НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

"На правах рукопису"
УДК *519.854.2 + 004.021*

До захисту допущено
Завідувач кафедри

_____ Олександр РОЛІК
)
“ ____” _____ 20 *22* р.

Магістерська дисертація

на здобуття ступеня магістра
за освітньо-професійною програмою
«Інформаційні управляючі системи та технології»
зі спеціальності 126 «Інформаційні системи та технології» на тему:
«Інформаційна система з підтримки дослідження задачі складання маршруту польоту БПЛА з базою на транспортному засобі»

Виконала:
студентка VI курсу, групи ІС-з11мп
Вознюк Олександра Віталіївна _____

Науковий керівник:
доцент каф. ІСТ, к.т.н., доц.
Жданова Олена Григорівна _____

Рецензент:
ст. викладач каф. ОТ, к.т.н.,
Антонюк Андрій Іванович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2022 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Рівень вищої освіти – другий (магістерський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інформаційні управляючі системи та технології»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____2022 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Вознюк Олександрі Віталіївні

1. Тема дисертації «Інформаційна система з підтримки дослідження задачі складання маршруту польоту БПЛА з базою на транспортному засобі», науковий керівник дисертації Жданова Олена Григорівна, доцент, к.т.н., доцент, затверджені наказом по університету від «08» 11 2022 р. № 4097-с

2. Термін подання студентом дисертації «12» 12 2022 р.

3. Об’єкт дослідження
Складання маршруту польоту БПЛА з базою на транспортному засобі

4. Вихідні дані
Результати роботи алгоритмів та досліджень задачі складання маршруту польоту БПЛА з базою на транспортному засобі

5. Перелік завдань, які потрібно розробити
- проаналізувати предметну область;
- сформулювати математичну модель задачі;
- розробити алгоритми для складання маршруту польоту БПЛА з базою на транспортному засобі;
- розробити інформаційну систему;
- провести аналіз результатів експериментальних досліджень;

6. Орієнтовний перелік графічного (ілюстративного) матеріалу
Блок-схема алгоритму, що базується на алгоритмів найближчого сусіда, схема декомпозиції цілей системи, UML – діаграма сценаріїв використання, діаграма діяльності фрагменту розв'язку задачі, структурна схема компонентів системи, діаграмма класів підсистеми з імплементацією алгоритмів, схема бази даних, структурна схема інтерфейсу системи

7. Орієнтовний перелік публікацій

8. Дата видачі завдання «05» 09 2022 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Дослідження предметної області	08.09.2022	
2	Огляд існуючих рішень та методів	15.09.2022	
3	Розробка математичної моделі та алгоритмів	22.09.2022	
4	Розробка програмного продукту	13.10.2022	
5	Тестування програмного продукту	20.10.2022	
6	Створення документації та аналіз отриманих результатів	27.10.2022	
7	Подання роботи на основний захист	12.12.2022	

Студент Олександра ВОЗНЮК

Науковий керівник Олена ЖДАНОВА

РЕФЕРАТ

Магістерська дисертація: 101 с., 44 рис., 35 табл., 15 джерел, 1 додаток.

Актуальність. На сьогодні БПЛА широко застосовуються у багатьох сферах, а саме у військовій справі, комерційній діяльності, персональному використанні. Тому задача складання оптимальних маршрутів польоту БПЛА є акутальною.

Зв'язок роботи з науковими програмами, планами, темами. Дана робота була виконана на базі кафедри інформаційних систем та технологій НТУУ «КПІ ім. Ігоря Сікорського».

Мета і задачі дослідження. Метою є розробка алгоритмів розв'язання задачі складання маршруту польоту БПЛА з базою на транспортному засобі (ТЗ) та дослідження їх ефективності. Для цього необхідно провести аналіз предметної області; сформулювати математичну модель задачі; провести аналіз методів розв'язання задач маршрутизації ТЗ та існуючих релевантних програмних продуктів; розробити алгоритми розв'язання задачі; розробити інформаційну систему з підтримки проведення досліджень ефективності розроблених алгоритмів; виконати аналіз результатів експериментів;

Об'єкт дослідження. Об'єктом дослідження у даній роботі є процес складання маршрутів для БПЛА при додаткових умовах.

Предмет дослідження. Методи розв'язання задачі складання маршрутів БПЛА з базою на ТЗ.

Методи дослідження. Методи комбінаторної оптимізації.

Наукова новизна одержаних результатів. Розроблені нові алгоритми розв'язання досліджуваної задачі.

Практичне значення одержаних результатів. Розроблені алгоритми можуть застосовуватися у системах складання маршрутів для БПЛА, що застосовуються у реальних умовах.

БЕЗПІЛОТНІ ТРАНСПОРТНІ ЗАСОБИ, ЗАДАЧА МАРШРУТИЗАЦІЇ ТРАНСПОРТНИХ ЗАСОБІВ, АЛГОРИТМИ КОМБІНАТОРНОЇ ОПТИМІЗАЦІЇ

ABSTRACT

Master's thesis: 101 pages, 44 figures, 35 tables, 15 sources, 1 appendix.

Relevance. For today UAVs are widely used in many areas, namely in military affairs, commercial activities and personal use. Therefore, the task of drawing up optimal UAV flight routes is topical.

Relationship of work with scientific programs, plans, topics. This work was performed on the basis of the Department of Information Systems and Technologies of NTUU "Igor Sikorsky KPI".

The purpose and objectives of the study. The aim is to develop algorithms for solving the problem of drawing up a UAV flight route with a vehicle base and study their effectiveness. To do this, it is necessary to analyze the subject area; to formulate a mathematical model of the problem; to analyze the methods of solving problems of vehicle routing and existing relevant software products; to develop algorithms for solving the problem; to develop an information system to support research on the effectiveness of the developed algorithms; to analyze the results of experiments.

Object of research. The object of research in this paper is the process of making routes for UAVs under additional conditions.

Предмет дослідження. The object of research in this paper is the process of making routes for UAVs under additional conditions.

Research methods. Methods of combinatorial optimization.

Scientific novelty of the results. New algorithms for solving the studied problem have been developed.

Practical significance of the obtained results. The developed algorithms can be used in the systems of route planning for UAVs used in real conditions.

UNMANNED VEHICLES, VEHICLE ROUTING PROBLEM, COMBINATORIAL OPTIMIZATION ALGORITHMS

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ТЗ - транспортний засіб

БПЛА - безпілотний літальний апарат

VRP - Vehicle Routing Problem (задача транспортної маршрутизації)

NNA - Nearest Neighbour Algorithm (алгоритм найближчого сусіда)

PSM - Producer Scrounger Method (метод виробника та крадія)

SO - Swap Operator (оператор Swap)

SS - послідовність операторів Swap

API - Application Programming Interface

ВСТУП

Останні декілька років безпілотні транспортні засоби або БПЛА або дрони відіграють важливу роль у багатьох сферах. Починаючи з швидкої доставки і закінчуючи повітряною військовою розвідкою, БПЛА показують себе як ідеальний інструмент для використання, коли час на дію обмежений або місце яке потрібно досягти є важкодоступним.

Початок відліку історії БПЛА можна вважати роки Першої світової війни, коли американські та французькі вчені працювали над розробкою перших керованих літальних пристроїв, щоб використовувати їх у військових цілях та таким чином мати перевагу у повітрі над противником. Не дивлячись на те, що с початку Першої світової війни пройшло вже більше ніж сто років, БПЛА почали активно розвиватися та широко використовуватися лише останні пару декад. Наразі БПЛА використовуються у багатьох різних напрямках, а саме: панорамне відео та фотозйомка для журналістики та кіно, географічні дослідження місцевості, доставка товарів, рятувальні операції, контроль державного кордону, моніторинг погоди та багато іншого. Загалом можна виділити чотири основних напрями, де широко використовуються БПЛА – військова справа, комерційна діяльність, персональне використання та наукова сфера.

На війні двадцять першого століття дрони відіграють майже як не ключову роль, так як активно використовуються у бойових операціях, для розвідки та для доставки невеликих грузів. Як повідомляє Globe Newswire до двадцять сьомого року сумарний ринок БПЛА, що використовуються у військових цілях може досягнути двадцяти чотирьох мільярдів доларів. Для прикладу, один американський військовий дрон Predator коштує близько чотирьох мільйонів доларів. Ці дані дають право зробити висновок, що БПЛА будуть і надалі розвиватися у військовій сфері.

Використання БПЛА у комерції це доволі новий напрям, що досі тільки набирає обороти. З кожним роком у Європі, Азії та Північній Америції з’являють нові інвестори, що готові розвивати даний напрям. Головною причиною цього є

той факт, що з розвитком технологій використання дронів є дешевшим, а кастомізація під свої потреби все простіше.

У багатьох країнах останнім часом з’явилися закони, що регулюють використання БПЛА серед цивільного населення, так як вже було зафіксовані випадки зіткнення БПЛА з аеропланами та іншими літальними апаратами. Загалом цивільне населення використовує дрони для зйомки фото та відео. Прогнозується, що покупці витратять близько сімнадцяти мільярдів доларів на цивільні БПЛА протягом наступних декілька років.

Як бачимо із аналізу областей де використовуються БПЛА є напрямки де важливо не просто керувати БПЛА, а робити це якомога більш ефективно і тут нам на допомогу приходить задача маршрутизації.

