінах скінченних послідовностей $y = (v_{s_1}, v_{s_2}, ...), v_{s_r} \in V,$ елементів V (або, що рівносильно, E), які на всіх проміжних кроках мурахи є фрагментами розв'язку задачі оптимізації. Якщо Y - множина всіх можливих послідовностей, то множина Y усіх (під)послідовностей, які задовольняють обмеження $\Pi = \Pi(V, E, t)$, є підмножиною $Y : Y \subseteq Y$, а її елементи визначають припустимі стани задачі. Нехай на певному кроці мураха k побудувала фрагмент розв'язку y, останньою компонентою якого є вершина $i \in V,$ тобто вона перебуває в цій вершині: $y = (..., i)$. Тоді мураха може переміститися до будь-якої вершини j із множини припустимих сусідніх вершин N_i^k , визначуваних як $N_i^k = \{ j : j \in N_i \wedge (y, j) \in Y \}$, де N_i - множина всіх сусідніх до i вершин графа задачі [58]. Вибір наступної вершини відбувається за псевдовипадковим пропорційним правилом. Введемо новий параметр $p_0 \in [0,1]$, кожна мураха переходить з вершини $i \in V$ до вершини $j \in N_i^k$ з імовірністю p_0, j визначається наступним чином:

$$j = \arg\max\{\tau_{ir}^{\alpha}(t)\eta_{ir}^{\beta}(t) : r \in N_i^k\}$$

З імовірністю $1-p_0$ обирається вершина за правилом колеса рулетки з використанням імовірності p_{ij}^k (описана далі).

Для диверсифікації пошуку використовується двокроковий перехід [58]. Тобто мурахи можуть робити перехід як на один крок, так і на два. Тоді визначимо імовірність переходу *k*-ої мурахи з вершини *i* в *j* через вершину *s* на поточній ітерації *t*:

$$p_{ij}{}^k = \frac{[\tau_{is}(t)+\tau_{sj}(t)]^\alpha[\eta_{is}(t)+\eta_{sj}(t)]^\beta}{\sum_{r\in N_{irj}{}^k}[\tau_{ir}(t)+\tau_{rj}(t)]^\alpha[\eta_{ir}(t)+\eta_{rj}(t)]^\beta},$$

де $N_{irj}{}^k=\{l: l\in N_i{}^k,\ j\in N_l{}^k\}$.

Також розглянемо мультиплікативний варіант даної формули:

$$p_{ij}{}^k = \frac{[\tau_{is}(t)\cdot\tau_{sj}(t)]^\alpha[\eta_{is}(t)+\eta_{sj}(t)]^\beta}{\sum_{r\in N_{irj}{}^k}[\tau_{ir}(t)\cdot\tau_{rj}(t)]^\alpha[\eta_{ir}(t)+\eta_{rj}(t)]^\beta}.$$

Оскільки ми розглядаємо одночасно переходи як на один, так і на два кроки, то виникає необхідність їх нормування, щоб у сумі була одиниця. Імовірність переходу на один крок від вершини *i* до вершини *j* , $j\in N_i{}^k$, та на два кроки від вершини *i* до іншої вершини *j* через вершину *r*, $r\in N_i{}^k,\ j\in N_r{}^k\setminus i$, для адитивного варіанту буде визначатися наступним чином:

$$p_{ij}{}^k=\begin{cases}\dfrac{\tau_{ij}(t)^\alpha\eta_{ij}(t)^\beta}{\sum_{r\in N_i{}^k}\tau_{ir}(t)^\alpha\eta_{ir}(t)^\beta+\sum_{r\in N_{irj}{}^k}[\tau_{ir}(t)+\tau_{rj}(t)]^\alpha[\eta_{ir}(t)+\eta_{rj}(t)]^\beta}, & \textit{якщо } j\in N_i{}^k,\\ \dfrac{[\tau_{is}(t)+\tau_{sj}(t)]^\alpha[\eta_{is}(t)+\eta_{sj}(t)]^\beta}{\sum_{r\in N_i{}^k}\tau_{ir}(t)^\alpha\eta_{ir}(t)^\beta+\sum_{r\in N_{irj}{}^k}[\tau_{ir}(t)+\tau_{rj}(t)]^\alpha[\eta_{ir}(t)+\eta_{rj}(t)]^\beta}, & \textit{якщо } j\in N_r{}^k.\end{cases}$$

Мультиплікативну версію можна подати наступним чином:

$$p_{ij}{}^k=\begin{cases}\dfrac{\tau_{ij}(t)^\alpha\eta_{ij}(t)^\beta}{\sum_{r\in N_i{}^k}\tau_{ir}(t)^\alpha\eta_{ir}(t)^\beta+\sum_{r\in N_{irj}{}^k}[\tau_{ir}(t)\square\tau_{rj}(t)]^\alpha[\eta_{ir}(t)+\eta_{rj}(t)]^\beta}, & \textit{якщо } j\in N_i{}^k,\\ \dfrac{[\tau_{is}(t)\square\tau_{sj}(t)]^\alpha[\eta_{is}(t)+\eta_{sj}(t)]^\beta}{\sum_{r\in N_i{}^k}\tau_{ir}(t)^\alpha\eta_{ir}(t)^\beta+\sum_{r\in N_{irj}{}^k}[\tau_{ir}(t)\square\tau_{rj}(t)]^\alpha[\eta_{ir}(t)+\eta_{rj}(t)]^\beta}, & \textit{якщо } j\in N_r{}^k.\end{cases}$$

Експериментальним шляхом було обрано мультиплікативний варіант, оскільки він показує кращі результати. Це можна пояснити тим, що при використанні адитивного варіанту сумарна імовірність двокрокових переходів значно перевищує сумарну імовірність однокрокових, це приводить до того, що майже завжди відбуваються двокрокові переходи. Використання лише двокрокових переходів робить алгоритм більш жадібним, що забезпечує лише миттєву вигоду, у той час як в цілому результат може виявитися несприятливим.

Одночасна можливість як однокрокових, так і двокрокових переходів за один такт дозволяє отримувати кращий розв'язок і при цьому уникати локальних оптимумів [58].

Відкладання та випаровування феромонів відбувається за такою формулою:

$$\tau_{ij}(t+1) = \rho\tau_{ir}(t) + \frac{(1-\rho)}{f^0_{\min}}\},$$

де ρ - коефіцієнт випаровування, що лежить в межах від 0 до 1, а $f^0_{\min}$ - найкраще значення цільової функції на початковій популяції мурах.

Нижня та верхня межа феромонів визначаються за наступними формулами [59]:

$$\tau_{\max} = 1/(\rho f^0_{\min}),$$

$$\tau_{\min} = [\tau_{\max}(1-\sqrt[n]{0.05})]/((\frac{n}{2}-1)\sqrt[n]{0.05})),$$

де $n = |V|$.

Коригування матриці феромонів відбувається наступним чином:

$$a = \min(a, \tau_{\max}),\ a = \max(a, \tau_{\min}),$$

де a – елемент матриці феромонів.

Для боротьби зі стагнацією здійснюється скиданням значень матриці феромонів у початковий стан, якщо за визначену кількість ітерацій не відбулося покращення найкращого розв'язку.

Псевдокод диверсифікованого макс-мін алгоритму мурашиних систем алгоритму наведений у графічних матеріалах.

У псевдокоді П – предикат, що описує обмеження задачі (більш детальніший опис обчислювальної схеми див. в [30]).

Кожна мураха формує лише частковий розв’язок, а не повний як у випадку задачі комівояжера. Набір часткових розв’язків формує розв’язок. Так як мураха формує не повний розв'язок, то не є доцільним відкладання феромону на одній з мурах. Цю проблему вирішимо за рахунок створення декількох популяцій, кожна з яких буде формувати розв'язок під час однієї ітерації. Ітерацією будемо вважати набір дій від формування популяцій до відкладання феромону. У кінця ітерації обирається популяція з найкращим розв'язком і лише на ній відбувається відкладання феромонів.

2.8 Застосування острівної моделі до макс-мін алгоритму мурашиних систем

Острівна модель – це модель паралельного алгоритму, яка вперше була застосована до генетичного алгоритму. Кожна популяція розбивається на частини, кожна з яких незалежно розвивається на окремому острові. При цьому інколи відбувається обмін особинами між островами. Описана далі острівна модель для макс-мін алгоритму мурашиних систем базується на роботах [59, 60].

На початку острівного алгоритму на островах незалежно відбувається ініціалізація мурашиних колоній, а далі починає роботу макс-мін алгоритм мурашиних систем, який був розглянутий у попередньому підрозділі. Єдина відмінність у тому що через кожні m ітерацій відбувається обмін інформацією про «хороші» розв’язки між островами, називатимемо цей процес міграцією.

Важливу роль у роботі острівної моделі відіграє топологія островів, тобто те, як вони між собою пов’язані. Лише пов’язані між собою острови можуть обмінюватися інформацією. Оскільки макс-мін алгоритм мурашиних систем зберігає лише найкращий глобальний розв’язок, то саме він буде виступати у ролі мігранта, тобто буде відправлений на «сусідні» острови під час процесу міграції. Нехай K_i - множина сусідніх островів для острова i. Тоді мігрант для острова i визначається наступним чином:

$$m_i = \arg\min(S_r : r \in K_i),$$

де S – множина, що містить найкращі глобальні розв’язв’язки на островах.

Після потрапляння мігранта на острів відбувається відкладання феромону на його роз’язку, за умови що розв’язок-мігрант є кращим за поточний найкращий глобальний розв’язок на острові.

Робота острівного алгоритму відбувається до тих пір поки є хоч один острів, на якому кількість ітерацій без покращення менше заданої.

Блок-схема організації острівної моделі та псевдокод острівної моделі диверсифікованого макс-мін алгоритму мурашиних систем для N островів наведені у графічних матеріалах.

Досить важливо правильно підібрати кількість островів та частоту міграції. При досить частій міграції можна втратити всю перевагу острівної моделі, алгоритм буде схожий більше на класичний, оскільки буде сильне змішування популяцій, і навпаки, при якщо міграції будуть проходити досить рідко, то можлива ситуації передчасної збіжності.

У даній роботі буде проведено аналіз чотирьох топологій:

- повна;
- кільце;
- решітка;
- гіперкуб.

При повній топології всі острова пов’язані між собою, таким чином, найкращий розв’язок поширюється найшвидше, але різноманітність при цьому зменшується.

Крім того, якщо найкращий розв’язок це локальний оптимум, то всі острови мають тенденцію застрявати в ньому.

Кільце - це топологія, згідно з якою всі острови пов'язані у кільце. Графічне зображення цієї топології представлено на рисунку 2.4.

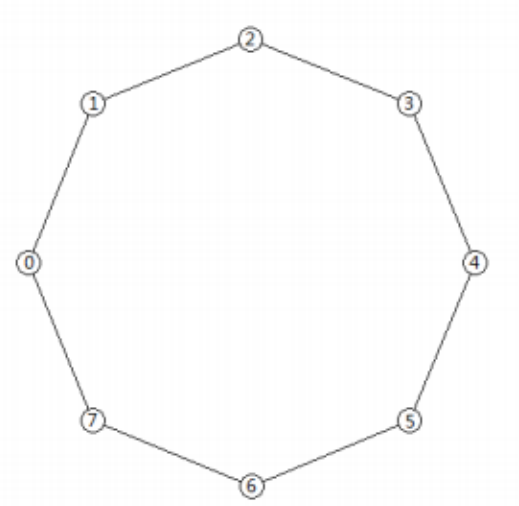


Рисунок 2.4 – Топологія кільце

Решітка - це альтернативна назва двовимірного (n, d)-тору. Двовимірний (n, d)-тор - це сітка, що має n-рядків та d-стовпців з краями, обернутими з усіх боків. Зразок (3,4)-тору зображений на рисунку 2.5.

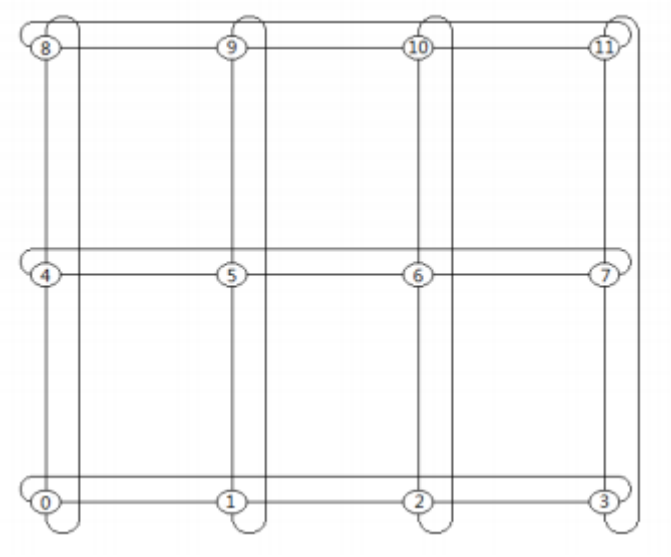


Рисунок 2.5 – Топологія решітка

Гіперкуб - топологія має обмеження на кількість островів $m = 2^k, k \in \square$. Нумерація островів йде від 0 до m -1. Два острови u і v пов'язані тоді і лише тоді, коли $u \ XOR \ v = 2^l, l \in \square$. Іншими словами, якщо записати номера островів у двійковій формі, тоді два острови з'єднані, якщо і лише тоді коли у них відрізняється значення лише одного двійкового значення. Зразок гіперкуба на 8 острів показаний на рисунку 2.6.

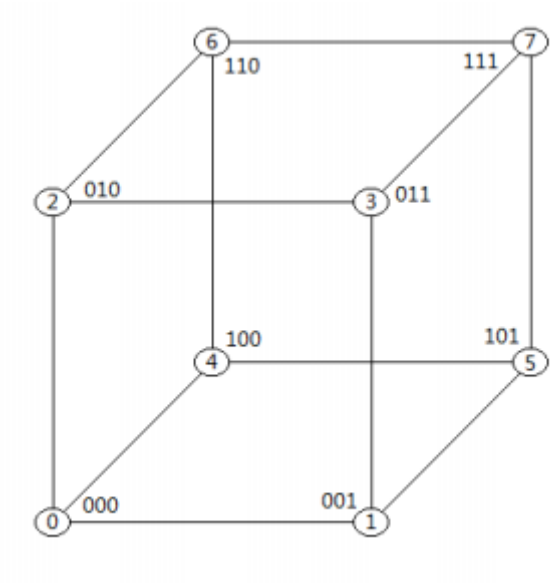


Рисунок 2.6 – Топологія гіперкуб

Далі буде проведений ряд експериментів, який дозволить виявити яка топологія є кращою. Використання острівної моделі дозволяє розпаралелити алгоритм для декількох популяцій і ефективно обмінюватися даними для отримання кращого розв’язку.

2.9 Підхід до налаштування параметрів алгоритму

Алгоритм мурашиних систем має набір параметрів, які потрібно налаштувати перед роботою алгоритму, від цього залежить якісь розв’язку. Складемо задачу комбінаторної оптимізації. Нехай у нас є m параметрів мурашиного алгоритму M. Введемо вектор параметрів $p = (p_1,\ldots,p_n)$. Кожен параметр має своє мінімальне та максимальне значення, задамо їх наступними векторами $p_{min} = (p_{1\ min},\ldots, p_{n\ min})$, $p_{max} = (p_{1\ max},\ldots, p_{n\ max})$. Також введемо значення на яке буде змінюватися кожен з параметрів: $\Delta h = (\Delta h_1,\ldots, \Delta h_n)$.

Для кожного набору параметрів будемо проводити k запусків алгоритму M, які дадуть нам вектор розв’язків $r = (r_1,\ldots,r_k)$. Введемо поняття середньої якості набору параметрів, яке знаходиться за наступною формулою 2.9:

$$l_{avg} = \frac{\sum_{i=1}^{k} r_k}{k} \tag{2.9}$$

Тоді задача налаштування параметрів виглядає так:

$$x_* = arg\min_{x \in R^n} F \quad p, p_{min}, p_{max}, \Delta h, k, M$$

За допомогою Δh ми керуємо точністю, з якою треба знайти значення параметрів для нашого алгоритму.

Зміна параметрів у решітці відбувається в одиничному околі шляхом зменшення або збільшення значення компоненти на величину з вектору Δh, при цьому враховуючи мінімальну та максимальну межу.

Для цієї задачі комбінаторної оптимізації за основу візьмемо один з варіантів детермінованого локального пошуку – метод векторного спуску (МВС). Цей алгоритм був запропонований І. В. Сергієнком [30].

Вхідними даними для алгоритму є початковий варіант розв’язку. Його псевдокод наводиться на рисунку 2.7.

```
procedure MBC(x)
  Задання ρ > 0;
  x^0 : = деякий припустимий варіант розв'язку;
  h : = 0;
  while окіл L_ρ(x^h) поточного варіанта не переглянутий
повністю do
    y : = Генерація Чергової Точки Околу L_ρ(x^h);
    Δ = f(x^h) – f(y);
    if Δ > 0 then
      h : = h + 1; x^h : = y;
    end if;
  end while;
  return x = x^h;   {x^h – локально оптимальний розв'язок задачі}
end
```

Рисунок 2.7 – Псевдокод МВС

де $f(x)$ – цільова функція задачі.

Вхідними даними для алгоритму налаштування параметрів алгоритму M є алгоритм M, вектори p_{min}, p_{max}, p_{start}, крок зміни Δh параметрів алгоритму M та k – кількість запусків алгоритму M на одному наборі параметрів. Псевдокод підходу до налаштування параметрів алгоритму наведений у графічних матеріалах.

Ключовими елементами даного алгоритму є:

- генератор нових значень параметрів алгоритму;

- вектори мінімальних та максимальних значень параметрів, p_{min} та p_{max} відповідно і вектор кроків параметрів Δh;
- параметр k, що відповідає за кількість прогонів алгоритму M.

Генератор значень параметрів, що використовується для даного алгоритму є кільцевим генератором, має радіус рівний одиниці, тобто за один прохід змінюється лише один параметр на певний крок з вектору Δh.

Висновки до розділу

Наведена постановка задачі маршрутизації безпілотних авіаційних систем, а також її математична модель.

Описані підходи до розв'язання задачі, такі як прямий перебір, локальний пошук, оптимізація мурашиними колоніями, острівна модель для оптимізації мурашиними колоніями. Також були подані ключові аспекти реалізації алгоритмів у вигляді псевдокоду.

Також були розглянуті різні топології острівної моделі, ефективність кожної з них буде досліджена в рамках обчислювального експерименту.

Сформульована підхід до підбору параметрів оптимізації мурашиними колоніями, де була поставлена задача оптимізації параметрів та розглянутий підхід до вирішення її на основі методу векторного спуску, який є представником детермінованого локального пошуку.

3 ОПИС ПРОГРАМНОГО ТА ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вимоги до програмного продукту

Програмний продукт містить лише одного актора, це користувач. Далі наведено опис даного актору.

Користувач. Це будь-який користувач даного програмного продукту. Він може розв’язувати задачу маршрутизації безпілотних авіаційних систем. Також здійснювати налаштування параметрів алгоритму у ручному режимі.

Наведемо варіанти використання:

- знаходження розв’язку – знайти розв’язок задачі, яку завантажив користувач;
- ручне налаштування параметрів алгоритму – ввести значення параметрів алгоритму оптимізації мурашиними колоніями;
- автоматичний підбір параметрів – автоматично налаштувати параметри алгоритму оптимізації мурашиними колоніями на основі задачі, яку завантажив користувач
- візуалізація розв’язку задачі – показати маршрут кожного літального апарату як граф, вершинами якого є точки, що потрібно відвідати .

У таблиці 3.1 наведено виявлені функціональні вимоги.

Таблиця 3.1 – Виявлені функціональні вимоги

Актор	**Варіант використання**	**Функціональна вимога**	**Пріоритет**
Користувач	Знайти розв’язок	Система надає користувачу розв’язати задачу комівояжера з вхідними даними, які надає користувач.	Високий

Продовження таблиці 3.1

Актор	Варіант використання	Функціональна вимога	Пріоритет
Користувач	Завантажити задачу	Система повинна надати користувачу можливість завантажити файл з задачею та перевірити коректність задачі	Високий
Користувач	Підібрати параметри для алгоритму	Система надає можливість ввести значення параметрів алгоритму.	Високий
Користувач	Автоматично підібрати параметри для алгоритму	Система надає можливість підібрати параметри алгоритму за рахунок прогонки серії випробувань на задачі, яку завантажить користувач.	Середній
Користувач	Візуалізувати розв’язок	Система надає користувачу можливість побачити маршрут кожного літального апарату	Середній

Як нефункціональні вимоги можна описати наступні:

- система повинна робити перевіряти коректність задачі, у разі помилок система повинна показувати користувачу повідомлення з інформацією про помилку;
- система повинна знаходити розв'язок кожної задачі, яка пройшла валідацію, за детермінований час.

3.2 Засоби розробки

Програмне забезпечення було створене за допомогою мови Kotlin. Kotlin – це статично типізована мова програмування, що орієнтована на Java Virtual Machine (віртуальна машина Java), Android, JavaScript та Native. Дана мова була розроблена компанією JetBrains як заміна мови Java. Розробка почалася у 2010

році і перший реліз відбувся у 2016. Kotlin розроблений під ліцензією Apache 2.0, має відкритий вихідний код та є доступним для безкоштовного використання. Дана мова використовує як об'єктно-орієнтовані, так і функціональні конструкції, що дозволяє вести гнучку розробку програмного забезпечення. У порівнянні з Java, Kotlin є більш лаконічним, деякі оцінки показують скорочення кількості рядків коду приблизно на 40% [62], є типо-безпечним, має розумне приведення типів, функції вищого порядку, функції розширення та лямбда функції, які забезпечують можливість виразного запису коду, а також полегшують створення DSL. Також Kotlin повністю сумісний з Java і основний акцент при створення цієї мови робився на можливості використання кодової бази Java, тобто можна викликати Kotlin код з Java і навпаки. Також в IDE вбудований автоматизований перетворювач Java в Kotlin, який спрощує міграцію існуючого коду. Kotlin підтримується всіма основними інтегрованими середовищами розробки, які підтримують Java, а саме IntelliJ IDEA, Android Studio, Eclipse та NetBeans. Також доступний компілятор з командного рядка, який забезпечує просту підтримку компіляції та запуску програм.

У якості середовища розробки було використано IntelliJ IDEA. Воно розробляється компанією JetBrains. Доступне у двох варіантах: Community Edition, що має ліцензію Apache 2.0, та комерційний варіант - Ultimate Edition. Для розробки даного проекту була використана версія Community Edition. Це перше середовище розробки для мови Java, що містило широкий набір інтегрованих інструментів для рефакторингу [63]. IntelliJ IDEA відрізняється від своїх аналогів легкістю у використанні, гнучкістю та інтуїтивно зрозумілим дизайном.

Ключові функціональні характеристики:

- розумне автодоповнення коду;
- аналіз потоку даних;
- автоматичний рефакторинг;
- ергономічне середовище, орієнтоване на розробку;
- всі часто використовувані функції мають гарячі клавіші;

- багатофункціональний дебагер;
- вбудований контроль версій;
- інтегровані інструменти побудови виконуваного файлу;
- вбудовані інструменти для тестування;
- вбудований декомпілятор [64].

Підтримуються наступні мови [64]:

- Java;
- Scala;
- Groovy;
- Kotlin;
- JavaScript;
- TypeScript;
- SQL.

IntelliJ IDEA можна запустити на Windows, Linux, macOS.

Сучасні програми мають потребу у графічному інтерфейсі користувача (GUI). Кожна із сучасних мов програмування надає безліч бібліотек для роботи зі стандартним набором елементів управління. Kotlin використовує ті ж самі бібліотеки, що і Java. Є дві найбільш поширені бібліотеки візуальних компонентів для створення графічного інтерфейсу користувача. Найбільш рання з них називається AWT. Вважається, що при її проектуванні був допущений ряд недоліків, внаслідок яких з нею досить складно працювати. Бібліотека Swing розроблена на базі AWT і замінює більшість її компонентів своїми, більш ретельно спроектованими і зручними [65].

3.3 Архітектура програмного забезпечення

Розглянемо модулі, з яких складається програмний комплекс, у таблицях 3.2 – 3.6 наведені класи модулів програмного забезпечення для розв’язання задачі маршрутизації безпілотних авіаційних систем та їх опис.

Схема структурна класів програмного забезпечення наведена у графічних матеріалах.

Таблиця 3.2 – Класи модулю для розв’язування задачі маршрутизації безпілотних авіаційних систем за допомогою оптимізації мурашиними колоніям.

Ім’я класу	**Опис**
AbstractTopology	Абстрактний клас, що містить базові функції отримання сусідів для заданої топології
HypercubeTopology	Клас, що містить реалізацію топології гіперкуб, а також перевіряє можливість її побудови на заданих даних
RingTopology	Клас, що містить реалізацію топології кільце
TorusTopology	Клас, що містить реалізацію топології решітка, а також перевіряє можливість її побудови на заданих даних
IslandDaemon	Клас, що відповідає за отримання мігрантів відповідно топології
IslandOptimization	Клас, що реалізує острівну модель, запускає паралельно задану кількість островів і проводить синхронізацію між ними
DefaultMMASOptimization	Клас, що реалізує макс-мінний алгоритм мурашиних колоній
FarsightedMMASOptimization	Клас, що реалізує диверсифіковану версію макс-мінного алгоритму мурашиних колоній

Продовження таблиці 3.2

Ім’я класу	**Опис**
AbstractRoute	Клас, що містить функції для побудови маршруту, а саме налаштування мурах, вибір наступного міста, перевірка коректності маршруту
BasicRoute	Клас, що визначає як відбувається обрахунок імовірностей при побудові маршруту для макс-мінного алгоритму мурашиних колоній
FarsightedRoute	Клас, що визначає як відбувається обрахунок імовірностей при побудові маршруту для диверсифікованого макс-мінного алгоритму мурашиних колоній
Ant	Клас, що представляє “мураху”, містить в собі номера міст, які були відвідані мурахою
City	Клас, що представляє місто, може бути як початковим депо, звичайним містом та кінцевим депо
Solution	Клас, що представляє розв'язок, тобто містить набір мурах і довжину маршруту
TaskSettings	Клас, що містить значення всіх параметрів алгоритму

Таблиця 3.3 – Класи модулю для розв’язування задачі маршрутизації безпілотних авіаційних систем за допомогою методу прямого перебору.

Ім’я класу	**Опис**
BruteforceSolver	Клас, що реалізує алгоритм повного перебору для розв’язування задачі маршрутизації безпілотних авіаційних систем
ListPartioner	Клас, що виконує розбиття заданої множини на всі можливі підмножини
ListUtils	Клас, що містить функції-утиліти для виконання операцій перестановок

Таблиця 3.4 – Класи модулю для розв’язування задачі маршрутизації безпілотних авіаційних систем за допомогою локального пошуку.

Ім’я класу	**Опис**
GreedySolver	Клас, що містить реалізацію жадібного алгоритму для задачі маршрутизації транспортних засобів
TabuSearchSolver	Клас, що містить реалізацію локального пошуку (табу-пошук) з використанням початкового розв'язку, що був отриманий за допомогою жадібного алгоритму
Node	Клас, що представляє місто, яке потрібно відвідати
Vehicle	Клас, що представляє транспортний засіб, містить маршрут

Таблиця 3.5 – Класи модулю для розв’язування задачі підбору параметрів для алгоритму оптимізації мурашиними колоніями.

Ім’я класу	**Опис**
RingGeneratorParameters	Клас, що представляє кільцевий генератор околів для алгоритму векторного спуску
RVM	Клас, що реалізує алгоритм векторного спуску для розв’язування задачі підбору параметрів для алгоритму оптимізації мурашиними колоніями

Таблиця 3.6 – Класи модулю для візуалізації точок (міст) на основі заданої матриці відстаней.

Ім’я класу	**Опис**
Circle	Клас, що реалізує коло заданого радіуса і містить методи перевірки перетину кіл
Point	Клас, що представляє собою точку у двовимірному просторі
ComparatorContext	Клас, що задає точність для порівняння чисел
DistMatrix	Клас, що перетворює матрицю відстаней у набір точок у двовимірному просторі
DoubleComparator	Клас, який виконує порівняння чисел з плаваючою комою

У таблиці 3.7 наведений опис функцій модулю для розв’язування задачі маршрутизації безпілотних авіаційних систем за допомогою оптимізації мурашиними колоніям.

Таблиця 3.7 – Опис основних функцій модулю для розв’язування задачі маршрутизації безпілотних авіаційних систем за допомогою оптимізації мурашиними колоніям

Ім’я класу	**Функція**	**Опис функції**
AbstractTopology	calculateNeighborhood	Розрахунок сусідів для заданої точки
	getNeighborhood	Отримання сусідів для заданої точки
	buildConnectionGraph	Розрахунок всіх зв’язків між точками у топології
IslandDaemon	getMigrants	Отримання списку найкращих “мурах” отриманих у ході фази міграції
IslandOptimization	start	Запуск острівної моделі
DefaultMMASOptimization	visualizePoints	Візуалізація точок і зв’язків між ними у двовимірному просторі і
	parseCities	Парсинг даних з файлу задачі
	startAntOptimization	Запуск алгоритму оптимізації мурашиними колоніям

Продовження таблиці 3.7

Ім’я класу	**Функція**	**Опис функції**
DefaultMMASOptimization	initMinMaxPheromone	Ініціалізація мінімального і максимального рівня феромону у феромонній матриці
	initializePheromones	Ініціалізація матриці феромонів
	runIterations	Запуск ітерацій алгоритму
	findBestIterationSolution	Знаходження найкращого розв’язку на поточній ітерації
	updatePheromones	Оновлення матриці феромонів (випаровування, відкладання феромону)
	updateMinAndMaxValues	Оновлення мінімального та максимального значення феромону для матриці феромонів
	restartCheck	Перевірка стагнації

Продовження таблиці 3.7

Ім’я класу	**Функція**	**Опис функції**
DefaultMMASOptimization	routeLength	Обрахунок довжини маршруту для вказаного розв’язку
	applyMigrant	Відкладання феромону на маршруті мігранта
AbstractRoute	setupAnts	Налаштування початкового стану мурах
	selectNextCity	Вибір наступного міста для відвідання мурахою
	moveAnts	Переміщення мурахи у наступне місто
	constructSolution	Побудова розв’язку
Ant	visitCity	Відвідання вказаного міста
	containsRoute	Перевірка чи вказане місто міститься у маршруті даної мурахи
	currentCity	Поточне місто, у якому знаходиться мураха
	printTrail	Виведення інформації про маршрут
	clear	Очищення даних про маршрут

3.4 Інструкція для користувача

На початку роботи користувач повинен завантажити файл з задачею. Файл подається у наступному форматі:

- назва задачі;
- кількість безпілотних літальних апаратів;
- кількість місць, які потрібно відвідати;
- кількість початкових депо;
- кількість кінцевих депо;
- матриця відстаней між місцями;
- номери початкових депо;
- номери кінцевих депо і їх ємність;
- тип задачі (симетрична або асиметрична).