Тож задача транспортної маршрутизації або VRP (Vehicle Routing Problem) вирішує складне питання спрощення процедури планування маршрутів. В дуже спрощеному варіанті VRP можна описати як деяку множину зупинок, де кожну зупинку необхідно відвідати з наданим набором ТЗ рівно один раз. Як ми бачимо суть даної проблеми доволі легко зрозуміти, але знайти розв’язок є не завжди тривіальною задачею. Причиною є часова складність та те, що реальні кейси вимагають додавання у постановку задачі нових умов та обмежень.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд типів задач транспортної маршрутизації маршрутизації

Мета цього підрозділу розглянути які типи VRP є актуальними на сьогодні, описати їх ідею та визначити які обмеження та додаткові умови фігурують у постановках задач найчастіше протягом останніх декілька років.

Почнемо з декількох слів про класичну VRP. Метою VRP є знайти оптимальні маршрути так, щоб кожний клієнт був відвіданий тільки один раз одним ТЗ. Реальні вимоги і нові виклики потребують розширення формулювання класичної VRP. Згідно статті [1], що містить аналіз статей по VRP за 2019 – 2021, найчастіше фігурують такі характеристики постановки задачі як часові вікна, обмежений час роботи кожного ТЗ, особливості дорожнього трафіку, ТЗ з різними характеристиками, різні типи ТЗ включаючи електричний транспорт та БПЛА. Метою розв’язання цих задач є:

— мінімізація витрат на експлуатацію ТЗ;

— мінімізація кількості ТЗ;

— мінімізація енерговитрат;

— мінімізація подоланої сумарної відстані;

— мінімізація часу очікування;

— мінімізація викидів парникових газів;

— мінімізація сумарного витраченого часу;

— мінімізація логістичних витрат;

— максимізація прибутку.

На рисунку 1.1 представлені найбільш згадані у наукових роботах останніх років характеристики задач маршрутизації, а саме:

— time window rate – задачі з часовими вікнами;

— heteronegenous – умова використання ТЗ різних типів;

— electronic rate – зменшення енерговитрат при складанні маршруту;

— single-object – оптимізація одного обраного тільки однієї величини;

— Cost/dis/emissions rate – зменшення собівартості/рівня викидів;

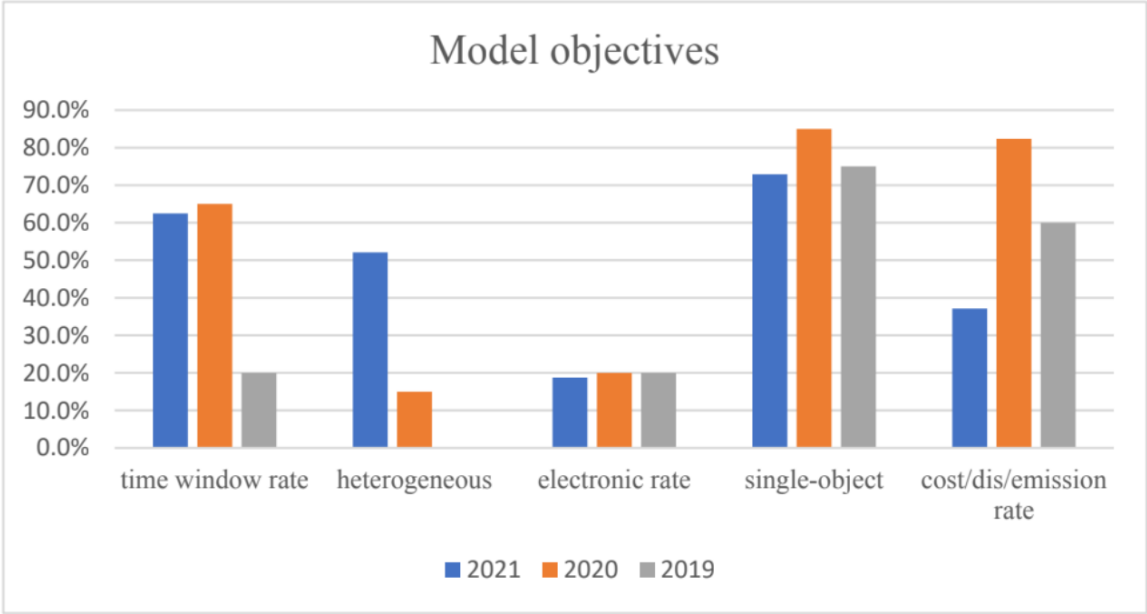


Рисунок 1.1 – Поширеність типів задач маршрутизації за 2019-2021 роки

Як видно з графіку часові вікна займають велику частину у просторі постановок задач. У порівнянні з попередніми роками, у 2021 можемо бачити стрімке збільшення уваги до задач з використанням гетерогенних ТЗ (heterogeneous). Постановки задач з фігуруванням в них електричних ТЗ хоч і зараз не займають лідуючі позиції, але все ж таки стабільно цікаві та впевнено тримають частку у 20 відсотків кожного року. Очевидно, причиною цього є тренд на технології збереження довколишнього середовища, що зберігається останні роки. На мою думку, в подальшому частка статей, що містять постановку VRP з електричними ТЗ буде зростати, так як сфера розробки та використання екологічно чистих продуктів та технологій є доволі новою і ще такою, що не вичерпала свій потенціал та актуальність.

Далі розглянемо які типи VRP окрім класичної були предметом дослідження 2019 – 2021 роки:

— VRPTW (VRP with time windows) – це VRP, де обслуговування кожного клієнта повинно відбуватися у заданому часовому інтервалі, що називають часовим вікном;

— CVRP (Capacitated VRP) – це VRP, де кожний ТЗ містить задану вантажопідйомність, яку не повинен перевищувати при перевезенні;

— EVRP (Electric VRP) – це VRP, де у ролі ТЗ виступають електричні транспортні засоби, що мають певний об’єм заряду;
— EVRPTW (Electric VRP with time widows) – це EVRP з часовими вікнами;
— EVRPTW-SP (EVRPTW at most a single (S) recharge per route, and partial (P) battery recharges are possible) - це EVRPTW, де можливі як повні так і часткові підзарядки між маршрутами.
— GVRP (Green VRP) – це VRP з ТЗ на альтернативних видах палива;
— HFVRP (VRP with heterogeneous fleets) – це VRP, де окрім визначення маршрутів необхідно визначити який тип ТЗ призначити на кожний маршрут;
— MDVRP (Multi-depot VRP) – це VRP з багатьма депо;
— TDVRP (Time-dependent VRP) – це VRP, де час поїздки між двома клієнтами залежить від дистанції між ними та часу доби;
— TDVRPTW (Time-dependent VRP with time windows) – це TDVRP з часовими вікнами;
— TWAVRP (Time window assignment VRP) – це VRP ціль якої розподілити часові вікна між клієнтами;
— VRPSD-PDC (VRP with stochastic demands and probabilistic duration constraints) – це VRP зі стохастичним попитом та імовірнісними обмеженнями на тривалість;

1.2 Методи розв’язання задач маршрутизації

Мета цього підрозділу розглянути які існують методи розв’язання VRP та зокрема визначити які з них є найбільш актуальними останні роки.

Отже, на сьогодні існує велике різноманіття стратегій розв’язання VRP. На рисунку 1.2 представлена їх класифікація, що була взята із [2].