На рисунку 3.1 зображений приклад задачі, що має на 10 місць, які потрібно відвідати, 2 безпілотні літальні апарати, 2 початкові депо і 2 кінцеві депо

```
NAME: test_task_10_2_2
VEHICLES: 2
CITY_COUNT: 10
START_DEPOT_COUNT: 2
END_DEPOT_COUNT: 2
EDGE_WEIGHT_SECTION
0.0 923.21 682.69 823.80 267.54 714.11 1047.44 799.32 410.99 869.45
923.21 0.0 497.87 583.46 1013.09 225.30 131.455 590.07 545.25 69.379
682.69 497.87 0.0 165.57 642.89 453.32 562.85 150.40 302.48 429.393
823.80 583.46 165.57 0.0 743.89 590.05 615.94 26.67 464.28 520.66
267.54 1013.09 642.89 43.89 0.0 836.01 1122.89 717.3 468.49 949.54
714.11 225.30 453.32 590.05 836.01 0.0 356.734 586.45 374.23 196.23
1047.45 131.45 562.85 615.94 1122.89 356.734 0.0 627.80 658.94 178.05
799.32 590.07 150.40 26.67 17 717.3 586.45 627.80 0.0 444.68 525.96
410.99 545.25 302.48 464.28 468.49 374.23 658.94 444.687 0.0 483.13
869.45 69.37 429.39 520.66 949.54 196.230 178.05 525.96 483.13 0.0
START_DEPOT_SECTION
0
1
END_DEPOT_SECTION
2 1
3 1
```

Рисунок 3.1 – Приклад коректного файлу задачі

Програмне забезпечення розроблене у вигляді десктопного додатку, його графічний інтерфейс зображений на рисунку 3.2.

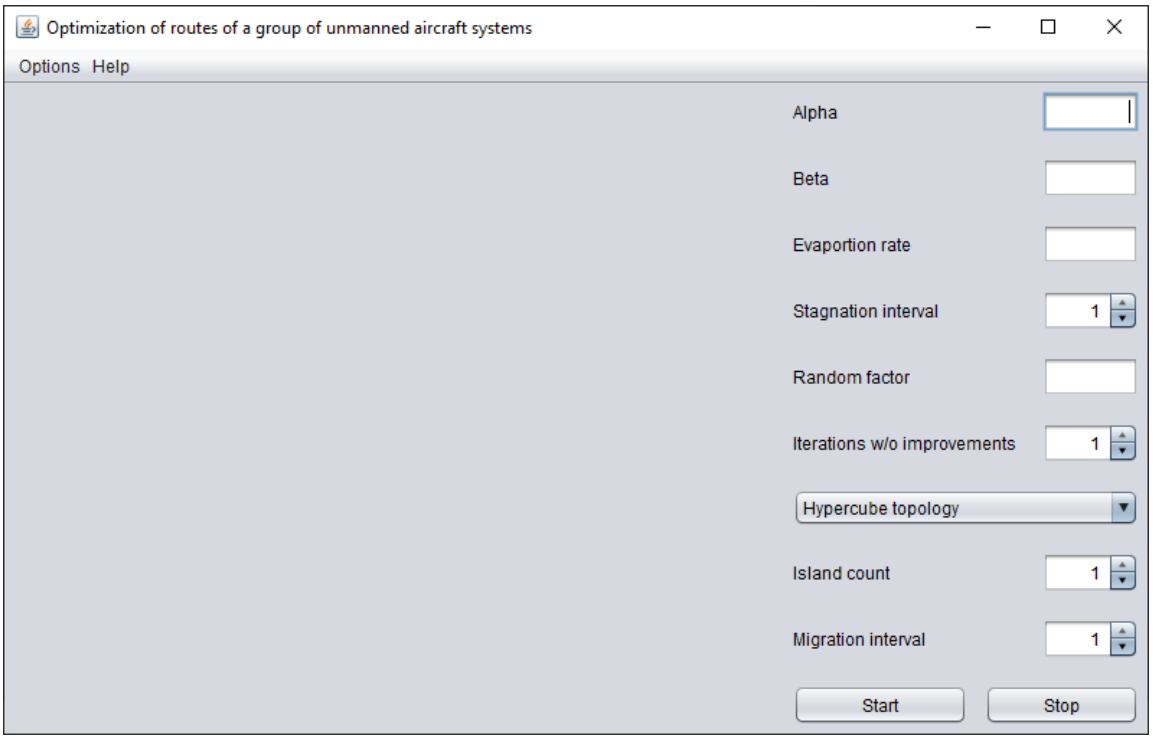


Рисунок 3.2 – Головний екран програмного забезпечення

Для початку роботи необхідно завантажити файл із задачею. Для цього потрібно натиснути меню «Options» і обрати пункт «Load task» як це зображено на рисунку 3.3

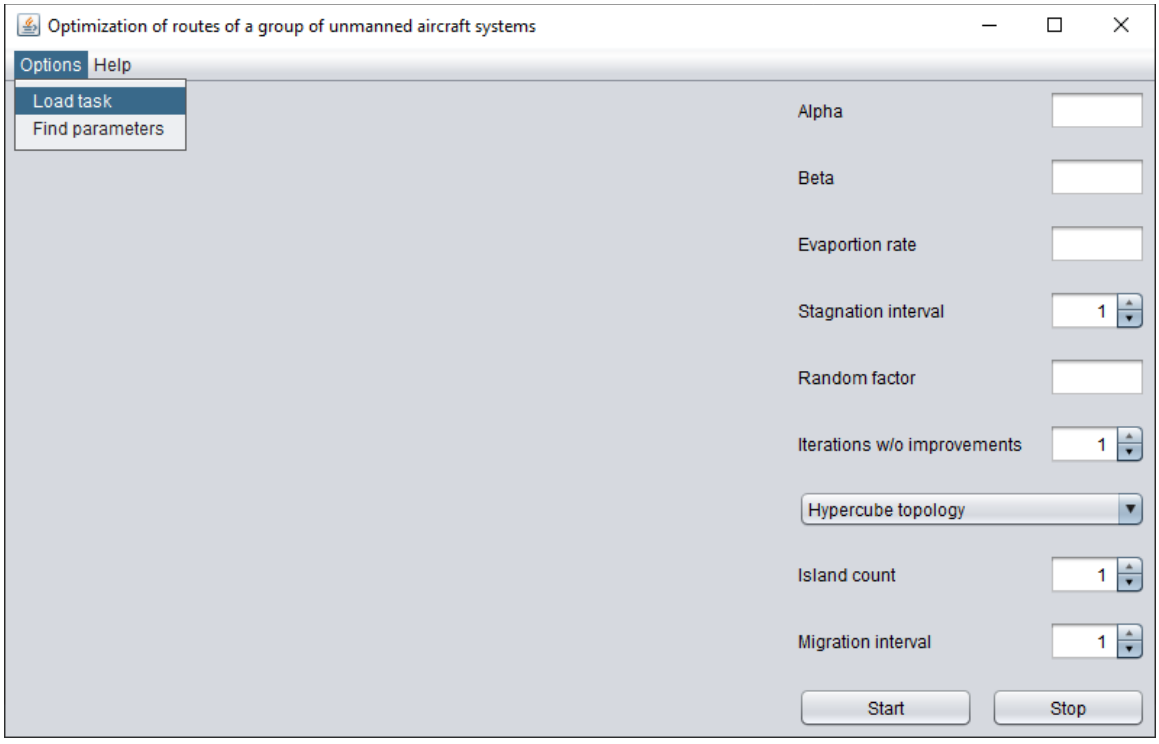


Рисунок 3.3 – Меню завантаження задачі

Після цього відкриється діалог для вибору файла задачі, який зображено на рисунку 3.4.

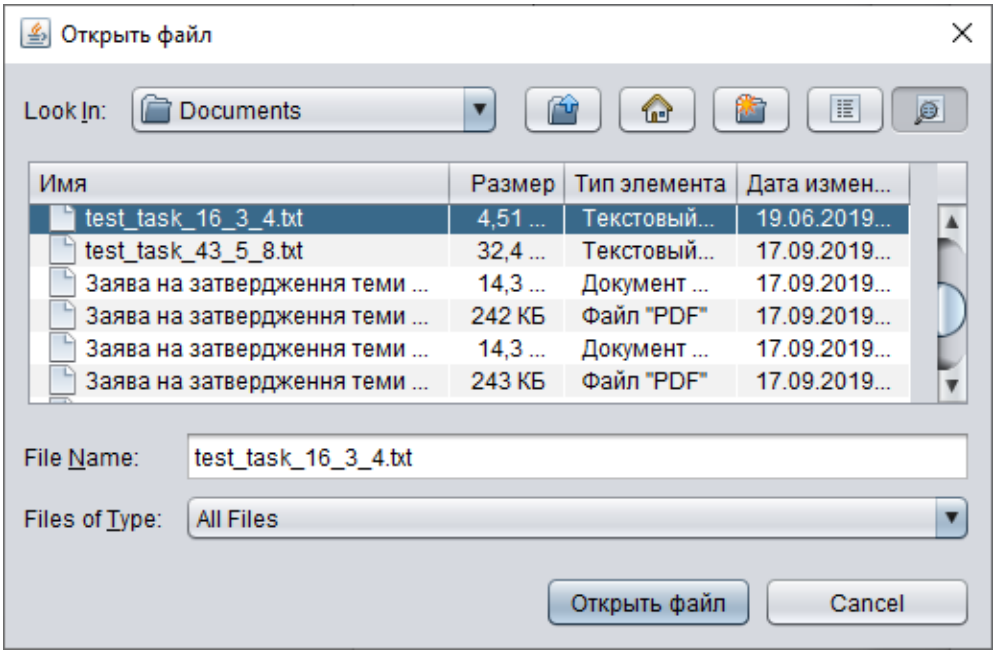


Рисунок 3.4 – Діалог з вибором файлу задачі

Далі необхідно ввести значення параметрів алгоритму розв’язування задачі маршрутизації безпілотних авіаційних систем за допомогою оптимізації мурашиними колоніям, секція з доступними параметрами зображена на рисунку 3.5.

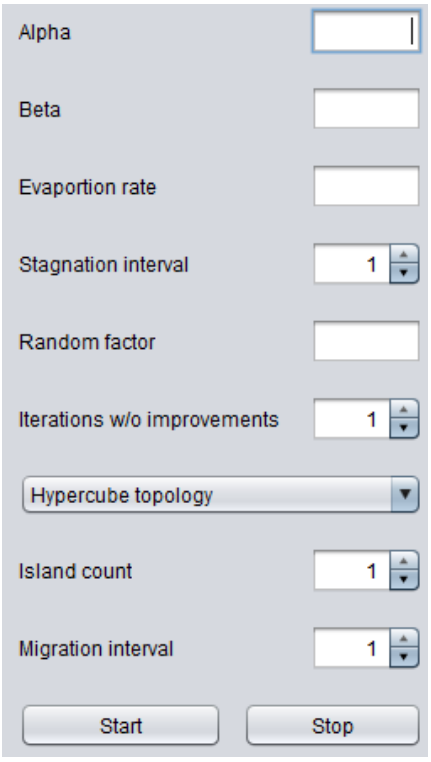


Рисунок 3.5 – Секція налаштування параметрів

Також користувач має змогу скористатися модулем для автоматичного підбору параметрів на основі задачі, яку він завантажить. Для цього необхідно завантажити задачу за раніше описаним алгоритмом і натиснути меню «Options» та обрати пункт «Find parameters» як це показано на рисунку 3.6.

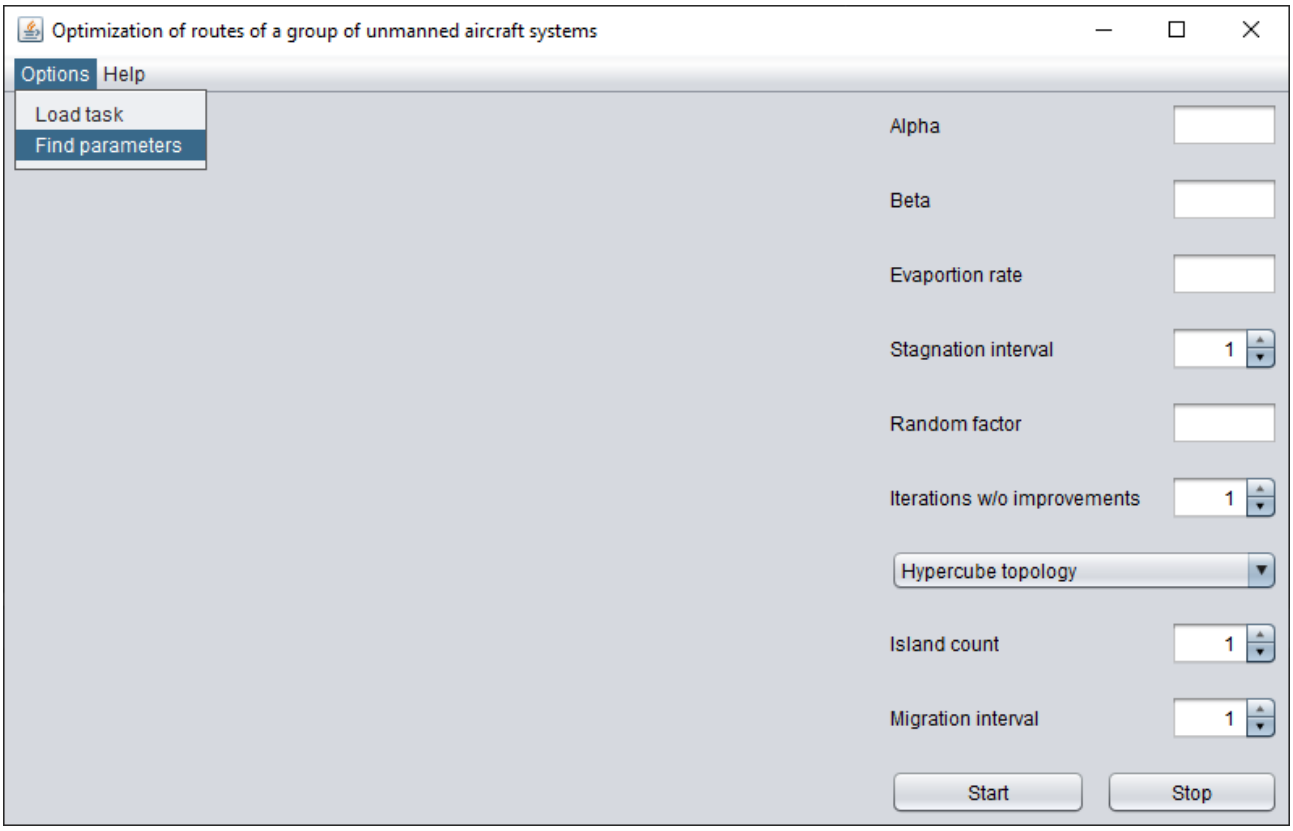


Рисунок 3.6 – Запуск алгоритму для автоматичного підбору параметрів

Після закінчення роботи алгоритму секція з параметрами буде заповнена найкращими значеннями параметрів, які були отримані у ході роботи алгоритмів. Секція з заповненими параметрами зображена на рисунку 3.7.