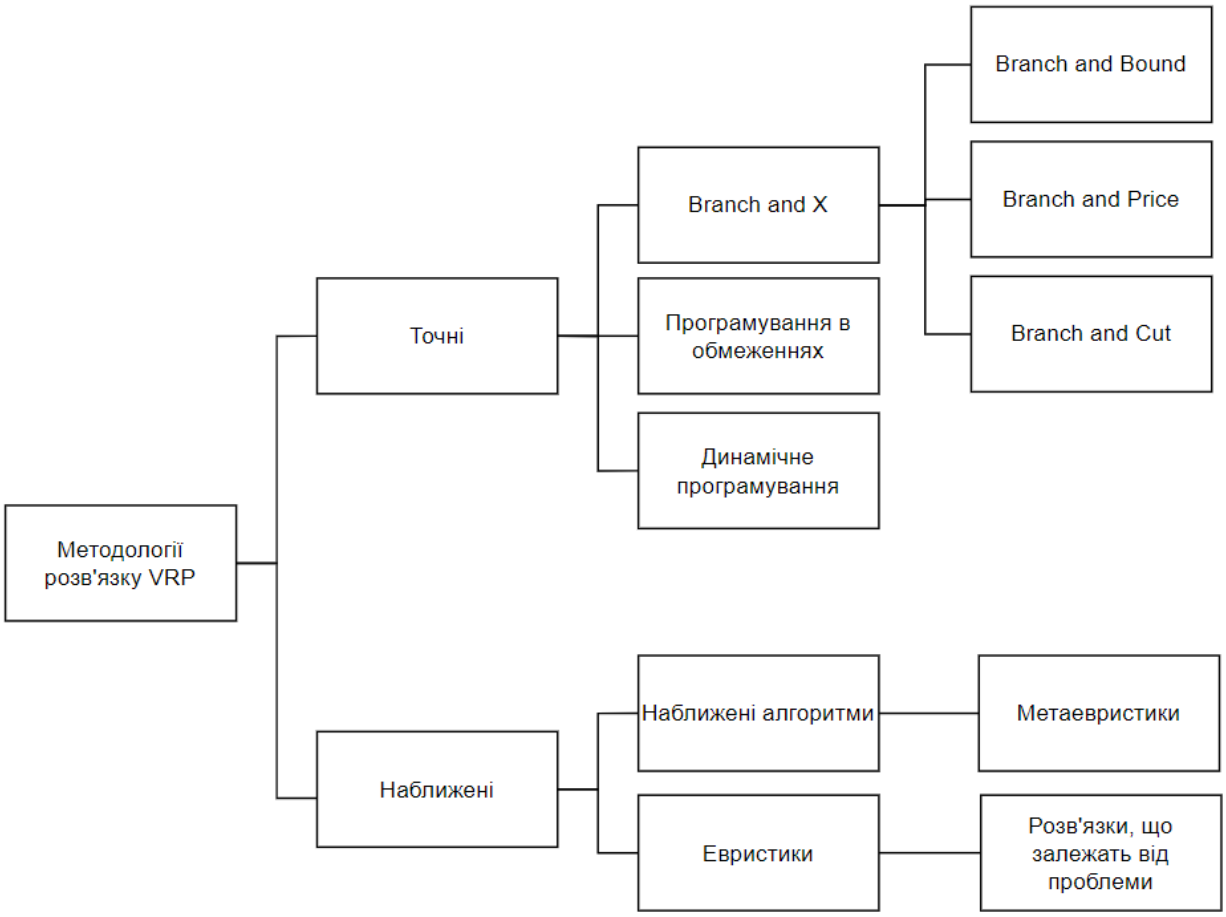


Рисунок 1.2 – Класифікація стратегій розв’язку VRP

Як видно з рисунку 1.2 ці стратегії можна поділити на точні та наближені. Точні методи надають оптимальний розв’язок у той час як евристичні та метаевристичні підходи намагаються лише наблизитися до нього. Точні методи зазвичай є придатними тільки для розв’язку невеликих за розмірністю задач (приблизно до 200 клієнтів). Причиною цього є те, що VRP є NP-складною задачею, що означає розв’язання з більш великими розмірностями є дуже дорогим за часом. Евристичні та матаевристичні зазвичай є значно менш затратні у плані часу, але при цьому доводиться жертвувати точністю знайденого розв’язку – обмежитися лише близьким до оптимального результатом.

Згідно статті [1] для розв’язку VRP у 2019-2021 найчастіше використовувалися евристичні та метаевристичні підходи. Наприклад, метод відпалу (simulated annealing), генетичний алгоритм (genetic algorithm), генетичний алгоритм недомінантного сортування (non-dominated sorting genetic algorithm), спрощена

ройова оптимізація (simplified swarm optimization) і так далі. Не дивлячись на основний тренд, у 2021 спостерігався приріст використання точних методів, зокрема гілок та меж. Тож можна припустити, що у наступних роках їх буде все більше і більше завдяки збільшенню обчислювальної потужності комп’ютерів.

1.3 Існуючі програмні забезпечення

1.3.1 Routific Route Optimization API

Routific це компанія яка надає програмне забезпечення Route Optimization API, що дозволяє розв’язувати VRP [3]. У портфоліо даної компанії є такі юз кейси:

— доставка посилок;

— виїзні служби;

— дистрибутивна логістика;

— квіткові магазини;

— доставка медичного обладнання;

— вивезення відходів;

Розробники стверджують, що їхні алгоритми здатні зберегти до 40 відсотків часу на доставку або палива і зменшити час, який необхідний для планування маршруту на 95 відсотків. Наразі програмний продукт використовується у більш ніж 900 містах та здатний враховувати такі обмеження як часові вікна, місткість ТЗ, час обслуговування, декілька депо, незамкнені маршрути, різні типи ТЗ та зміни водіїв.

1.3.2 Google OR-Tools

Google OR-Tools - це комплекс програмних продуктів з відкритим кодом, що надає можливість розв’язувати задачі оптимізації. Google OR-Tools надає бібліотеку для розв’язку VRP на мові Java, C#, Python та C++. Окрім класичної VRP даний програмний продукт також дозволяє розв’язувати задачу з обмеженням на місткість ТЗ та часовими вікнами [4].

1.3.3 AIMMS

AIMMS це платформа яка надає можливості створювати та публікувати застосунки, що пов’язані з прийняттям рішень [5]. Середовище для розробки комбінує у собі унікальні можливості для моделювання та інструменти для дизайну, що дозволяють швидко створювати нові програмні продукти. У рамках своєї платформи AIMMS надає бібліотеку для розв’язання CVRP (Задача маршрутизації з обмеженням місткості).

Висновки до розділу

У рамках даного підрозділу була проаналізована предметна область. Розглянута класифікація типів задач маршрутизації та методів їх розв’язання. Згідно аналізу можна зробити висновок, що останні декілька років популярними є евристичні та метаевристичні алгоритми. Причиною цього є те, що такі алгоритми менш затратні по часу ніж інші існуючі методи. Це є важливим, оскільки, сучасні задачі маршрутизації мають велику кількість обмежень та додаткових умов, що ускладнюють математичну модель і підвищують часову складність. Також були проаналізовані деякі вже існуючі програмні продукти для складання маршрутів. Всі з них мають доволі обмежений вибір у методах розв’язання задачі. Також вони не надають гнучке налаштування параметрів задачі, що дозволяє розв’язувати лише класичні задачі маршрутизації без додаткових умов.

2 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Математична постановка задачі

2.1.1 Задача 1 (задача з однією базою)

2.1.1.1 Змістовна постановка Задачі 1

Дано один БПЛА, множину цілей для обстеження та один ТЗ, який знаходиться на базі B_1. БПЛА повинен облетіти усі цілі, стартуючи з бази B_1. Оскільки БПЛА має обмежений польотний ресурс, то до його закінчення він повинен повернутися на базу, пройти обслуговування (поповнення польотного ресурсу) і продовжити обліт цілей.

Завдання полягає у мінімізації сумарного часу, який необхідний для обльоту усього маршруту враховуючи при цьому обмеженість польотного ресурсу БПЛА.

На рисунку 2.1 зображене графічне представлення задачі з 8-ма цілями та наведено один із можливих варіантів обльоту цих цілей.

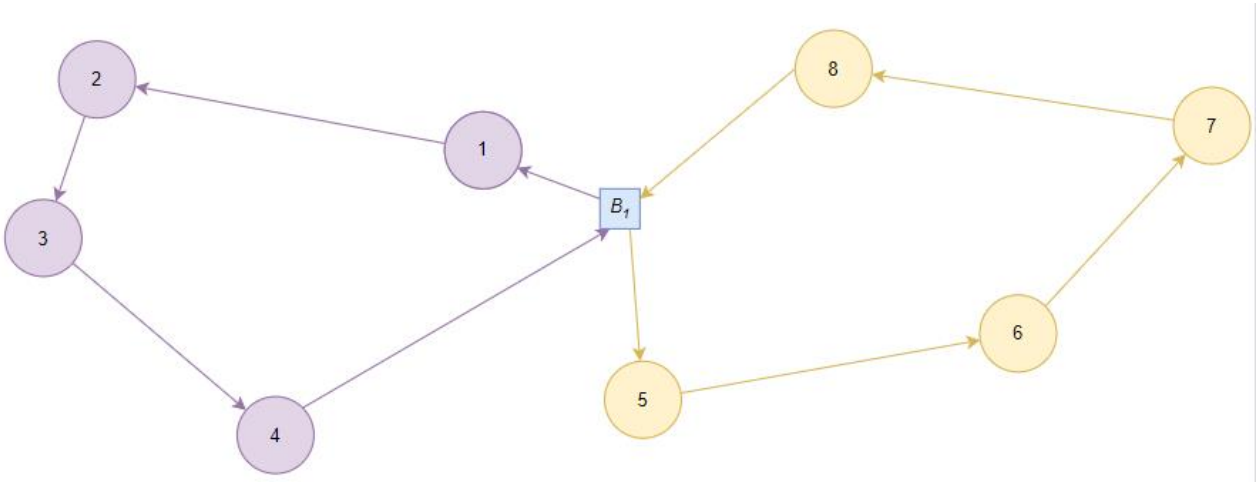


Рисунок 2.1 – Графічне представлення варіанту розв’язку задачі із вісьма цілями та однією базою

Відповідно до рисунка 2.1 у розв’язку задачі наявні два підмаршрути, послідовність обльоту яких $B_1 - 1 - 2 - 3 - 4 - B_1$ та $B_1 - 5 - 6 - 7 - 8 - B_1$.

2.1.1.2 Вхідні, проміжні та вихідні дані

Наведемо вхідні дані, що необхідні для розв’язку Задачі 1. Також проміжні дані, що будуть використовуватися в описі математичної моделі і вихідні дані, що є результатом розв’язку задачі.

Вхідні дані:

— координати бази B_1;

— координати цілей для обльоту;

— t – місткість БПЛА у часовій величині (час у польоті без поповнення);

— τ – час на обслуговування БПЛА на базі (поповнення польотного ресурсу);

— v – швидкість БПЛА.

Проміжні дані:

— матриця H відстаней між цілями та між базою і цілями;

— матриця S часів переміщення БПЛА між цілями.

Вихідні дані:

— кількість підмаршрутів для обльоту;

— порядок за яким цілі у кожному підмаршруті будуть відвідані;

— сумарний час на виконання обльоту цілей;

— часовий розклад обслуговування БПАЛ (поповнення польотного ресурсу).

Квадратна матриця H складається із координат баз та цілей. В свою чергу квадратна матриця S визначається на основі елементів матриці H, що поділені на швидкість БПЛА v.

2.1.1.3 Математична модель Задачі 1

Дано:

— $J = \{1, \dots, n\}$ – множина цілей БПЛА, де n – їх кількість;

— $\{0\}$ - місце розташування бази B_1;

— h_{kl} $(k, l \in \{0,1, \dots, n\})$ – матриця відстаней (як між цілями та базою так і між цілями);

— $s_{kl} = \frac{h_{kl}}{v}$, $k, l \in \{0,1, \dots, n\}$, – матриця часів пересування БПЛА між цілями k, l .