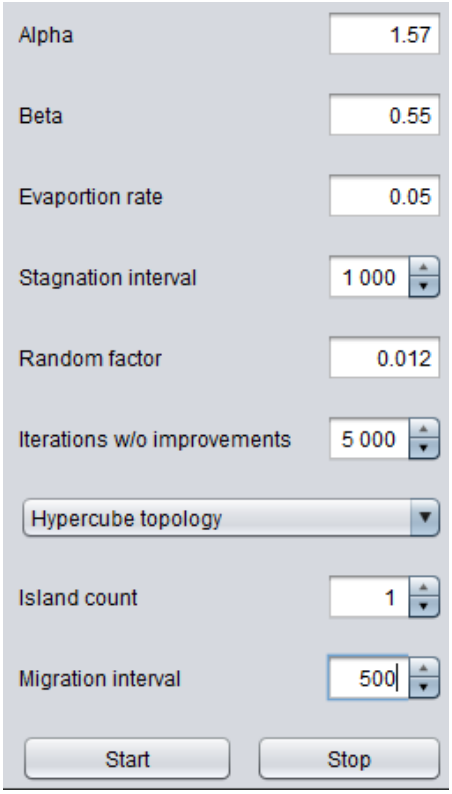


Рисунок 3.7 – Результат роботи модулю налаштування параметрів

Далі потрібно натиснути на кнопку «Start» для запуску розв’язування задачі маршрутизації безпілотних авіаційних систем за допомогою оптимізації мурашиними колоніям. Кнопка «Stop» зупиняє роботу алгоритму і звільняє ресурси комп’ютера. Після закінчення роботи алгоритму користувач отримає візуальне представлення маршруту, що зображене на рисунку 3.8.

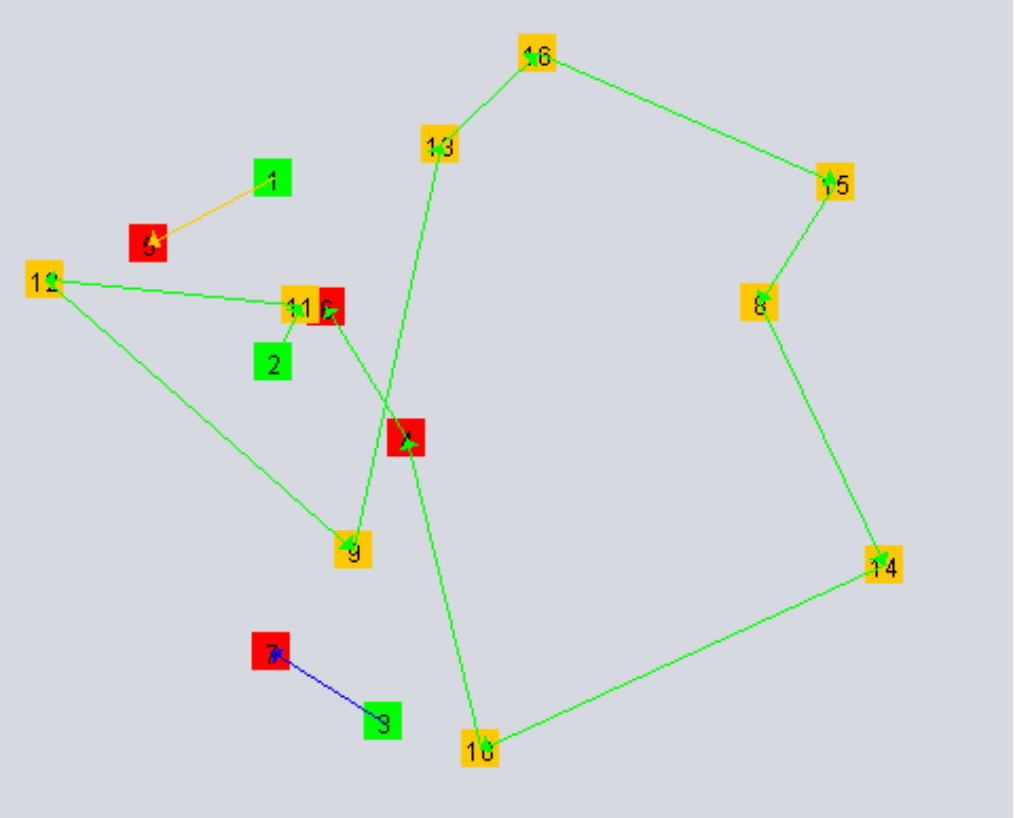


Рисунок 3.8 – Візуальне представлення маршруту

Також маршрут у текстовому вигляді буде збережений у файл на диску. Файл з результатом зображений на рисунку 3.9.

```
Current solution:
Ant #1:
0 -> 12 -> 15 -> 14 -> 7 -> 13 -> 8 -> 3 -> 7
Ant #2:
1 -> 10 -> 11 -> 4
Ant #3:
2 -> 9 -> 6
Length: 3438.5085007861985
```

Рисунок 3.9 – Текстове представлення розв’язку задачі

3.5 Опис технічного забезпечення

Оскільки для роботи програмного продукту потрібна віртуальна машина Java, то для саме її мінімальні системні вимоги будуть братися за основу. Для запуску віртуальної машини Java потрібен комп’ютер з наступними характеристиками:

- процесор з тактовою частотою 256 МГц;
- оперативна пам’ять 128 МБ;

- наявність клавіатури та мишки;
- ОС: Windows, Linux, macOS;
- наявність підключення до мережі Інтернет.

Висновки до розділу

Були розглянуті варіанти використання на основі яких були сформовані функціональні вимоги до програмного продукту. Враховуючи сформовані вимоги було створено схему структурну варіантів використання.

Описані основні засоби, які були використані для розробки даного програмного забезпечення. Детально було розглянуто переваги використання мови Kotlin замість Java.

Наведена інформація про структуру всіх програмних модулів, описане призначення класів. Також детально описані основні функції модулю для розв’язування задачі маршрутизації безпілотних авіаційних систем за допомогою оптимізації мурашиними колоніям.

Описані вимоги до технічного забезпечення для комп’ютера на якому буде запускатися програмне забезпечення.

4 РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТАЛЬНИХ ДОСЛІДЖЕНЬ

4.1 Опис проведення експериментів

План проведення експериментів:

- підготувати задачі, на яких будуть проводитися експерименти;
- підготувати початкові параметри, а також діапазон і крок їх зміни для алгоритму локального пошуку;
- підготувати початкові параметри, а також діапазон і крок їх зміни для макс-мін алгоритму мурашиних систем;
- запустити розв’язування заданої задачі із застосуванням покрокової зміни параметрів для визначення найкращого набору параметрів;
- провести серію запусків алгоритму локального пошуку для розв'язання відповідного набору задач;
- провести серію запусків макс-мін алгоритму мурашиних систем для розв'язання відповідного набору задач;
- провести серію запусків макс-мін алгоритму мурашиних систем із застосуванням острівної моделі для розв'язання відповідного набору задач;
- зробити висновки.

Оскільки у відкритому доступі немає даних для цієї задачі, то був згенерований набір задач. Згенеровані задачі мають розмірність від 10 до 70 об’єктів, які необхідно відвідати, у середньому 4-7 місць старту та 2-5 зон приймання.

Перейдемо до налаштування параметрів для локального пошуку.

Табу-пошук має наступні параметри:

- *horizon* – кількість ітерацій під час яких кожне переміщення може знаходитися у списку заборон;
- *listSize* – розмір списку заборон;
- *iterCondition* – кількість можливих ітерацій, на яких не відбувається покращення розв’язку, є критерієм завершенн роботи алгоритму.

Отже, застосуємо модуль автоматичного підбору параметрів (див. 2.9) для налаштування параметрів алгоритму табу-пошуку. Будемо змінювати лише перші два параметри, оскільки очевидно, що чим більший параметер *iterCondition*, тим довше працює алгоритм, але при цьому є більша імовірність отримати кращий розв'язок. Даний параметр буде мати таке ж значення, як і аналогічний параметр у макс-мін алгоритмі мурашиних систем.

Наведемо діапазони зміни значень параметрів та їх крок для табу-пошуку

- *horizon*: від 10 до 500 з кроком 10;
- *listSize*: від 1 до 20 з кроком 1.

Параметер *iterCondition* = 1000.

Після застосування модулю автоматичного підбору параметрів отримуємо наступні значення:

- *horizon* = 190;
- *listSize* = 8.

Перейдемо до налаштування параметрів для макс-мін алгоритму мурашиних систем. Він має наступні параметри:

- *antCount* – кількість мурах;
- α – ступінь значущості феромонного сліду;
- β – ступінь значущості евристичної інформації;
- *e* – стала випаровування;
- *iterCondition* – кількість можливих ітерацій, на яких не відбувається покращення розв'язку;
- *iterStagnation* – кількість ітерацій стагнації;
- *solutionsPerIteration* – кількість побудованих розв'язків однією мурахою за ітерацію;
- *randomFactor* – імовірність відвідування міста без застосування правилу рулетки.

Параметр *antCount* завжди дорівнює кількості визначених задачею БПЛА, параметер *iterCondition* = 1000 як і для локального пошуку. Наведемо діапазони зміни значень параметрів та їх крок:

- α – від 0.1 до 2 з кроком 0.01;

- β – від 0.1 до 2 з кроком 0.01;
- e – від 0.05 до 0.3 з кроком 0.01;
- *iterStagnation* – від 100 до 1000 з кроком 50;
- *solutionsPerIteration* – від 2 до 10 з кроком 1;
- *randomFactor* – від 0.01 до 0.1 з кроком 0.001.

Після застосування модулю автоматичного підбору параметрів отримуємо наступні значення:

- α = 1.57;
- β = 0.55;
- e = 0.05;
- *iterStagnation* = 1000;
- *solutionsPerIteration* = 7;
- *randomFactor* = 0.012.

Також введемо додаткові позначення:

- n – кількість об’єктів, що необхідно відвідати;
- *MMAS* – макс-мінний алгоритм мурашиних систем;
- *IMMAS* – макс-мінний алгоритм мурашиних систем із застосуванням острівної моделі;
- *TabuSearch* – локальний пошук (табу-пошук) з використанням оператору переміщення;
- f – розв’язок задачі, отриманий алгоритмом;
- $q = \frac{f - f_*}{f_*} \cdot 100\%$ – відносна похибка (у відсотках);
- f^* – найкращий розв’язок задачі серед всіх, що отримані алгоритмами;
- t – час виконання алгоритму (у наносекундах).

Всі алгоритми реалізовані на одній і тій же програмній базі на мові Kotlin, пошук розв’язків здійснювався на персональному комп’ютері з 16 ГБ оперативної пам’яті та восьмиядерним процесором з тактовою частотою 3.6 ГГц. Це дозволяє порівнювати час роботи алгоритмів. Основні результати

наведено у табл. 4.1, 4.2. Жирним шрифтом виділені найкращі розв’язки між всіма алгоритмами.

4.2 Результати експериментів

Проведемо десять запусків вищезазначених алгоритмів для семи задач різної розмірності.

Спочатку розглянемо найкращі отримані значення (табл. 4.1).

Таблиця 4.1 – Найкращі значення отриманих розв’язків кожним алгоритмом за серію запусків

Задача	***n***	**TabuSearch**			**MMAS**			**IMMAS**		
		f	***q***	***t***	***f***	***q***	***t***	***f***	***q***	***t***
1	16	3657	26.9	2.3E6	**2880**	0	7.0E9	**2880**	0	8.9E9
2	30	3906	6.6	2.6E6	3770	2.9	4.5E10	**3662**	0	5.5E10
3	36	4962	16.6	6.8E6	4395	3.3	7.8E10	**4253**	0	8.6E10
4	43	4858	10.5	7.4E7	4529	3	1.5E11	**4396**	0	1.7E11
5	57	7035	13.7	1.4E7	6361	2.8	2.9E11	**6185**	0	3.2E11
6	65	8414	10.1	1.5E7	7925	3.7	8.5E11	**7641**	0	9.8E11
7	70	**7912**	0	1.5E7	8940	12.9	8.7E11	8179	3	1E12

Середні значення q для найкращих значень відображені у табл 4.2.

Таблиця 4.2 – Середні значення q для найкращих значень отриманих розв’язків кожним алгоритмом за серію запусків

Алгоритм	***Середнє значення q***
TabuSearch	12.057
MMAS	4.08
IMMAS	0.42

Далі розглянемо середні значення отриманих розв’язків для кожного алгоритму (табл. 4.3).

Таблиця 4.3 – Середні значення отриманих розв’язків кожним алгоритмом за серію запусків

Задача	***n***	**TabuSearch**			**MMAS**			**IMMAS**		
		f	***q***	***t***	***f***	***q***	***t***	***f***	***q***	***t***
1	16	3657	25.9	3.9E6	2949	1.6	6.0E9	**2903**	0	8.8E9
2	30	3906	2.8	2.6E6	3881	2.2	4.5E10	**3797**	0	5.5E10
3	36	4962	12.9	6.2E6	4601	4.7	7.8E10	**4393**	0	8.8E10
4	43	4858	5.4	4.3E7	4922	6.8	1.5E11	**4606**	0	1.7E11
5	57	7035	8.6	1.4E7	6774	4.6	2.9E11	**6472**	0	3.3E11
6	65	8414	5	1.5E7	8424	5.1	8.5E11	**8010**	0	9.8E11
7	70	**7912**	0	2E7	9084	14.8	8.7E11	8472	7	1E12

Середні значення q для середніх значень відображені у табл 4.4.

Таблиця 4.4 – Середні значення q для середніх значень отриманих розв’язків кожним алгоритмом за серію запусків

Алгоритм	***Середнє значення q***
TabuSearch	8.65
MMAS	5.68
IMMAS	1

Як бачимо, макс-мінний алгоритм мурашиних систем із застосуванням острівної моделі показує найкращі результати як у знаходженні абсолютно найкращого розв’язку, так і в усередненому. У порівнянні брав участь алгоритм з повною топологією, проведемо серію експериментів для визначення яка з топологій показує найкращі результати.

Аналогічно, спочатку розглянемо найкращі отримані значення при використанні різних топологій (табл. 4.5).