Для мінімізації сумарного часу, що витрачається на обліт цілей у формулі 2.1 наводиться розбиття множини J на R упорядкованих підмножин, кожна з яких відповідає одному підмаршруту для обльоту цілей.

$$J_1 = \left(j_1^1, j_2^1 \dots j_{n_1}^1\right), \dots, J_R = \left(j_1^R, j_2^R \dots j_{n_R}^R\right), \tag{2.1}$$

таких що

$$\bigcup J_r{}_{r=1}^{R} = J, \tag{2.2}$$

$$J_i \cap J_j = \emptyset, \text{для } i \neq j. \tag{2.3}$$

Варто зауважити, що час обльоту кожного підмаршруту не повинен бути більшим ніж місткість БПЛА у часовому вимірі без поповнення. Дане обмеження описане у рівнянні 2.4:

$$\mathrm{S}_r = s_{0j_1^r} + \sum_{q=1}^{n_r-1} s_{j_q^r j_{q+1}^r} + s_{j_{n_r}^r 0} \leq t, \;\; r = 1, \dots, R. \tag{2.4}$$

Цільова функція представлена у рівнянні 2.5 являє собою мінімізацію сумарного часу обльотів усіх підмаршрутів, враховуючи час на обслуговування БПЛА між обльотами.

$$z = (R-1)\tau + \sum_{r=1}^{R} S_r \ \rightarrow min. \qquad (2.5)$$

Примітка: якщо кількість допустимих підмаршрутів $R = 1$, то $z = \sum_{r=1}^{R} S_r$, оскільки час обслуговування БПЛА на базі можна не враховувати. Зауважимо, що тут і далі:

$$n_r = |J_r|, \qquad r = 1, \dots, R, \qquad (2.6)$$

$$\sum_{r=1}^{R} n_r = n. \qquad (2.7)$$

Часовий розклад поповнення ресурсу визначається за результатами розв'язання задачі, а саме з урахуванням кількості підмаршрутів обльоту цілей та порядку обльоту цілей в кожному підмаршруті.

2.1.2 Задача 2 (задача з синхронізацією БПЛА та ТЗ з двома рівнозначними базами)

2.1.2.1 Змістовна постановка Задачі 2

Дано один ТЗ, один БПЛА, множину цілей для обльоту, розташування баз B_1 та B_2 та час переміщення ТЗ між ними. БПЛА стартує з ТЗ, що розташовується на базі B_1 та облітає множину цілей формуючи перший підмаршрут. Після цього БПЛА повертається на базу B_1 та встає на обслуговування. Так може повторюватись декілька разів. Далі ТЗ вирушає до бази B_2, тому приземлення чи обслуговування далі може відбуватися уже лише на базі B_2. Сенс синхронізації полягає у тому, що час переміщення транспортного засобу до бази B_2 має бути не меншим, ніж час проходження підмаршруту БПЛА від бази B_1 до бази B_2, тобто щоб ТЗ прибув до B_2 раніше, ніж БПЛА, і чекав на нього.

Завдання полягає у мінімізації сумарного часу з урахуванням обмеженості польотного ресурсу БПЛА. Отже, задача розбивається на три підзадачі:

— підзадача 1 - визначення підмаршрутів БПЛА, що починаються з бази B_1;

— підзадача 2 - пошук такого маршруту БПЛА, щоб він дістався до B_2 за час, не менший, ніж час ТЗ в дорозі;

— підзадача 3 - визначення підмаршрутів БПЛА, що починаються з бази B_2.

Підзадача 1 по суті співпадає із Задачею 1 за винятком того, що в даному випадку обстежується тільки деяка підмножина цілей I з множини J ($I \subseteq J$).

На рисунку 2.2 зображене графічне представлення можливого варіанту обльоту цілей для Задачі 2. А саме послідовний обліт БПЛА цілей у трьох підмаршрутах: $B_1 - 8 - 7 - 6 - B_1$, $B_1 - 5 - B_2$ та $B_2 - 4 - 3 - 2 - 1 - B_2$.

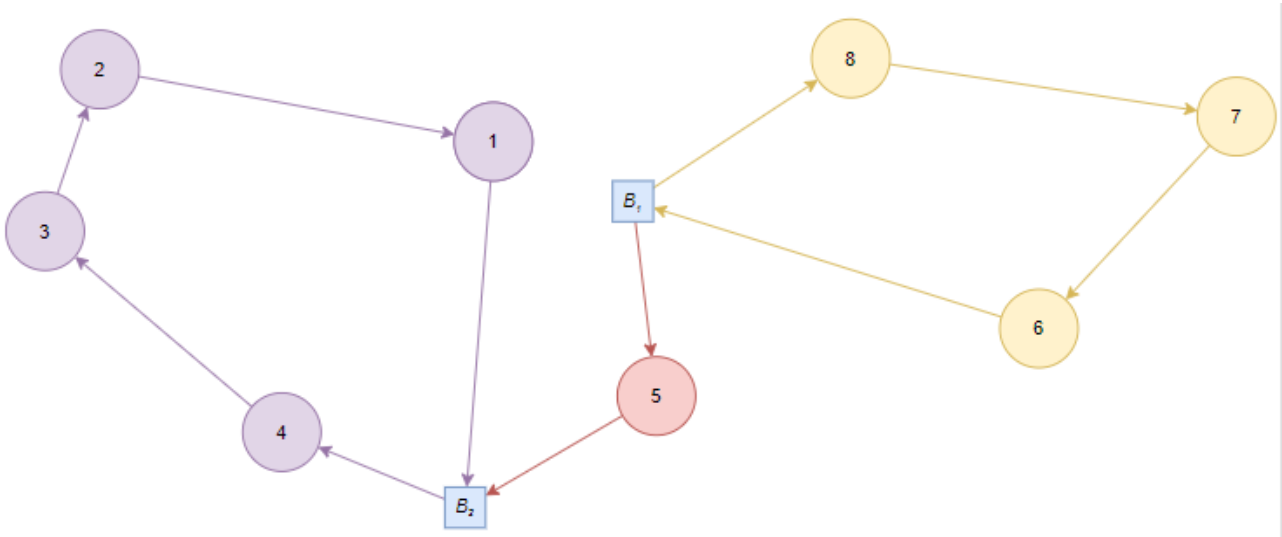


Рисунок 2.2 - Графічне представлення можливого розв’язку задачі із вісьма цілями та двома базами

2.1.2.2 Вхідні, проміжні та вихідні дані

Вхідні дані співпадають із вхідними даними із Задачі 1, окрім наявності двох баз B_1 та B_2, а також нового доданого параметра Δ – часу переміщення ТЗ від бази B_1 до бази B_2.

2.1.2.3 Математична модель Задачі 2

Дано:

— $J = \{1, \dots, n\}$ – множина цілей для обльоту БПЛА;

— $\{0, n+1\}$ - місця розташування баз B_1 та B_2;

— $\{h_{kl}\}$ $(k, l \in \{0,1, \dots n\})$ – матриця відстаней між цілями та базами;

— $\{s_{kl}\}(k, l \in \{0,1, \dots n, n+1\})$ – матриця часів переміщення БПЛА.

У формулах 2.8 – 2.10 представлене розбиття множини J на $R = W + Y + 1$ упорядкованих підмножин (W та Y є шуканими величинами), що відповідають підмаршрутам обльоту цілей.

$$J_1 = \left(j_1^1, j_2^1 \dots j_{n_1}^1\right) \dots J_R = \left(j_1^W, j_2^W \dots j_{n_W}^W\right), \tag{2.8}$$

$$J_{W+1} = \left(j_1^{W+1}, j_2^{W+1} \dots j_{n_{W+1}}^{W+1}\right) \dots J_{W+Y} = \left(j_1^{W+Y}, j_2^{W+Y} \dots j_{n_{W+Y}}^{W+Y}\right), \tag{2.9}$$

$$J_R = \left(j_1^R, j_2^R \dots j_{n_R}^R\right), \tag{2.10}$$

таких що

$$\bigcup_{r=1}^{R} J_r = J, \tag{2.11}$$

$$J_i \cap J_j = \emptyset, \text{де } i \neq j, \tag{2.12}$$

де W – кількість підмаршрутів, що починаються і закінчуються на базі B_1;

Y – кількість підмаршрутів, що починаються і закінчуються на базі B_2;

J_R – множина, що відповідає підмаршруту, що починається в B_1 та закінчується в B_2 (у виразі $R = W + Y + 1$ їй відповідає доданок $+1$).