Таблиця 4.5 – Найкращі результати розв’язку задач при різних топологіях

Задача	***n***	**COMPLETE**		**HYPERCUBE**		**RING**		**TORUS**	
		f	***t***	***f***	***t***	***f***	***t***	***f***	***t***
1	16	**2880**	8.9E9	**2880**	9.0E9	**2880**	8.5E9	**2880**	8.3E9
2	30	3662	5.5E10	3646	5.0E10	**3617**	5.1E10	**3617**	5.3E10
3	36	4253	8.6E10	4379	9.0E10	**4184**	8.9E10	4237	8.6E10
4	43	4396	1.7E11	4447	1.7E11	**4236**	1.7E11	4537	1.7E11
5	57	6185	3.2E11	6277	3.3E11	**6077**	3.4E11	6147	3.3E11
6	65	**7641**	9.8E11	7734	9.8E11	7898	9.7E11	7872	9.7E11
7	70	**8179**	1E12	8204	1.0E12	8243	1.0E12	8189	1.1E12

Далі розглянемо середні значення отриманих розв’язків при використанні різних топологій (табл. 4.6).

Таблиця 4.6 – Середні значення отриманих розв’язків при різних топологіях

Задача	***n***	**COMPLETE**		**HYPERCUBE**		**RING**		**TORUS**	
		f	***t***	***f***	***t***	***f***	***t***	***f***	***t***
1	16	2903	8.8E9	2886	8.8E9	**2883**	8.6E9	**2883**	8.7E9
2	30	3797	5.5E10	**3731**	5.2E10	3761	5.2E10	3736	5.2E10
3	36	4393	8.8E10	4492	8.8E10	**4391**	8.8E10	4421	8.7E10
4	43	4606	1.7E11	4679	1.7E11	**4551**	1.7E11	4670	1.7E11
5	57	6472	3.3E11	6466	3.3E11	6417	3.3E11	**6329**	3.3E11
6	65	**8010**	9.8E11	8057	9.9E11	8132	9.8E11	8057	9.8E11
7	70	**8472**	1E12	8568	1E12	8589	1E12	8537	1E12

Візуалізуємо розв’язки отримані для всієї серії випробувань (рисунок 4.1).

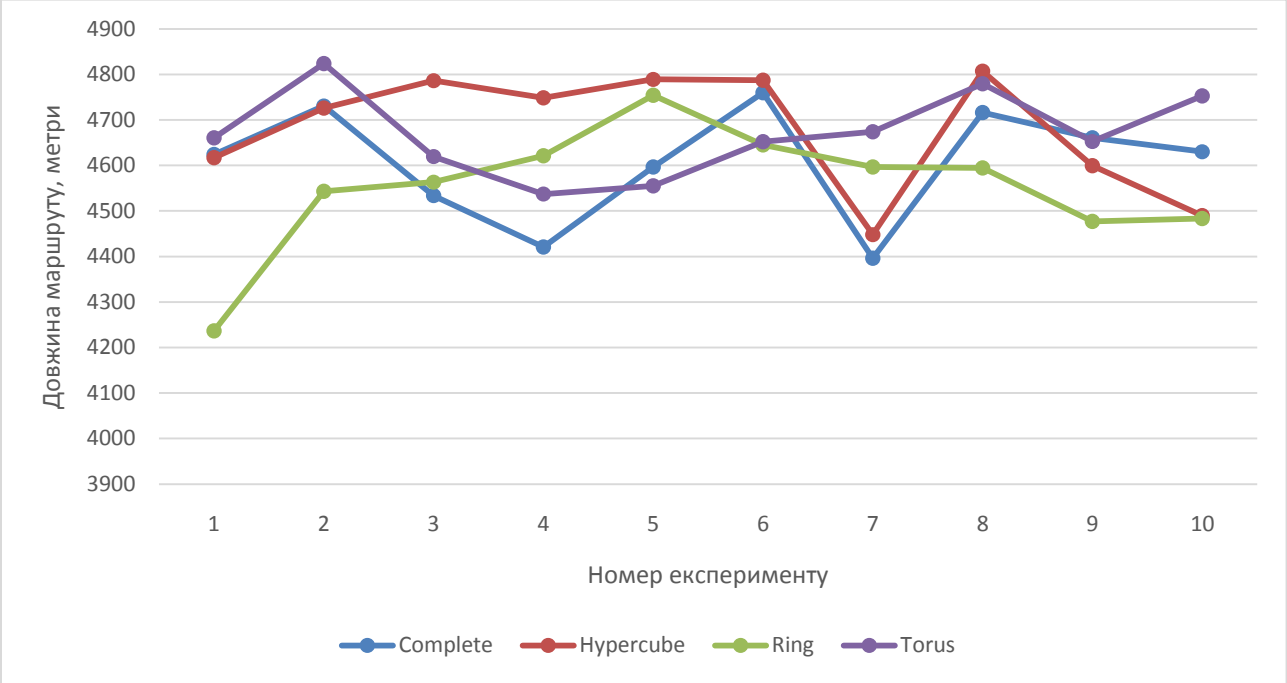


Рисунок 4.1 – Найкращі знайдені розв'язки для серії запусків з різними топологіями

Якщо брати до уваги середні значення, то яскраво вираженого лідеру немає, найкраще себе показує кільцева топологія, а найгірше себе показала топологія гіперкуб. Якщо брати до уваги найкращий отриманий розв’язок, то кільцева топологія виграє з великим відривом, гіперкуб при цьому як і в попередньому випадку показує найгірші результати. Час виконання для всіх топологій майже однаковий, тому можна не брати його до уваги при порівнянні.

Висновки до розділу

Для перевірки ефективності розв’язування задачі маршрутизації безпілотних авіаційних систем проведено серію обчислювальних експериментів із застосуванням запропонованих і реалізованих алгоритмів. Для експериментів були використані згенеровані випадковим чином задачі.

На основі проведених експериментів можна зробити висновки:

- серед наведених трьох алгоритмів, які використані в експериментах, беззаперечним лідером є макс-мін алгоритм мурашиних систем із застосуванням острівної моделі;
- топологія острівною моделі має вплив на знаходження розв’язку. Найкраще себе зарекомендувала топологія кільце, але при цьому вона поступилася повній топології на двох останніх задачах найбільшої розмірності, тому можна зробити висновок, що кожна топологія має свої переваги і недоліки, тому потрібно підбирати її індивідуально під задачу.

5 РОЗРОБКА СТАРТАП-ПРОЕКТУ

5.1 Опис ідеї проекту

Основною ідеєю проекту є надання програмного забезпечення для розв’язування задачі маршрутизації безпілотних авіаційних систем з використанням рухомих платформ.

Узагальнену опис ідей стартап-проекту наводиться в таблиці 5.1.

Таблиця 5.1 – Опис ідеї стартап-проекту

Зміст ідеї	*Напрямки застосування*	*Вигоди для користувача*
Надання сервісу для розв’язування розв’язування задачі маршрутизації безпілотних авіаційних систем з використанням рухомих платформ	1. Пошук оптимальних маршрутів для групи спеціальних літальних апаратів, зокрема, безпілотних літальних апаратів для проведення розвідки.	Економія заряду живлення літальних апаратів та зменшення часу перебування літальних апаратів у небезпечних зона, а також зниження ймовірності розкриття місця старту БПЛА.
	2. Оптимізація перевезень на маршрутах, де початкові і кінцеві депо різняться.	Економія пального для транспортних засобів, економія людино-годин

Далі потрібно визначити потенційних техніко-економічних переваг ідеї порівняно із пропозиціями конкурентів. Для цього сформуємо переліку техніко-економічних властивостей та характеристик ідеї. До цього переліку входять:

- вартість;
- швидкодія;
- зручність інтерфейсу;
- задоволення специфіці військових задач;
- складність в експлуатації;
- можливість роботи без інтернету;
- надійність;
- швидкість розгортання;
- унікальність алгоритму;
- документованість;
- функціонал.

Розглянемо продукти-аналоги, з якими будемо далі проводити порівняння. У відкритому доступі немає аналогів, що дозволяють розв'язувати задачу маршрутизації БАС, яка має альтернативний вибір депо і не передбачає прив’язку літальних апаратів до конкретного депо, тому порівняння будемо проводити з програмними комплексами, що дають можливість розв’язувати класичну задачу маршрутизації транспортних засобів.

TransTrade. Спеціалізоване програмне забезпечення для транспортної логістики. Воно дозволяє вирішувати повсякденні завдання логістичних компаній, діяльність яких пов'язана з перевезеннями, і виробників, що мають власний автопарк. Ефективна автоматизація транспортних підприємств дозволяє налагодити процес надання транспортних послуг, оптимізувати витрати і скоротити помилки, викликані людським фактором. Програма TransTrade ідеально підходить для цих цілей. Крім базових можливостей, для програми доступні додаткові модулі, що дозволяють розширити функціональні можливості, виходячи зі специфіки роботи і вимог вашої компанії.

Особливості та переваги:

- гнучкі налаштування під індивідуальні потреби кожного клієнта;
- широкий функціонал, що дозволяє істотно прискорити і спростити велику кількість стандартних операцій і рутинних дій;
- оперативна і продуктивна взаємодія всіх співробітників в єдиній системі: від менеджера-логіста до бухгалтера і керівника;
- зручний інтуїтивно зрозумілий для будь-якого користувача інтерфейс.
- швидке впровадження за 1 день.

Мегалогіст. Програма для управління доставкою замовлень в 1С, ефективно вирішує комплекс транспортних завдань:

- планування рейсів, розподіл завдань по водіях;
- відправка завдань водіям на мобільний додаток;
- моніторинг і диспетчеризація доставки в онлайн режимі;
- контроль витрат на доставку і розрахунок собівартості;
- взаєморозрахунки з водіями, контроль боргів і недостач;
- звітність по доставці для логістів та керівників.

Програмне забезпечення вбудовується у програму 1С клієнта і дозволяє логіста і водіям працювати в єдиному середовищі. Містить інтеграцію з модулем планування маршрутів magenta technology, що дозволяє:

- будувати оптимальний маршрут слідування для кожної з машин в автоматичному режимі;
- враховувати бажані вікна доставки;
- планувати маршрути по гео-зонам;
- враховувати вантажопідйомність і місткість машин, підбирати машини в залежності від замовлення (легкі / важкі);
- застосовувати різні стратегії оптимізації доставки (мінімізація пробігу / робочого часу / рівномірний розподіл робіт по автомобілях).

ИАС Грузоперевозки. Програма Грузоперевозки призначена для автоматизації вантажоперевезень, логістики. Програма повністю автоматизує роботу транспортної компанії. Програма надає безліч інструментів для роботи з вантажами, угодами, платежами, контрагентами, транспортом, причепами, водіями, поштою, паливом тощо. Засоби отримання звітної документації дозволяють переглянути та роздрукувати оперативні дані по транспортному обліку, оплаті, боргам і багато іншого.

У таблиці 5.2 наведено результати аналізу характеристик ідеї проекту.

Таблиця 5.2 - Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	*Техніко-економічні характеристики ідеї*	*(потенційні) товари/концепції конкурентів*				*W (слабка сторона)*	*N (нейтральна сторона)*	*S (сильна сторона)*
		Мій проект	TransTrade	*Мегалогіст*	*ИАС Грузоперевозки*			
1.	Вартість	Безкоштовний, платними є замовлення на модернізацію під певні вимоги	2500 грн	В залежності від об’єму необхідних задач, SAAS модель	40000 грн, є можливість взяття програми у аренду за 5000 на 6 місяців			+

Продовження таблиці 5.2

2.	Швидкодія	Середня	Нижче середньої	Середня	Середня		+	
3.	Зручність інтерфейсу	Висока	Висока	Середня	Середня			+
4.	Задоволення специфіці військових задач	Задовольняє	Не задовольняє	Не задовольняє	Не задовольняє			+
5.	Складність в експлуатації	Інтуїтивно зрозумілий	Високий поріг входження	Високий поріг входження	Високий поріг входження			+
6.	Можливість роботи без Інтернету	Є	Є	Нема	Є		+	
7.	Надійність	Висока	Середня	Середня	Середня			+
8.	Швидкість розгортання	Висока	Середня	Низька	Середня			+
9.	Унікальність алгоритму	Є	Невідомо	Невідомо	Невідомо		+	
10.	Документованість	Часткова	Повна	Повна	Повна	+		
11.	Функціонал	Середній, вузькоспеціалізований	Широкий	Широкий	Широкий	+		

Отже, програмні продукти, що існують на ринку, містять досить широкий функціонал для вирішення багатьох задач логістики, але через це вони містять перевантажений інтерфейс, низьку швидкість розгортання, а також середню надійність. Перевагами мого продукту є його ціна, відсутність складності в експлуатації, а також направленість для вирішення військових задач.

5.2 Технологічний аудит ідеї проекту

В межах даного підрозділу проведемо аудит технології, за допомогою якої можна реалізувати ідею проекту (технології створення товару).

Визначення технологічної здійсненності ідеї проекту передбачає аналіз таких складових (табл. 5.3):

− за якою технологією буде виготовлено товар згідно ідеї проекту;

- чи існують такі технології, чи їх потрібно розробити/доробити;
- чи доступні такі технології авторам проекту.

Таблиця 5.3 – Технологічна здійсненність ідеї проекту

№ п/п	*Ідея проекту*	*Технології її реалізації*	*Наявність технологій*	*Доступність технологій*
1	Візуалізація маршруту	Бібліотека RPlot	Наявна	Доступні
2	Зберігання маршруту	Структура даних	Необхідно розробити	Доступна
3	Використання алгоритмів розробки оптимальних маршрутів	Алгоритм маршрутизації	Необхідно розробити	Доступна
4	Використання алгоритмів локального покращення результату	Алгоритми локального пошуку	Необхідно розробити	Доступна
Обрана технологія реалізації ідеї проекту: з огляду на те що всі технології для реалізації ідей доступні, то ми можемо реалізувати всі заплановані ідеї, три з чотирьох технологій необхідно розробити, але є розуміння які сімейства алгоритмів треба взяти за основу для створення свого.				

5.3 Аналіз ринкових можливостей запуску стартап-проекту

Розглянемо попередню характеристику потенційного ринку стартап-проекту (табл. 5.4)

Таблиця 5.4 – Попередня характеристика потенційного ринку стартап-проекту

№ п/п	*Показники стану ринку (найменування)*	*Характеристика*
1	Кількість головних гравців, од	2
2	Загальний обсяг продаж, грн/ум.од	Невідомий
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Юридичні обмеження
5	Специфічні вимоги до стандартизації та сертифікації	Згідно військовим вимогам до програмного забезпечення

Продовження таблиці 5.4

6	Середня норма рентабельності в галузі (або по ринку), %	40%

Ринок програмного забезпечення для прокладання маршрутів для безпілотних літальних апаратів є достатньо привабливим, оскільки він має небагато головних гравців і також вони лише частково задовольняють військовим вимогам, оскільки в першу чергу орієнтуються на цивільний сектор. Тому існує потреба у створенні програмного продукту для маршрутизації літальних апаратів у військових цілях.

Надалі визначимо потенційні групи клієнтів, їх характеристики, та сформуємо орієнтовний перелік вимог до товару для кожної групи (табл. 5.5).

Таблиця 5.5 – Характеристика потенційних клієнтів стартап-проекту

№ п/п	*Потреба, що формує ринок*	*Цільова аудиторія (цільові сегменти ринку)*	*Відмінності у поведінці різних потенційних цільових груп клієнтів*	*Вимоги споживачів до товару*
1	Пошук оптимальних маршрутів для групи спеціальних літальних апаратів	Військові логісти	Військовий регламент щодо програмного забезпечення (безпека даних, ізольованість від інших процесів)	Безпека даних, інтуїтивна зрозумілість,

Після наведення потенційних клієнтів сформуємо список факторів, що сприяють ринковому впровадженню проекту, та факторів, що йому перешкоджають. Результати сформованих факторів наведені у таблицях 5.6 та 5.7.