Сумарний час кожного із W підмаршрутів повинен бути не більшим ніж час у польоті БПЛА без поповнення:

$$S_r = s_{0j_1^r} + \sum_{q=1}^{n_r-1} s_{j_q^r j_{q+1}^r} + s_{j_{n_r}^r 0} \le t, \qquad r = 1,..,W. \tag{2.13}$$

Крім того, час $(W+1)-$ого підмаршруту БПЛА (з двома базами) повинен бути не меншим за час пересування ТЗ від першої до другої бази:

$$S_r = s_{0j_1^r} + \sum_{q=1}^{n_1-1} s_{j_q^r j_{q+1}^r} + s_{j_{n_r}^r (n+1)} \ge \Delta, \qquad r = W+1. \tag{2.14}$$

Сумарний час усіх наступних Y підмаршрутів, що представлений у рівнянні 2.15, також повинен бути не більшим ніж місткість БПЛА:

$$S_r = s_{(n+1)j_1^r} + \sum_{q=1}^{n_r-1} s_{j_q^r j_{q+1}^r} + s_{j_{n_r}^r (n+1)} \le t, \qquad r = R-Y,..,R. \tag{2.15}$$

Цільова функція представлена у рівнянні 2.16 полягає у мінімізація сумарного часу, який витрачається на обліт маршруту із врахуванням часу, який необхідний на обслуговування БПЛА.

$$z = (R-1)\tau + \sum_{r=1}^{R} S_r \to min. \tag{2.16}$$

2.2 Опис розроблених алгоритмів

Розроблені алгоритми складаються з двох етапів:

— пошук початкового розв’язку задачі;

— покращення знайденого розв’язку за допомогою методів локального пошуку.

На першому етапі використовуються модифікації існуючих методів таким чином, щоб пристосувати їх до математичної моделі Задачі 2. Отже, були розроблені такі алгоритми як алгоритм на основі методу найближчого сусіда та алгоритм на основі кластеризації k-means та алгоритму Sweep. На другому етапі виконується покращення маршруту, що був знайдений на першому етапі. Для покращення сумарного маршруту обльоту цілей БПЛА була обрана стратегія покращення кожного підмаршруту окремо. Для цього були використані оператори Sweep та Insertion, а також алгоритми на основі Firefly та Producer-Scrounger. Також був розроблений алгоритм, мета якого покращити розв’язок задачі за допомогою розширення підмаршруту між базами, оскільки після пошуку початкового розв’язку задачі цей підмаршрут не містить цілей для обльоту. При проведені експериментальних досліджень буде представлена варіанти комбінацій описаних методів між собою для створення нових алгоритмів розв’язання задачі.

2.2.1 Методи ініціалізації початкового розв’язку задачі

2.2.1.1 Алгоритм на основі алгоритму найближчого сусіда

Даний алгоритм базується на основі алгоритму найближчого сусіда (NNA - Nearest Neighbour Algorithm), що використовує “жадібний” підхід для пошуку розв’язку. Даний алгоритм є одним з перших і найбільш простих евристичних методів розв'язування задачі комівояжера. За кожен крок його виконання до знайденої частини маршруту додається нове ребро, що є найближчим від останнього доданого. Алгоритм припиняє роботу, коли розв'язок знайдено і не намагається його покращити. При даній постановці задачі алгоритм має деякі модифікації. Перша модифікація це те, що алгоритм виконується декілька разів поки всі цілі не будуть розподілені по підмаршрутам. Також оскільки дано дві бази, то алгоритм виконується два рази – починаючи з першої бази або з другої (враховуючи втрати у часі при переміщенні ТЗ з першої бази до другої). Далі обирається кращий розв’язок з двох отриманих. Блок-схема цього алгоритму наведена у Додатку А.

Даний алгоритм для знаходження підмаршрутів для однієї бази можна коротко описати у вигляді таких кроків:

а) встановити індекс кластеру (підмаршруту) $K = 1$;

б) додавати найближчу ціль від останнього доданого елементу в кластері при умові допоки не порушується умова місткості БПЛА;

в) зупинитися коли умова місткості БПЛА буде порушена;

г) створити новий кластер $K + 1$ та продовжити додавання цілей до підмаршруту, що залишилися;

2.2.1.2 Алгоритм на основі кластеризації k-means та алгоритму Sweep

Даний алгоритм складається з двох кроків: кластеризація за допомогою алгоритму k-means та побудова підмаршрутів в межах знайдених кластерів за допомогою алгоритму Sweep.

Для пошуку початкових розв'язків для даної задачі евристичний механізм починається з групування цілей за допомогою метода, що є аналогічним алгоритму кластеризації k-середніх. Зокрема, клієнти групуються з використанням набору вагових значень, визначених на основі їх відповідних відстаней від кожної бази. Вагові значення визначаються за формулою 2.17 [6]:

$$w_{ij} = \frac{1}{d_{ij} * \sum_{k=1}^{D} \frac{1}{d_{ik}}}, \tag{2.17}$$

де w_{ij} – це вага відстані між і-м клієнтом та j-ою базою;

d_{ij} - відстань між клієнтом і та базою j;

d_{ik} - відстань між клієнтом I та базою k.

У кожному випадку більше значення ваги вказує на меншу відстань між ціллю та базою.

Після формування двох кластерів, що відповідають за першу та другу базу необхідно розподілити цілі по підмаршрутам за допомогою алгоритму Sweep. Ідея даного алгоритму полягає у розрахунку полярного кута для кожної цілі. Далі формування кластерів починається з 0 і просувається до 360 градусів, для того щоб розподілити кожну з цілей під різні підмаршрути, не перевищуючи при цьому місткість БПЛА. Такий вид “підмітання” (sweep) називається прямим підмітанням. Формула обчислення полярного кута θ по відношенню до цілей виглядає наступним чином:

$$\theta = tan^{-1}\left(\frac{y}{x}\right), \tag{2.18}$$

де $x, y -$ координати цілей.

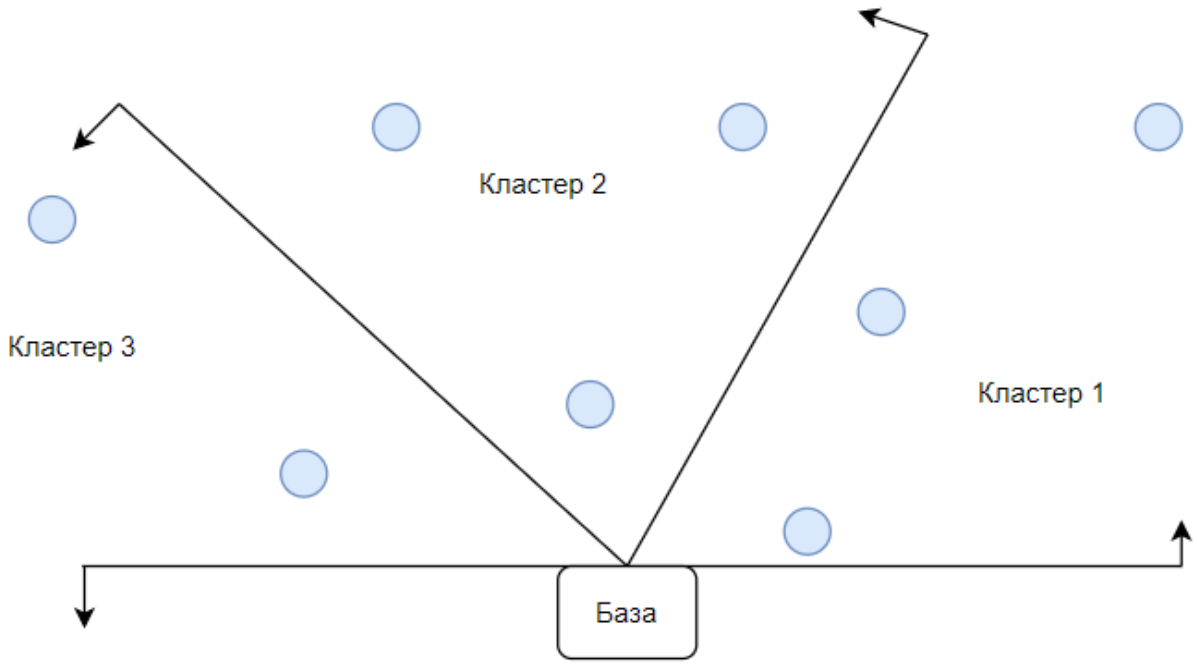


Рисунок 2.3 – Формування підмаршрутів за допомогою алгоритму Sweep

Тож алгоритм можна представити у вигляді двох етапів: ініціалізація та кластеризація (формування підмаршрутів) [7]. Наведемо більш детальний опис цих двох етапів:

а) ініціалізація:

1) розрахувати полярний кут для кожного клієнта використовуючи рівняння;

2) розмістити клієнтів у зростаючому порядку отриманих полярних кутів;

3) встановити індекс кластеру (підмаршруту) $C = 1$.

б) кластеризація:

1) враховувати клієнтів до кластеру C починаючи з клієнта з найменшим полярним кутом;

2) зупинитися коли додавання нового клієнта перевищить місткість БПЛА;

3) створити новий кластер $C + 1$ та поновити додавання клієнтів, що залишилися;

4) повторити кроки а.1) – а.3) допоки всі клієнти не будуть відвідані.

2.2.2 Алгоритми локального пошуку

В даній роботі алгоритми локального пошуку застосовуються для покращення отриманого розв’язку задачі методом покращення кожного підмаршруту окремо або усього маршруту одночасно.

2.2.2.1 Оператор Swap

Ідея оператора Swap (SO) у попарній зміні положення цілей у підмаршруті. У даній роботі застосовується як одиничний оператор так і як ланцюжок із Swap операторів:

$$SS = (SO_1, SO_2, SO_3, \dots, SO_n). \quad (2.19)$$

SO представляє пару з індексів двох позицій, які будуть змінені місцями. Візьмемо підмаршрут S із чотирма цілями $B_1 - 1 - 2 - 3 - 4 - B_1$. Тоді $SO(4,6)$ буде давати новий розв'язок наведений у рівнянні 2.20:

$$S' = S + SO(2,4) = (B_1 - 1 - 2 - 3 - 4 - B_1) + \\ + SO(4,6) = (B_1 - 1 - 4 - 3 - 2 - B_1). \quad (2.20)$$

2.2.2.2 Оператор Insertion

Ідея даної функції у виборі випадкової цілі у підмаршруті та вставки її на іншу випадкову позицію. В даному випадку ціль може міняти свою позицію тільки у рамках свого підмаршруту. Принцип роботи даного оператора можна описати такими кроками:

а) випадково обрати підмаршрут;

б) випадково обрати ціль, яку необхідно перемістити;

в) випадково обрати підмаршрут, у який необхідно перемістити обрану ціль;

г) випадково обрати позицію у обраному підмаршруті, на яку необхідно розмістити обрану ціль;

д) перемістити ціль, якщо при цьому покращується розв'язок та не порушується умова місткості БПЛА.

2.2.2.3 Алгоритм розширення підмаршруту між базами

Даний алгоритм створений для того, щоб покращити розв'язок шляхом додавання цілей до підмаршруту між базами B_1та B_2. Його ідея у послідовному додаванні цілей до підмаршруту між базами допоки розв'язок покращується. При цьому розглядаються не усі наявні цілі як потенційні для додавання, а лише ті, часова відстань БПЛА яких до прямої підмаршруту B_1B_2 не перевищує половину часової відстані для БПЛА між базами. Як видно на рисунку 2.4 єдина потенційна ціль, що

може бути додана це ціль з індексом 5, так як її довжина до прямої становить 1 годину, що є менше ніж $\frac{B_1B_2}{2} = 3.5$ годин.

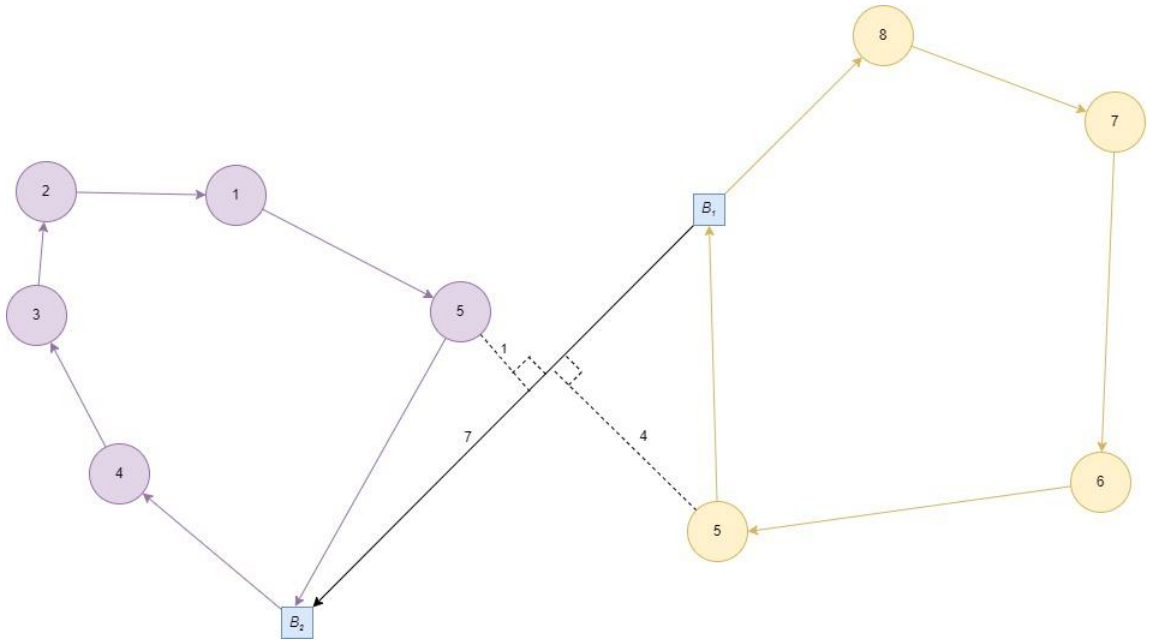


Рисунок 2.4 – Пошук потенційних цілей для підмаршруту B_1B_2

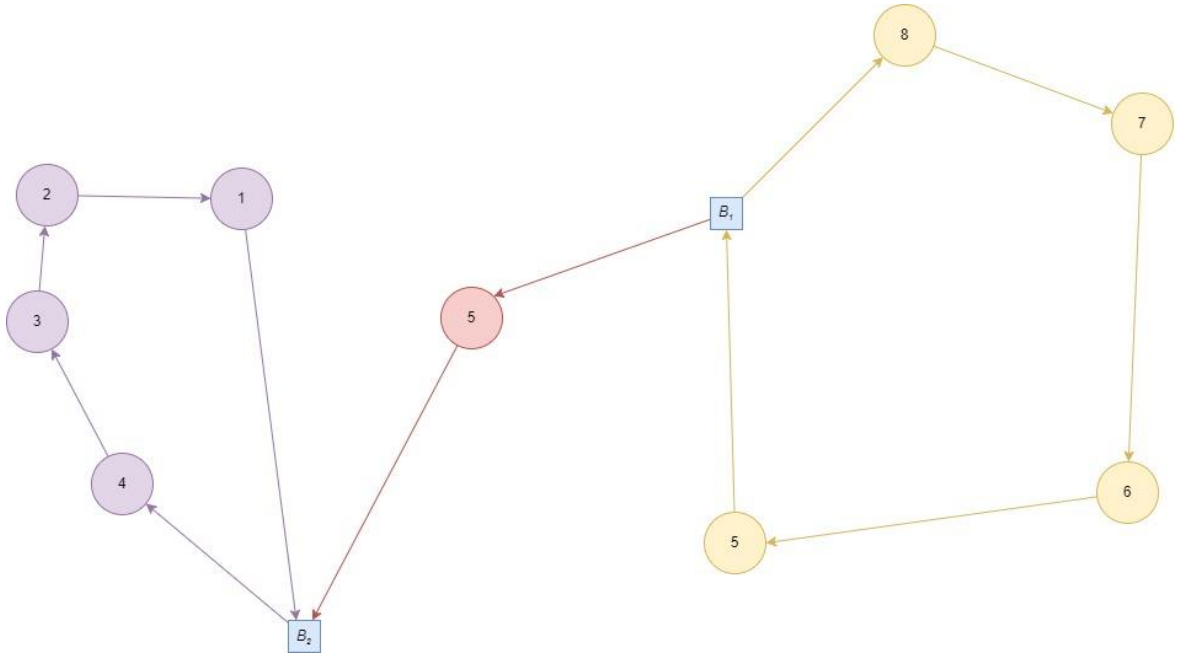


Рисунок 2.5 – Розв’язок задачі після застосування алгоритму розширення підмаршруту між базами

2.2.2.4 Метод Producer-Scrounger

PSM (Producer-Scrounger Method) - це нова техніка розв’язання задачі комівояжера, ідея якої з’явилася із спостереження за поведінкою груп тварин. Цей метод пропонує розбити задану популяцію на три ролі: виробник, крадії та блукаючі [8]. Роль кожного учасника в популяції змінюється в залежності від результату його цільової функції на кожній ітерації. Для задачі, що розглядається у даній роботі метод PSM застосовується ізольовано для кожного підмаршруту.

Виробник має завжди найкращий розв’язок задачі із усієї популяції і його мета це пошук їжі. Він виконує сканування місцевості поблизу, щоб знайти кращі ресурси. Алгоритм для пересування виробника можна представити у вигляді таких кроків:

а) обрати випадкову ціль;

б) знайти наступну ціль, що є найближчою до обраної;

в) створити зв’язок між цими двома цілями.

У пункті в) при встановлені зв’язку між двома цілями створюється декілька нових потенційних розв’язків, так як дві обрані цілі можна з’єднати декількома способами. Наприклад маємо розв’язок виробника $B_1 - 1 - 2 - 3 - 4 - 6 - 5 - 7 - 8 - 9 - 10 - B_1$. Випадково обрана ціль – ціль з індексом 6. Нехай ціль з індексом 7 найближча до обраної цілі з індексом 6. Тоді зв’язок між 6 і 7 може покращити розв’язок і відповідно до опису вище з’являться такі нові розв’язки:

а) $B_1 - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10 - B_1$;

б) $B_1 - 1 - 2 - 3 - 4 - 5 - 7 - 6 - 8 - 9 - 10 - B_1$;

в) $B_1 - 1 - 2 - 3 - 4 - 7 - 6 - 5 - 8 - 9 - 10 - B_1$;

г) $B_1 - 1 - 2 - 3 - 4 - 6 - 7 - 5 - 8 - 9 - 10 - B_1$;

Завдання крадіїв це переслідувати виробника, щоб розділити ресурси знайдені ним. Для пересування крадіїв використовується SS, що описаний у пункті 2.2.2.1.

Крадій прийме новий розв’язок за поточний тільки при умові, що це наблизить його до виробника. Для пошуку відстані між виробником та крадієм

використовується відстань Геммінга, де підраховується кількість позицій у двох розв’язках де цілі відрізняються. Наприклад маємо два підмаршрути:

а) $B_1 - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10 - B_1$;

б) $B_1 - 1 - 3 - 2 - 5 - 4 - 6 - 7 - 8 - 9 - 10 - B_1$.

Відстань Геммінга між підмаршрутами а) та б) становить 4.

Також існує якась частина популяції (блукаючі), що відірвані від групи виробника та крадіїв, тож вони бродять у випадкових напрямах. Для руху блукаючих також будемо використовувати SS не зважаючи при цьому на місцезнаходження виробника.