Таблиця 5.6 – Фактори загроз

№ п/п	*Фактор*	*Зміст загрози*	*Можлива реакція компанії*
1.	Військовий регламент	Відповідність військовому регламенту	Уточнити у військових
2.	Розвиток продукту	В якому напрямку слід рухати проект	Уточнити у військових
3.	Задоволення потреб користувачів	Який набір функціоналу необхідний користувачу	Уточнити у військових

Таблиця 5.7 – Фактори можливостей

№ п/п	*Фактор*	*Зміст можливості*	*Можлива реакція компанії*
1.	Зацікавлення представників з інших сфер діяльності	Адаптування програми під інші сфери діяльності	Можливість покращити логістику інших сфер діяльності
2.	Відсутність аналогів для військового сектору	В Україні не має аналогів такого програмного продукту	Можливість стати монополістом в даній галузі

Проведемо аналіз пропозиції: визначимо загальні риси конкуренції на ринку. Результати аналізу наводяться у таблиці 5.8.

Таблиця 5.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	*В чому проявляється дана характеристика*	*Вплив на діяльність підприємства*
1. Вказати тип конкуренції Монополістична	Існують схожі програмні засоби, їх небагато і вони широко направлені, досить складні у налаштуванні для задоволення військових потреб	Зробити акцент на базовому функціоналі, не перевантажувати його
2. За рівнем конкурентної боротьби Національний	Багато держав розробляють програми щодо покращення військової логістики.	Запропонувати прийнятну цінову політику, зробити базовий функціонал безкоштовним, а модифікації - платними
3. За галузевою ознакою Внутрішньогалузева	Боротьба серед аналогів програм за клієнта	Необхідно створити кращий товар ніж у конкурентів
4. Конкуренція за видами товарів: Товарно-видова	Існують схожі програми, але не для військового сектору	Створити програмний продукт у відповідності вимог військового сектора
5. За характером конкурентних переваг Нецінова	Програмне забезпечення буде суттєво відрізнятися від звичайних логістичних програм	Необхідно вивчити вимоги до роботи у військовому середовищі
6. За інтенсивністю Не марочна	Фірма нова, невідома в Україні	Необхідно зарекомендувати себе за допомогою якісного продукту, а також співвідношення ціна-якість

Після аналізу конкуренції проведемо більш детальний аналіз умов конкуренції в галузі (за моделлю 5 сил М. Портера) (табл. 5.9).

Таблиця 5.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	*Прямі конкуренти в галузі*	*Потенційні конкуренти*	*Постачальники*	*Клієнти*	*Товари-замінники*

	Мегалогіст, TransTrade	Юридичний аспект роботи у військовому секторі, патенти на продукти	Немає	Контроль якості, наявність документації про продукт	Є часткова заміна
Висновки	Висока інтенсивність	Є потенційні конкуренти, а також можливість входу в ринок, але вона ускладнюється багатьма бюрократичними процесами	Немає постачальників	Встановлюють вимоги до безпеки, якості, швидкодії	Товари-замінники частково перекривають функціонал мого програмного продукту за рахунок складних налаштувань, але це не є їх основним функціоналом і він погано розкритий

У результаті аналізу таблиці 5.8 робимо висновок, що для того щоб впевнено увійти на ринку нам необхідно цілком і повністю задовольнити побажання військових логістів та розробити достатньо якісний програмний продукт, який є кращим за його аналоги. На основі аналізу конкуренції, проведеного викладеного в табл. 5.9, а також із урахуванням характеристик ідеї проекту (табл. 5.2), вимог споживачів до товару (табл. 5.5) та факторів маркетингового середовища (табл. 5.6-5.7) визначимо та обґрунтуємо перелік факторів конкурентоспроможності. Аналіз оформимо у вигляді табл. 5.10.

Таблиця 5.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	*Фактор конкурентоспроможності*	*Обґрунтування*
1.	Ціна	Ціна є одним з основних чинників конкуренції, оскільки для державних закупівель використовуються тендери, то продукт повинен мати низьку ціну для виграшу.

Продовження таблиці 5.10

2.	Легкість у користуванні	Даний фактор є важливим для швидкого впровадження, що є важливим у військовій сфері. Час на опанування програмного продукту військовими логістами повинен бути мінімальним
3.	Контроль якості	Якість і безвідмовна робота є першочерговими критеріями відбору програмного забезпечення у військовій сфері
4.	Задоволення потреб військових логістів	Особливо важливо щоб продукт задовольняв усі потреби, що мають військові логісти
5.	Репутація виробника	Якщо компанія має бездоганну репутацію, особливо у сфері якості своєї продукції, то рівень довіри до неї зростає. Також репутація виробника важлива при виході на ринок з новими товарами, або при виході на нові сегменти, що полегшує позитивне сприйняття новинок.

За визначеними факторами конкурентоспроможності (табл. 5.10) проведемо аналіз сильних та слабких сторін стартап-проекту (табл. 5.11).

Таблиця 5.11 – Порівняльний аналіз сильних та слабких сторін проекту

№ п/п	*Фактор конкурентоспроможності*	*Бали 1-20*	*Рейтинг товарів-конкурентів у порівнянні з моїм підприємством*						
			–3	*–2*	*–1*	*0*	*+1*	*+2*	*+3*
1.	*Ціна*	16	*						
2.	*Легкість у користуванні*	19		*					
3.	*Контроль якості*	17				*			
4.	*Задоволення потреб військових логістів*	20	*						
5.	*Репутація виробника*	10							*

Далі проведемо SWOT-аналіз стартапу. Результати аналізу представлені у табл. 5.12.

Таблиця 5.12 – SWOT-аналіз стартап-проекту

Сильні сторони: порівняно низька ціна, інтуїтивний інтерфейс і легкість в експлуатації, задоволення потреб військових логістів	Слабкі сторони: обмеженість у ресурсах та часі, відсутність досвіду у реалізації військових проектів, відсутність репутації у виробника
Можливості: зацікавлення представників з інших сфер діяльності, відсутність аналогів	Загрози: задоволення потреб клієнтів, агресивні дії великих конкурентів в галузі

На основі SWOT-аналізу розробимо альтернативи ринкової поведінки (перелік заходів) для виведення стартап-проекту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок (див. табл. 5.9, аналіз потенційних конкурентів).

Визначені альтернативи аналізуються з точки зору строків та ймовірності отримання ресурсів (табл. 5.13).

Таблиця 5.13 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	*Альтернатива (орієнтовний комплекс заходів) ринкової поведінки*	*Ймовірність отримання ресурсів*	*Строки реалізації*
1.	Стратегія спеціалізації	90%	1,5 роки
2.	Стратегія диференціації	50%	2 роки
3.	Позиціонування за співвідношенням "ціна - якість"	95%	1 рік
4.	Стратегія зайняття конкурентної ніші	50%	1 рік

Найкращим варіантом буде стратегія позиціонування за співвідношенням “ціна-якість”, оскільки ймовірність отримання ресурсів та строки реалізації є найкращими.

5.4 Розроблення ринкової стратегії проекту

Спочатку визначаємо опис цільових груп потенційних споживачів, який сформуємо у таблиці 5.14.

Таблиця 5.14 – Вибір цільових груп потенційних споживачів

№ п/п	*Опис профілю цільової групи потенційних клієнтів*	*Готовність споживачів сприйняти продукт*	*Орієнтовний попит в межах цільової групи (сегменту)*	*Інтенсивність конкуренції в сегменті*	*Простота входу у сегмент*
1.	Військові логісти	90%	100%	Середня	Середня
2.	Цивільні логісти	80%	50%	Висока	Низька
Які цільові групи обрано: військові логісти					

Оскільки проект працює з одним сегментом клієнтів, то використаємо стратегію концентрованого маркетингу (табл. 5.15).

Таблиця 5.15 – Визначення базової стратегії розвитку

№ п/п	*Обрана альтернатива*	*Стратегія охоплення ринку*	*Ключові конкурентоспроможні*	*Базова стратегія розвитку*

	розвитку проекту		*позиції відповідно до обраної альтернативи*	
1.	Зосередження на військовому сегменті	Стратегія концентрованого маркетингу	Високий рівень автоматизації, швидкодія	Стратегія спеціалізації

Далі визначимо базову стратегію конкурентної поведінки (табл. 5.16).

Таблиця 5.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	*Чи є проект «першопрохідцем» на ринку?*	*Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?*	*Чи буде компанія копіювати основні характеристики товару конкурента, і які?*	*Стратегія конкурентної поведінки*
1.	Так (на ринку України)	Так	Ні	Зайняття конкурентної ніші

На основі вимог споживачів з обраних сегментів до постачальника (стартап-компанії) та до продукту (див. табл. 5.5), а також в залежності від обраної базової стратегії розвитку (табл. 5.15) та стратегії конкурентної поведінки (табл. 5.16) розробимо стратегію позиціонування (табл. 5.17), що полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торговельну марку/проект.

Таблиця 5.17 – Визначення стратегії позиціонування

№ п/п	*Вимоги до товару цільової аудиторії*	*Базова стратегія розвитку*	*Ключові конкурентоспроможні позиції власного стартап-проекту*	*Вибір асоціацій, які мають сформувати комплексну позицію власного проекту*
1.	Пошук оптимальних маршрутів для групи спеціальних літальних апаратів	Задоволення потреб вибраного цільового сегменту краще, ніж конкуренти	Оптимальне співвідношення ціна-якість проекту; Задоволення військовим потребам	Доступна ціна, направленість на військові потреби, якість

5.5 Розроблення маркетингової програми стартап-проекту

Визначимо ключові переваги концепції потенційного товару (табл. 5.18.)

Таблиця 5.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	*Потреба*	*Вигода, яку пропонує товар*	*Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити*

1.	Пошук оптимальних маршрутів для групи спеціальних літальних апаратів	Економія заряду живлення літальних апаратів та зменшення часу перебування літальних апаратів у небезпечних зона, а також зниження ймовірності розкриття місця старту БПЛА	Співвідношення якість-ціна, урахування потреб військового сектору

Далі створимо трирівневу маркетингову модель товару (табл. 5.18).

Таблиця 5.18 – Опис трьох рівнів моделі товару

Рівні товару	*Сутність та складові*		
I. Товар за задумом	Програмний продукт для вирішення спеціальної задачі маршрутизації з альтернативними депо		
II. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	1. Якість 2. Простота у використанні 3. Видача оптимальних розкладів маршрутів 4. Низька ціна	-	-
	Якість: згідно до стандарту IEEE 829-2008 буде проведено тестування		
	Пакування нема		
	Марка (власна): Log-Studio		
III. Товар із підкріпленням	До продажу: тріал версія з обмеженням розмірності задачі		
	Після продажу: підтримка та додавання нового функціоналу по замовленню		
За рахунок чого потенційний товар буде захищено від копіювання: захист інтелектуальної власності, а також використання технології обфускації програмного коду			

Далі визначаємо межі встановлення ціни (табл. 5.19).

Таблиця 5.19 – Визначення меж встановлення ціни

№ п/п	*Рівень цін на товари-замінники*	*Рівень цін на товари-аналоги*	*Рівень доходів цільової групи споживачів*	*Верхня та нижня межі встановлення ціни на товар/послугу*
1	500$	5000$	2.5 млрд $	500-5000

Наступним кроком є визначення оптимального каналу збуту (табл. 5.20).

Таблиця 5.20 – Формування системи збуту

№ п/п	*Специфіка закупівельної поведінки цільових клієнтів*	*Функції збуту, які має виконувати постачальник товару*	*Глибина каналу збуту*	*Оптимальна система збуту*
1	Одна одиниця на військову частину	Домовлятися з військовими про вимоги та ціну	Перший рівень	Власні сили та через посередників

Останнім етапом є формування маркетингових комунікацій (табл. 5.21).

Таблиця 5.21 – Концепція маркетингових комунікацій

№ п/п	*Специфіка поведінки цільових клієнтів*	*Канали комунікацій, якими користуються цільові клієнти*	*Ключові позиції, обрані для позиціонування*	*Завдання рекламного повідомлення*	*Концепція рекламного звернення*
1	Орієнтовані на якість	Контент-маркетинг	Ціна, простота використання, кросплатформеність	Показати переваги продукту, низьку ціну	Демо-ролик з використанням, реклама.

Результатом підрозділу стала ринкова (маркетингова) програма, що включає в себе концепції товару, збуту, просування та попередній аналіз можливостей ціноутворення, спирається на цінності та потреби потенційних клієнтів, конкурентні переваги ідеї, стан та динаміку ринкового середовища, в межах якого впроваджено проект, та відповідну обрану альтернативу ринкової поведінки.

Висновки до розділу

Проведено аналіз програмного продукту у якості стартап проекту. Можна зазначити, що у проекту є можливість комерціалізації, оскільки ринок побудови маршрутів для військової сфери динамічно розвивається. На ринку наявна монополістична конкуренція, існує декілька фірм конкурентів, тому вихід на нього не буде легким. Через те, що проект є повністю програмним, його розробка не потребує витрат на різноманітні матеріали та обладнання. Можна сказати, що подальший розвиток проекту є доцільним, оскільки він має свою цільову аудиторію, також схожі проекти є дорогими і не орієнтуються на військову сферу.

ВИСНОВКИ

Проведено аналіз постановок та підходів до розв'язування задач маршрутизації груп БПЛА.

Запропоновано математичну модель задачі маршрутизації групи БАС.

Розроблено 4 прикладні алгоритми розв’язування задачі маршрутизації групи БАС, яка використовує альтернативні депо і не передбачає апріорну прив’язку літальних апаратів до конкретного депо.

Подано формулювання задачі пошуку оптимальних параметрів для алгоритмів, а також алгоритм для розв’язання цієї поставленої задачі.

Розроблено спеціалізований програмний комплекс для автоматичного налаштування параметрів алгоритмів та розв'язування сформульованої задачі.

Ефективність запропонованих алгоритмів досліджено шляхом аналізу результатів обчислювального експерименту, в якому диверсифікований макс-мін алгоритм мурашиних систем з використанням острівної моделі продемонстрував підвищену точність при розв’язуванні серії задач маршрутизації БАС.

За матеріалами дисертації було опубліковано 3 наукові роботи: 2 статті та 1 тези доповіді на конференції [26, 28, 66].

РЕКОМЕНДАЦІЇ

Напрямками подальшого дослідження є:

- використання популяцій з різними параметрами (ступінь значущості феромонного сліду, ступінь значущості евристичної інформації, стала випаровування, тощо) на островах;
- обмін феромонними матрицями між островами під час фази міграції;
- оптимізація кількості задіяних БПЛА;
- запуск декількох острівних систем з різними топологіями і обмін інформацією між системами;
- використання локального пошуку як демону в диверсифікованому макс-мін алгоритму мурашиних систем.

ПЕРЕЛІК ЛІТЕРАТУРИ

1. R. W. Harder, R. R. Hill, and J. T. Moore, “A Java universal vehicle router for routing unmanned aerial vehicles,” International Transactions in Operational Research, vol. 11, no. 3, pp. 259–275, 2004.