Графічне представлення даного методу, а саме характер поведінки кожної з трьох описаних груп наведено на рисунку 2.6.

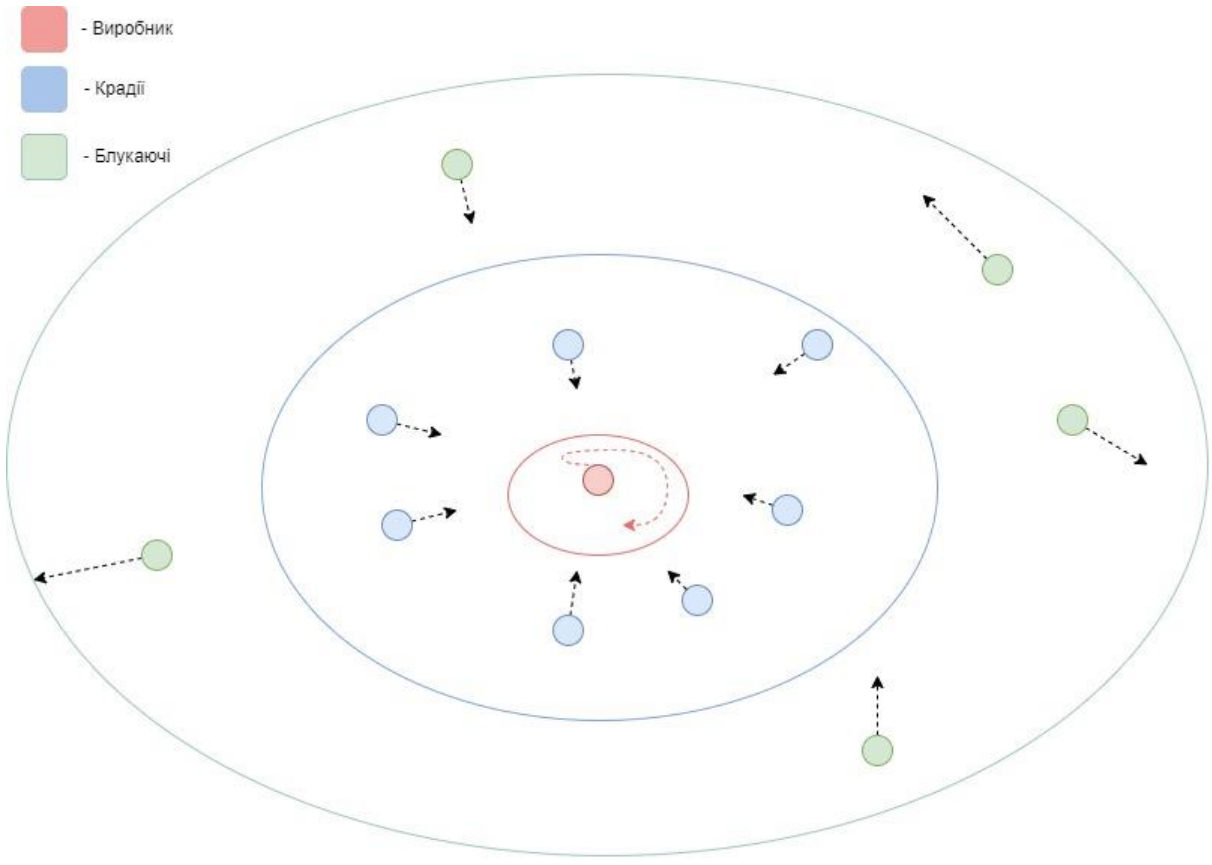


Рисунок 2.6 – Графічне представлення ідеї методу Producer-Scrounger

2.2.2.5 Алгоритм на основі алгоритму Firefly

Для покращення підмаршрутів в рамках даної задачі будемо використовувати алгоритм, що базується на ідеї алгоритму Firefly.

Класичний алгоритм Firefly базується на рисі світляків випромінювати світло. Наведемо три основні правила цього алгоритму [9]:

— усі світляки із популяції однієї статі, тому вони всі є однаково привабливими для одне одного;

— світляк що випромінює більшу кількість світла завжди буде приваблювати світляка, що випромінює меншу кількість світла;

— яскравість світляка визначається за допомогою цільової функції задачі, що розглядається.

Для розв’язання задачі, що розглядається у даній роботі необхідно модифікувати класичний підхід. Відстань між двома світляками буде вираховуватись за допомогою відстані Геммінга. Для руху світляків будемо використовувати функцію вставки (Insertion Function). Далі слідуючи такій же самій філософії що і у роботах [10, 11], світляки не мають напрямку руху. Замість цього вони рухаються використовуючи еволюційну стратегію. Таким чином кожен світляк буде застосовувати n разів функцію вставки генеруючи нові розв’язки. Після того як усі n рухів було зроблено – обирається найкращий і починається нова ітерація.

Висновки до розділу

У рамках даного розділу були описані математичні моделі Задачі 1 та Задачі 2. Для пошуку початкових розв’язків були обрані два методи: алгоритм на основі методу найближчого сусіда та алгоритм, що у свої основі використовує кластеризацію k-means та метод Sweep. Для покращення знайденого розв’язку задач були обрані такі методи: Swap та Insertion оператори, алгоритми на основі алгоритму Firefly та Producer-Scrounger, а також розроблений алгоритм мета якого покращити розв’язок шляхом розширення підмаршруту між базами.

3 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Вимоги до системи

На рисунку 3.1 представлені існуючі види вимог до системи, що можна поділити на такі категорії:

— бізнес-вимоги – описують потреби бізнесу на узагальненому рівні;

— користувацькі вимоги – описують цілі, які може досягти користувач використовуючи програмний продукт; зазвичай вимоги цієї групи документують у вигляді користувацьких історій, варіантів використання або сценаріїв;

— продуктові вимоги – описують як системі необхідно працювати, щоб задовільнити бізнес-вимоги та користувацькі вимоги; включають у себе функціональні та нефункціональні вимоги.

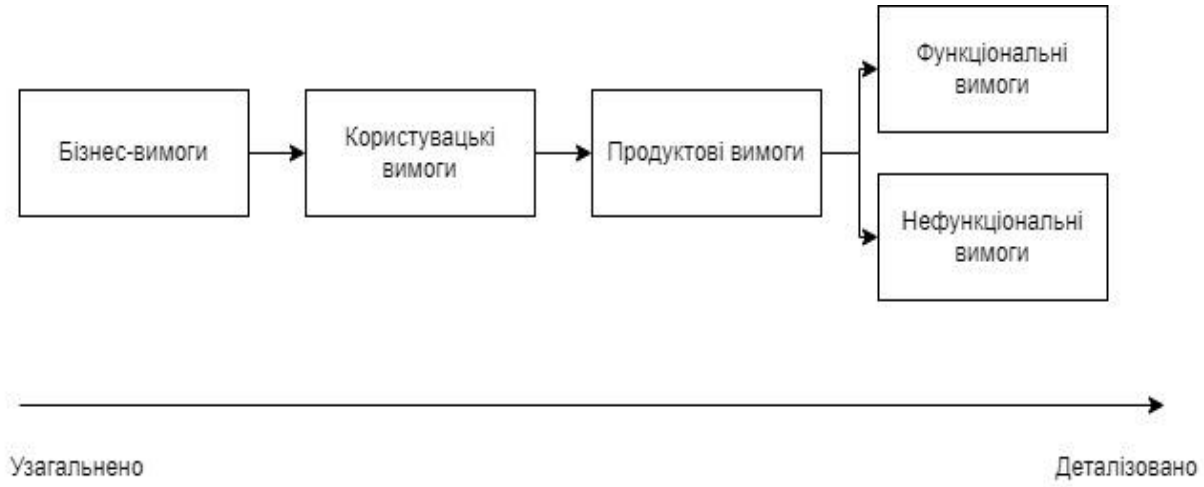


Рисунок 3.1 – Види вимог

3.1.1 Функціональні вимоги

Документування функціональних вимог має декілька вагомих переваг , а саме [12]:

— замовник та розробники мають єдине бачення задачі, що допомагає уникнути непорозумінь та заощадити час, що витрачається на комунікацію;

— проєкт стає більш передбачуваним, що дозволяє більш точно оцінити час на розробку.

Функціональні вимоги повинні бути простими, зрозумілими та недвозначними. У таблиці 3.1 представлені функціональні вимоги розроблюваної системи, що поділені на користувацькі історії.

Таблиця 3.1 – Опис функціональних вимог

Користувацька історія	Функціональні вимоги
Як користувач, я хочу мати можливість розв’язати задачу.	Система повинна надавати можливість вводу параметрів задачі.
	Система повинна надавати можливість вибору алгоритму.
	Система повинна надавати можливість імпорту координат баз та цілей із csv файлу.
	Система повинна надавати можливість налаштування випадкової генерації координат баз та цілей за допомогою визначення числових меж у яких повинні генеруватися дані.
	Система повинна повідомити користувача у разі вводу невалідних даних.
	Система повинна відображати розв’язок задачі у вигляді списку підмаршрутів.

	Система повинна відображати введені координати баз та цілей у графічному вигляді.
	Система повинна відображати розв’язок задачі у графічному вигляді.
Як користувач, я хочу мати можливість зберігати розв’язану задачу, переглядати її та видаляти.	Система повинна надавати можливість збереження задачі та її розв’язку після її розв’язання
	Система повинна надавати можливість перегляду збереженої задачі, а саме усіх вхідних та вихідних даних.
	Система повинна надавати можливість експорту вхідних даних задачі у csv форматі.
	Система повинна надавати можливість експорту вихідних даних збереженої задачі у csv форматі.
	Система повинна відображати координати цілей та баз збереженої задачі у графічному вигляді.
	Система повинна відображати розв’язок задачі у графічному вигляді.
	Система повинна надавати можливість видалення збереженої задачі.
Як користувач, я хочу мати можливість проводити експерименти порівняння ефективності алгоритмів.	Система повинна надавати можливість налаштування випадкової генерації координат для баз та цілей.
	Система повинна надавати можливість вводу параметрів задачі.