2. V. K. Shetty, M. Sudit, and R. Nagi, “Priority-based assignment and routing of a feet of unmanned combat aerial vehicles,” Computers & Operations Research, vol. 35, no. 6, pp. 1813–1828, 2008.

3. A. M. Ham, “Integrated scheduling of m-truck, m-drone, and m-depot constrained by time-window, drop-pickup, and mvisit using constraint programming,” Transportation Research Part C: Emerging Technologies, vol. 91, pp. 1–14, 2018.

4. Multi-uav resource constrained online monitoring of large-scale spatio-temporal environment with homing guarantee.

5. Avellar, G.S.C., Pereira, G.A.S., Pimenta, L.C.A., Iscold, P., 2015. Multi-UAV routing for area coverage and remote sensing with minimum time. Sensors 15 (11), 27783–27803.

6. Torres, M., Pelta, D. A., Verdegay, J. L., Coverage path planning with unmanned aerial vehicles for 3D terrain reconstruction. Expert Systems with Applications. 2016. Vol. 55. C. 441–451.

7. Guerriero, F., Surace, R., Loscr, V., Natalizio, E., 2014. A multi-objective approach for unmanned aerial vehicle routing problem with soft time windows constraints. Appl. Math. Model. 38 (3), 839–852.

8. Di Puglia Pugliese, L., Guerriero, F., Zorbas, D., Razafindralambo, T., 2016. Modelling the mobile target covering problem using flying drones. Optim. Lett. 10 (5), 1021–1052.

9. Ponda, S. S., Johnson, L. B., Geramifard, A. Cooperative mission planning for multi-UAV teams. In: Handbook of Unmanned Aerial Vehicles. Dordrecht: Springer. 2015. P. 1447-1490.

10. Liu, Y., Liu, Z., Shi, J. Optimization of Base Location and Patrol Routes for Unmanned Aerial Vehicles in Border Intelligence, Surveillance, and Reconnaissance. Journal of Advanced Transportation. 2019. Vol. 2019. C. 1–13.

11. G. S. C. Avellar, G. A. S. Pereira, L. C. A. Pimenta, and P. Iscold, "Multi-UAV routing for area coverage and remote sensing with minimum time," Sensors, vol. 15, no. 11, pp. 27783–27803, 2015.

12. Alotaibi, K.A., Rosenberger, J.M., Mattingly, S.P., Punugu, R.K., Visoldilokpun, S., 2018. Unmanned aerial vehicle routing in the presence of threats. Comput. Ind. Eng. 115, 190–205.

13. Ines Khouf "A Survey of Recent Extended Variants of the Traveling Salesman and Vehicle Routing Problems for Unmanned Aerial Vehicles".

14. Colnarič, M., Behnck, L. P., Doering, D., Pereira, C. E., and Rettberg, A., "A Modified Simulated Annealing Algorithm for SUAVs Path Planning," in 2nd IFAC Conference on Embedded Systems, Computer Intelligence and Telematics, Maribor, Slovenia, 2015, vol. 48, pp. 63–68.

15. Ergezer, H. and Leblebicioglu, K., "Path Planning for UAVs for Maximum Information Collection," IEEE Trans. Aerosp. Electron. Syst., vol. 49, no. 1, pp. 502–520, Jan. 2013.

16. Zhang, X. and Duan, H., "An improved constrained differential evolution algorithm for unmanned aerial vehicle global route planning," Appl. Soft Comput., vol. 26, pp. 270–284, Jan. 2015.

17. Yangguang Fu, Mingyue Ding, and Chengping Zhou, "Phase Angle-Encoded and Quantum-Behaved Particle Swarm Optimization Applied to ThreeDimensional Route Planning for UAV," Syst. Man Cybern. Part Syst. Hum. IEEE Trans. On, vol. 42, no. 2, pp. 511–526, 2012.

18. Cekmez, U., Ozsiginan, M., and Sahingoz, O. K., "A UAV path planning with parallel ACO algorithm on CUDA platform," in 2014 International Conference on Unmanned Aircraft Systems (ICUAS), 2014, pp. 347–354.

19. Xu, C., Duan, H., and Liu, F., "Chaotic artificial bee colony approach to Uninhabited Combat Air Vehicle (UCAV) path planning," Aerosp. Sci. Technol., vol. 14, no. 8, pp. 535–541, Dec. 2010.

20. Zhu, W. and Duan, H., “Chaotic predator–prey biogeography-based optimization approach for UCAV path planning,” Aerosp. Sci. Technol., vol. 32, no. 1, pp. 153–161, Jan. 2014.

21. Zhang, S., Zhou, Y., Li, Z., and Pan, W., “Grey wolf optimizer for unmanned combat aerial vehicle path planning,” Adv. Eng. Softw., vol. 99, pp. 121–136, Sep. 2016.

22. Gremlins [Електронний ресурс] – Режим доступу до ресурсу: https://www.darpa.mil/program/gremlins.

23. Dantzig, G. B., Ramser, J. H. The Truck Dispatching Problem. Management Science. 1959. Vol. 6, No. 1. P. 80–91.

24. Baldacci, R., Toth, P., Vigo, D. Exact algorithms for routing problems under vehicle capacity constraints. Annals of Operations Research. 2010. Vol. 175, No. 1. P. 213–245.

25. Гуляницкий Л.Ф. Проблема оптимизации маршрутов транспортных средств с временными окнами// Компьютерная математика. – 2007. – № 1. – С. 122–132.

26. Гуляницький Л. Ф., Сторчевий В. В. Одна спеціальна задача маршрутизації БПЛА // Науковий вісник Ужгородського університету. – 2019. – Випуск 34. – С.69-78.

27. Montoya-Torres, J. R., López Franco, J., Nieto Isaza, S. A literature review on the vehicle routing problem with multiple depots. Computers & Industrial Engineering. 2015. Vol. 79. P. 115–129.

28. Гуляницький Л. Ф., Сторчевий В. В. Оптимізація маршрутів групи БПЛА модифікованим алгоритмом мурашиних систем // Комп'ютерна математика. – 2019. – №1.

29. Гуляницький Л.Ф., Рибальченко О.В. Формалізація та розв'язування одного типу задач маршрутизації БПЛА // Теорія оптимальних рішень. – 2018. – 17. – с. 107-114.

30. Гуляницький Л.Ф., Мулеса О.Ю. Прикладні методи комбінаторної оптимізації: навч. посіб. – К.: Видавничо-поліграфічний центр "Київський університет", 2016. – 142 с.

31. A. Colorni and M. Dorigo and V. Maniezzo, An Investigation of Some Properties of an Ant Algorithm, Proceedings of PPSN-II, Second International Conference on Parallel Problem Solving from Nature, R. Manner and B. Manderick (Eds.), Elsevier, Amsterdam, The Netherlands, 509-520, 1992.

32. Eberhart, Russell C.. “Particle Swarm Optimization - Neural Networks, 1995. Proceedings., IEEE International Conference on.” (2004).

33. Camacho, Christian & Dorigo, Marco & Stützle, Thomas. (2019). The intelligent water drops algorithm: why it cannot be considered a novel algorithm: A brief discussion on the use of metaphors in optimization. Swarm Intelligence.

34. Cuevas, E., Miguel, C., Zaldívar, D., & Pérez-Cisneros, M. (2013). A swarm optimization algorithm inspired in the behavior of the social-spider. Expert Systems with Applications, 40(16), 6374–6384.

35. Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. Advances in Engineering Software, 95, 51–67.

36. Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. Advances in Engineering Software, 69, 46–61.

37. Askarzadeh, A. (2014). Bird mating optimizer: An optimization algorithm inspired by bird mating strategies. Communications in Nonlinear Science and Numerical Simulation, 19(4), 1213–1228.

38. Passino,K.M.(2002). Biomimicry of bacterial foraging for distributed optimization and control. IEEE Control Systems, 22(3), 52–67.

39. Yang, X.S.(2009). Firefly algorithms for multimodal optimization. In International symposium on stochastic algorithms (pp. 169–178). Springer.

40. Mehrabian, A. R., & Lucas, C. (2006). A novel numerical optimization algorithm inspired from weed colonization. Ecological informatics, 1(4), 355–366.

41. Birbil, ¸S. I., & Fang, S. C. (2003). An electromagnetism-like mechanism for global optimization. Journal of Global Optimization, 25(3), 263–282.

42. Hatamlou, A. (2013). Black hole: A new heuristic optimization approach for data clustering. Information Sciences, 222, 175–184.

43. Shah-Hosseini, H. (2007). Problem solving by intelligent water drops. In Proceedings of the 2007 congress on evolutionary computation (CEC 2007) (pp. 3226–3231). IEEE, IEEE Press, Piscataway, NJ.

44. Erol, O. K., & Eksin, I. (2006). A new optimization method: Big bang-big crunch. Advances in Engineering Software, 37(2), 106–111.

45. Kaveh, A., & Talatahari, S. (2010). A novel heuristic optimization method: Charged system search. Acta Mechanica, 213(3–4), 267–289.

46. Eskandar, H., Sadollah, A., Bahreininejad, A., & Hamdi, M. (2012). Water cycle algorithm-a novel metaheuristic optimization method for solving constrained engineering optimization problems.Computers& Structures, 110, 151–166.

47. Geem, Z. W., Kim, J. H., & Loganathan, G. V. (2001). A new heuristic optimization algorithm: Harmony search. Journal of Simulation, 76(2), 60–68.

48. Atashpaz-Gargari, E., & Lucas, C. (2007). Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition.In Proceedings of the 2007 congress on evolutionary computation (CEC 2007) (pp. 3226–3231).

49. Rao, R. V., Savsani, V. J., & Vakharia, D. (2011). Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems. Computer-Aided Design, 43(3), 303–315.

50. Ray, T., & Liew, K. M. (2003). Society and civilization: An optimization algorithm based on the simulation of social behavior. IEEE Transactions on Evolutionary Computation, 7(4), 386–396.

51. Liu D., Li S. Research on efficient online planning of emergency logistics path based on double-layer ant colony optimization algorithm // International Journal of Computers and Applications. – 2018. – 33. – p. 1-7.

52. Franco C., Aguilar H., Ochoa-Zezzatti A., Gallegos P. Comparison between instances to solve the CVRP // International Journal of Combinatorial Optimization Problems and Infor-matics. – 2018. – 9(2). – pp. 41-54.

53. Yu M., Li S., Kong M., Song J., Yang J., Ren G. Comparison of advantages and disad-vantages among various algorithms in logistics path design—

Taking H-group as an example // Cognitive Systems Research. – 2018. – 52. – pp. 843-852.

54. Kohl N. Exact methods for Time Constrained Routing and Related Scheduling Problems // PhD thesis, Department of Mathematical Modelling, Technical University of Denmark, 1995.

55. Soto M., Sevaux M., Rossi A., Reinholz A. Multiple neighborhood search, tabu search and ejection chains for the multi-depot open vehicle routing problem // Computers & Industrial Engineering. – 2017. – Volume 107. – pp. 211-222.

56. Dorigo M., Stützle T. Ant Colony Optimization: Overview and Recent Advances // Springer International Publishing AG. – 2019. – pp. 311-352.

57. Stützle T., Hoos H.H MAX-MIN ant system // Future Generation Computer Systems. – 2000. – pp. 889 - 914.

58. Гуляницький Л. Ф. Диверсифікація пошуку в алгоритмах оптимізації мурашиними колоніями / Л. Ф. Гуляницький // Теорія оптим. рішень : зб. наук. пр. - 2017. - Вип. 2017. - С. 47-57. - Бібліогр.: 11 назв. - укр.

59. Dorigo, M., Stutzle, T.: Ant Colony Optimization. MIT Press, Cambridge 2004.

60. La□ssig, J., Sudholt, D.: The benefit of migration in parallel evolutionary algorithms. In: Proceedings of GECCO 2010, pp. 1105–1112. ACM, New York (2010).

61. La□ssig, J., Sudholt, D: Experimental Supplements to the Theoretical Analysis of Migration in the Island Model. In: PPSN XI, Part I, LNCS 6238, pp. 224–233. Springer (2010).

62. FAQ – Kotlin Programming Language [Електронний ресурс] – Режим доступу до ресурсу: https://kotlinlang.org/docs/reference/faq.html.

63. Crossing Refactoring’s Rubicon [Електронний ресурс] – Режим доступу до ресурсу: https://www.martinfowler.com/articles/refactoringRubicon.html.

64. IntelliJ IDEA: The Java IDE for Professional Developers by JetBrains [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/idea/.

65. javax.swing (Java Platform SE 7) [Електронний ресурс] – Режим доступу до ресурсу: https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html.

66. Сторчевий В.В. Оптимізація маршрутів групи безпілотних авіаційних систем / В.В. Сторчевий, Л.Ф. Гуляницький // Матеріали III всеукраїнської науково-практичної конференції молодих вчених та студентів «Інформаційні системи та технології управління» (ІСТУ-2019) – м. Київ.: НТУУ «КПІ ім. Ігоря Сікорського», 20-22 листопада 2019 р. – С. 17-23.

ДОДАТОК А

Графічний матеріал

Псевдокод алгоритму табуйованого пошуку

procedure TABU_SEARCH_WITH_GREEDY (x);

ініціалізація_алгоритму;

$x :=$ припустимий варіант розв'язку, сформований жадібним алгоритмом;

$T := \varnothing$;

$x_{rec} := x$

while не_виконується_критерій_завершення **do**

пошук_прийнятного_варіанта $y \in O(x) \setminus T$;

$x := y$;

$T := T \cup x$;

if $f(x) < f(x_{rec})$ **then** $x_{rec} := x$ **;**

if $|T| >$ максимальний_розмір_списку_заборон **then**

видалення_з_множини_T_найдавніше_доданого_елемента;

endwhile

end

Демонстраційний плакат до магістерської дисертації

на тему «Оптимізація маршрутів групи безпілотних авіаційних систем»

Виконав студент гр. ІС-82мп Сторчевий Владислав Володимирович

Керівник Гуляницький Леонід Федорович

Псевдокод диверсифікованого макс-мін алгоритму мурашиних систем

```
procedure MMAS (x)
    ініціалізація_алоритму;
    while кількість_ітерацій_без_покращення_менша_заданої do
      for 1 to кількість_побудованих_розв'язків_за_ітерацію do
        формування_популяції_мурах;
        while не_всі_вершини_відвідані do
          foreach мураха_з_популяції do
            ініціалізація_поточного_кроку_мурахи;
            M = оновлення_пам'яті_мурахи;
            A = локальна_матриця_мурашиних_маршрутів;
            сформувати_множину_припустимих_вершин;
            p = обчислити_ймовірність_переходів(A, M, П);
            наступний_стан = правило_прийняття_рішення(p, П);
            вибравши вершину, перейти_в_наступний_стан;
            M = оновити_внутрішній_стан;
            вилучити вибрану вершину із списку припустимих;
          endfor
        endwhile
        запам'ятати_розв'язок;
      endfor
    завершити_діяльність;
    відкласти_феромон_на_найкращому_розв'язку_в_поточній_ітерації;
    випаровування_феромону;
    оновлення_матриці_феромонів;
    перевірка_стагнації;
    оновлення_рекорду (x);
    endwhile
end
```

ітерація

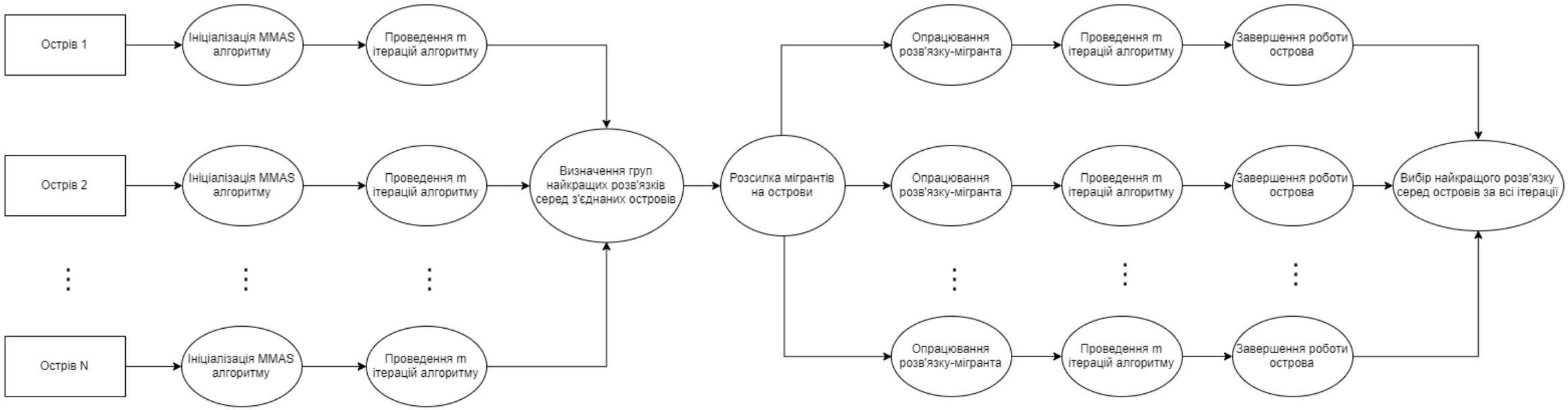
Демонстраційний плакат до магістерської дисертації

на тему «Оптимізація маршрутів групи безпілотних авіаційних систем»

Виконав студент гр. ІС-82мп Сторчевий Владислав Володимирович

Керівник Гуляницький Леонід Федорович

Блок-схема організації острівної моделі



Демонстраційний плакат до магістерської дисертації

на тему «Оптимізація маршрутів групи безпілотних авіаційних систем»

Виконав студент гр. ІС-82мп Сторчевий Владислав Володимирович

Керівник Гуляницький Леонід Федорович

Псевдокод острівної моделі диверсифікованого макс-мін алгоритму мурашиних систем

```
procedure ISLAND_MMAS (x)
    for (i = 0; i < N; i++)
        ініціалізація_алоритму;
        while кількість_ітерацій_без_покращення_менша_заданої do
        if настав_час_міграції then
            відправити_найкращий_глобальний_розв'язок_сусіднім_островам;
            if відправлене_значення_краще_за_поточне_на_острові then
            оновити_найкращий_розв'язок_на_острові;
            відкласти_феромон_на_даному_маршруті;
        else
            for 1 to кількість_побудованих_розв'язків_за_ітерацію do
              формування_популяції_мурах;
              while не_всі_вершини_відвідані do
                foreach мураха_з_популяції do
                  ініціалізація_поточного_кроку_мурахи;
                  M = оновлення_пам'яті_мурахи;
                  A = локальна_матриця_мурашиних_маршрутів;
                  сформувати_множину_припустимих_вершин;
                  p = обчислити_ймовірність_переходів(A, M, П);
                  наступний_стан = правило_прийняття_рішення(p, П);    I);
                  вибравши вершину, перейти_в_наступний_стан;
                  M = оновити_внутрішній_стан;
                  вилучити вибрану вершину із списку припустимих;
                endfor
              endwhile
              запам'ятати_розв'язок;
            endfor
            завершити_діяльність;
            відкласти_феромон_на_найкращому_розв'язку_в_поточній_ітерації;
            випаровування_феромону;
            оновлення_матриці_феромонів;
            перевірка_стагнації;
            оновлення_рекорду (x);
            endwhile
        endif
    end for
end
```

ітерація

Демонстраційний плакат до магістерської дисертації

на тему «Оптимізація маршрутів групи безпілотних авіаційних систем»

Виконав студент гр. ІС-82мп Сторчевий Владислав Володимирович

Керівник Гуляницький Леонід Федорович

Псевдокод підходу до налаштування параметрів алгоритму

procedure PARAMS_SEARCH_VRM (M, p_{start}, p_{min}, p_{max}, k, Δh);

$p_{current} := p_{start}$;

$l_{bestAvg} := \infty$;

$p_{best} := \varnothing$;

$r :=$ вектор розв'язків;

while (не_перебрані_всі_комбінації_ $p_{current}$) **do**

$p_{current} :=$ згенерувати_чергову_точку_околу;

for (i=0; i < k; i++) **do**

$r_i := M(x_{current})$;

endfor

знайти_l_{avg}_за_формулою_2.9;

if ($l_{bestAvg} > l_{avg}$) **then**

$l_{bestAvg} := l_{avg}$;

$p_{best} := p_{current}$;

endif

endwhile

return p_{best}

end

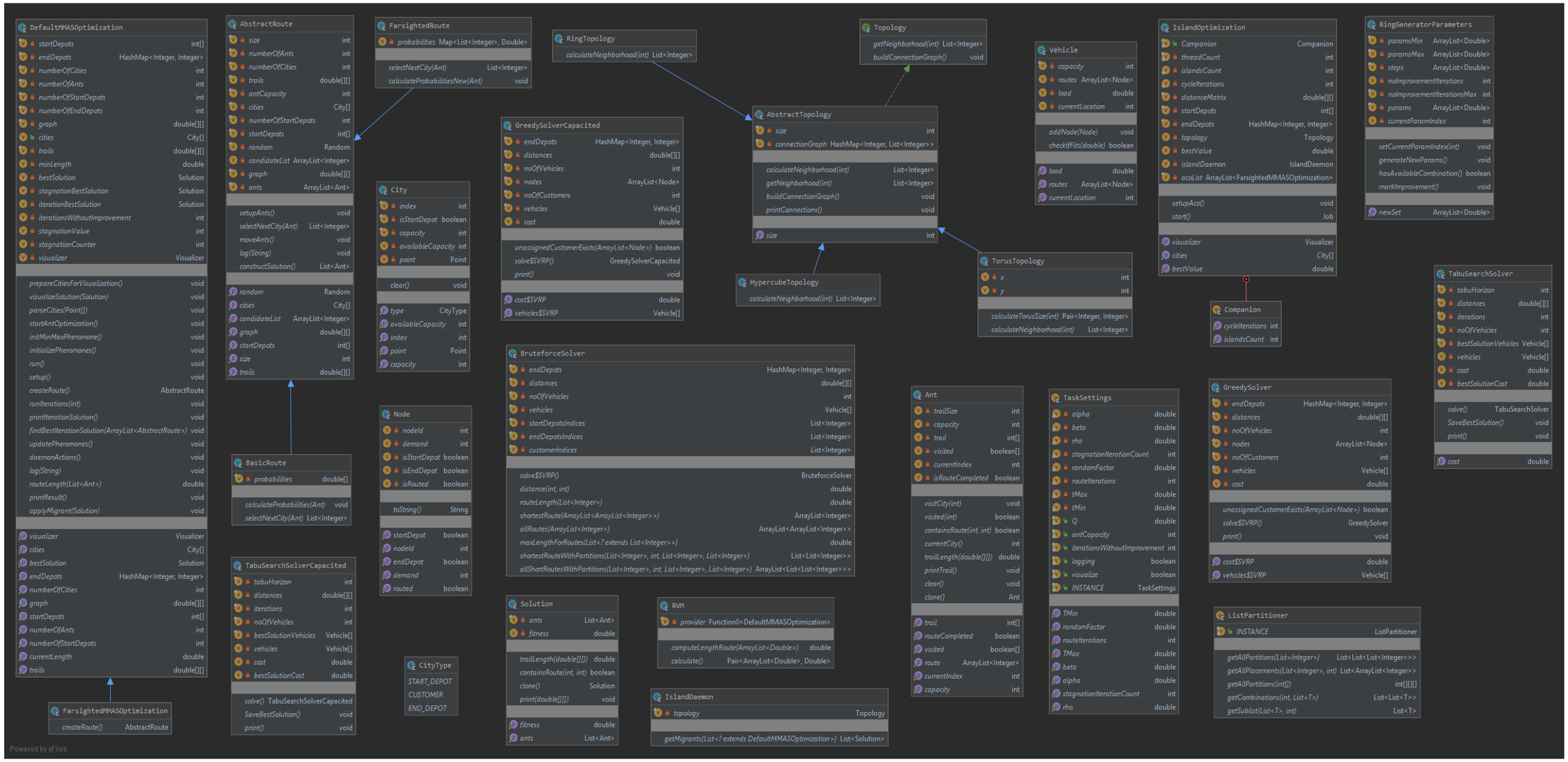
Демонстраційний плакат до магістерської дисертації

на тему «Оптимізація маршрутів групи безпілотних авіаційних систем»

Виконав студент гр. ІС-82мп Сторчевий Владислав Володимирович

Керівник Гуляницький Леонід Федорович

Схема структурна класів програмного забезпечення



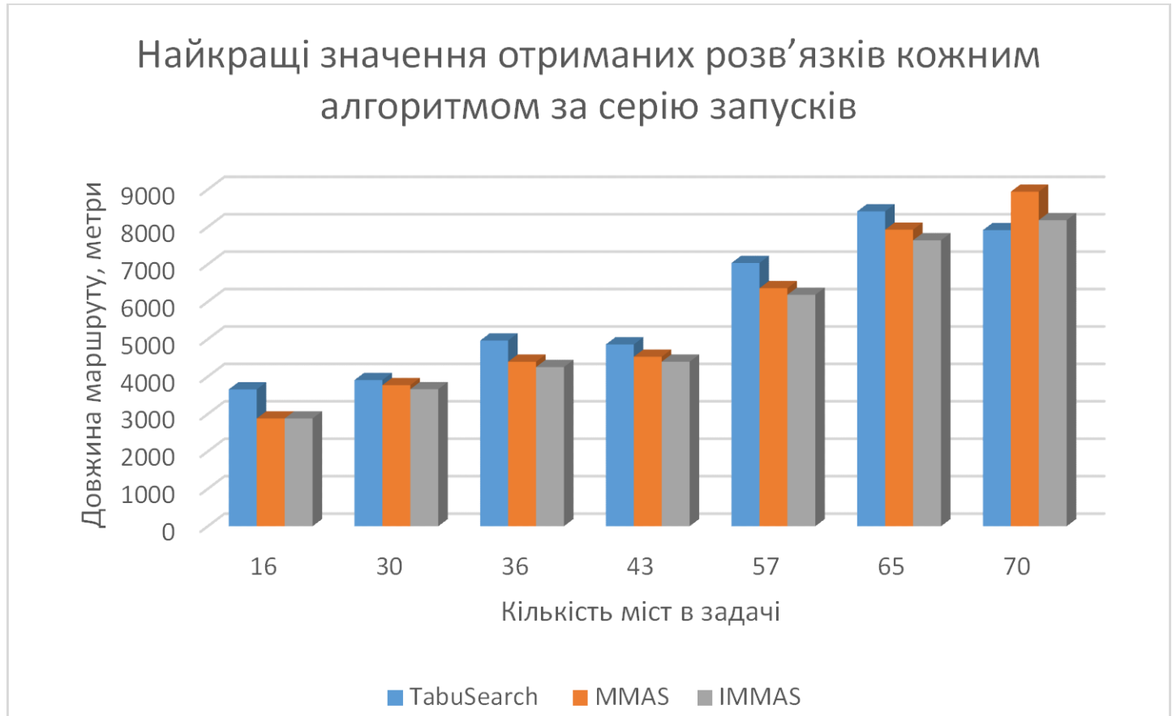
Демонстраційний плакат до магістерської дисертації

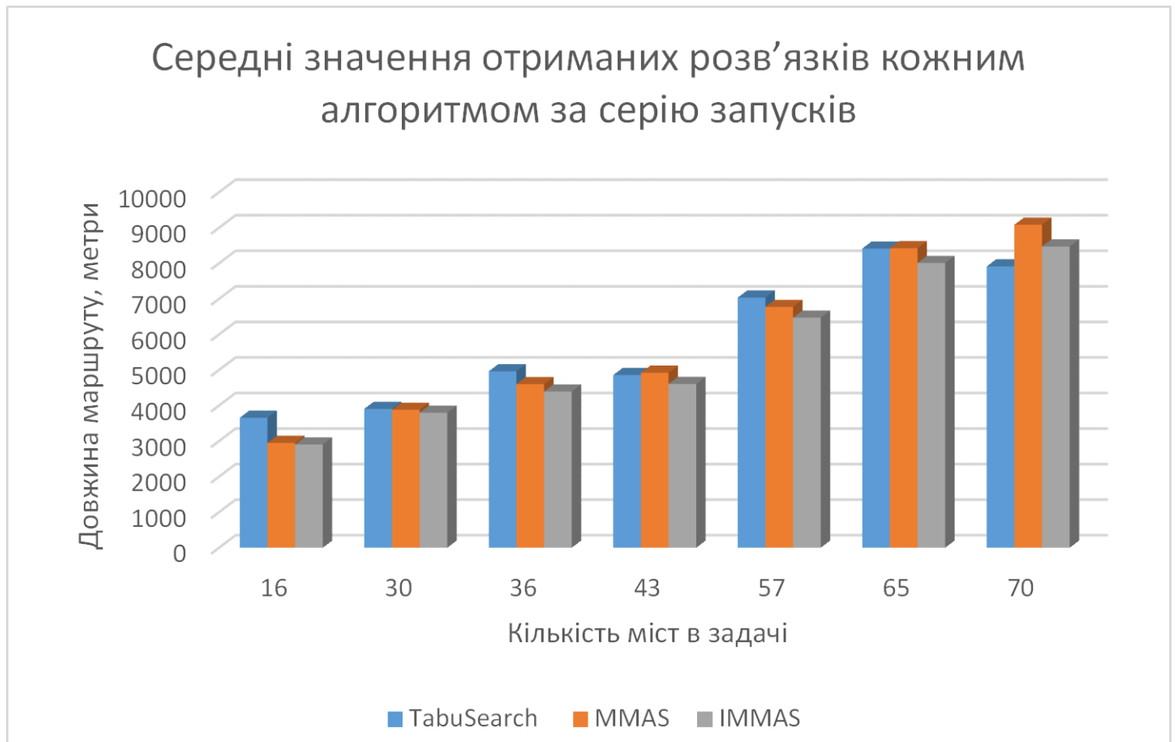
на тему «Оптимізація маршрутів групи безпілотних авіаційних систем»

Виконав студент гр. ІС-82мп — Сторчевий Владислав Володимирович

Керівник — Гуляницький Леонід Федорович

Результати експериментів





Демонстраційний плакат до магістерської дисертації

на тему «Оптимізація маршрутів групи безпілотних авіаційних систем»

Виконав студент гр. ІС-82мп Сторчевий Владислав Володимирович

Керівник Гуляницький Леонід Федорович