	Система повинна повідомити користувача у разу вводу невалідних даних.
	Система повинна представляти результати експерименту у табличному вигляді.
Як користувач, я хочу мати можливість проводити експерименти для дослідження однієї задачі або алгоритму.	Система повинна надавати можливість вводу вхідних параметрів задачі.
	Система повинна надавати можливість імпорту вхідних координат баз та цілей у csv форматі.
	Система повинна надавати можливість налаштування випадкової генерації вхідних координат цілей та баз.
	Система повинна надавати можливість налаштування числової величини кроку з якою буде змінюватися обраний параметр задачі на кожній ітерації.
	Система повинна повідомити користувача у разі вводу невалідних даних.
	Система повинна надавати можливість перегляду результатів експерименту у вигляді графіків.
Як користувач, я хочу мати можливість зберігати проведений експеримент, переглядати його та видаляти.	Система повинна надавати можливість збереження експерименту після його проведення.
	Система повинна надавати можливість перегляду проведеного експерименту, а саме усіх вхідних та вихідних даних.

	Система повинна надавати можливість експорту вхідних даних задачі у csv форматі.
	Система повинна надавати можливість експорту вихідних даних збереженого експерименту у csv форматі.
	Система повинна відображати результати збереженого експерименту у графічному вигляді.
	Система повинна надавати можливість видалення збереженого експерименту.
Як користувач, я хочу мати можливість переглядати усі збережені задачі та експерименти.	Система повинна надавати можливість перегляду усіх збережених задач та експериментів у вигляді списку.
	Система повинна надавати можливість фільтрації списку збережених елементів, для того, щоб відображати або тільки збережені задачі або тільки експерименти.
	Система повинна надавати можливість пошуку збереженого об’єкта за назвою.
	Система повинна надавати можливість видалення збереженого об’єкту.

3.1.2 Нефункціональні вимоги

Нефункціональні вимоги більш абстрактні ніж функціональні. Їх мета накласти обмеження на програмний продукт у термінах швидкодії, масштабованості, надійності, портативності, безпеки і так далі.

В рамках даної роботи нефункціональною вимогою до системи є те, що швидкість відповіді інтерфейсу програмного продукту на дії користувача не повинна перевищувати задані межі.

3.2 Опис бізнес-логіки системи

У додатку Б представлена декомпозиція цілей системи. Сценарії використання розробленої системи наведені у Додатку В. Вони описують що робить система не даючи деталей яким чином вона це робить. Такі сценарії є корисними як для замовників так і для розробників. Зі сторони замовника це корисний інструмент, щоб зрозуміти чи розробники правильно зрозуміли усі поставлені вимоги. В той же час для сторони розробників це зручне та структуроване представлення потоку подій, що повинні відбуватися у програмному продукті.

У додатку Г наведена діаграма діяльності для опису процесу розв’язання задачі у системі. Для виконання цієї операції з точки зору користувача доступні такі основні дії:

— введення параметрів задачі;
— введення координат баз та цілей за допомогою імпортування із файлу або випадкової генерації;
— ініціювання розв’язання задачі.

Система ж у свою чергу виконує такі дії як відображення вхідних та вихідних даних задачі у текстовому та графічному вигляді.

3.3 Вибір та обґрунтування елементів та технологій

Після того як усі вимоги були визначені та задокументовані необхідно на основі цього обрати технології які будуть використовуватися при розробці системи.

У рамках даного проєкту інтерфейс програмного продукту було вирішено реалізувати за допомогою ігрового рушія Unity. Такий вибір, більшою мірою, був зроблений по причині необхідності наявності графічного представлення розв’язку

задачі. Даний рушій є широко розповсюдженим і наразі налічує вже більше ніж два з половиною мільйони зареєстрованих розробників з усього світу [13]. В нього є такі переваги як підтримка 25 різних платформ, можливість працювати з 2D, 3D, VR та AR графікою, умовна безкоштовність, сильне ком'юніті та вбудовані засоби для аналітики дій користувача.

Так як основною мовою розробки з Unity є C#, то було вирішено використовувати цю мову програмування як основну. C# це сучасна мова програмування розроблена компанією Microsoft, що підходить для вирішення широкого спектру задач. На рисунку 3.2 представлені основні переваги C#.

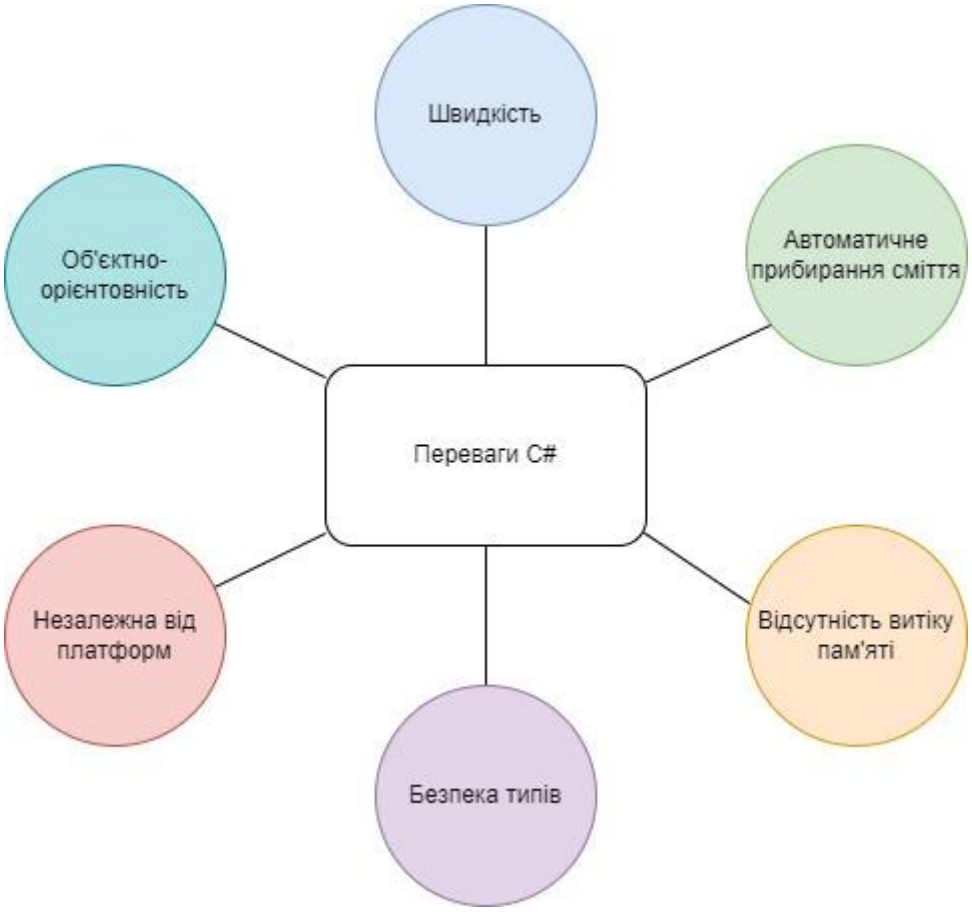


Рисунок 3.2 – Основні переваги мови програмування C#

Для сховища даних була обрана база даних SQLite. SQLite це бібліотека, що містить у собі імплементацію SQL рушія. Це база даних, що не вимагає жодного попереднього налаштування з боку адміністратора та не має ніяких сторонніх

залежностей. Крім того вона безсерверна, а дані зберігаються у одному файлі на диску у файловій системі. Схема того яким чином застосунок зазвичай з'єднується з даною базою даних наведений на рисунку 3.3 SQLite транзакції є ACID-сумісними. Тож, враховуючи вище сказане, SQLite добре підходить для даного проєкту [14].

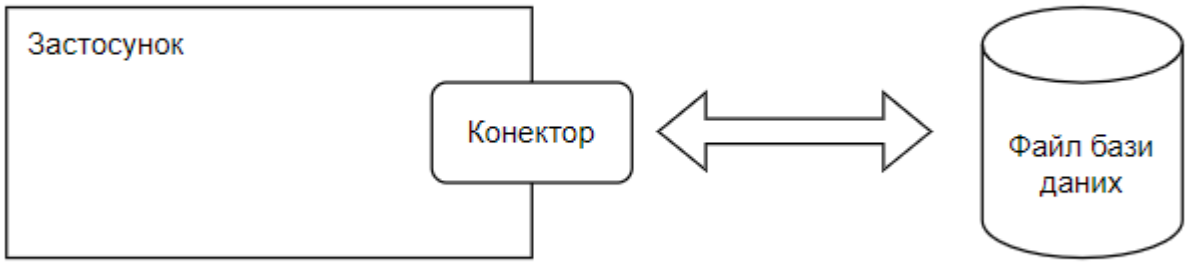


Рисунок 3.3 – Схема комунікації між застосунком та базою даних SQLite

3.4 Архітектура системи

Схема компонентів наведена у додатку Д. Далі у таблиці 3.2 дамо короткий опис основних функцій кожного компонента системи.

Таблиця 3.2 – Опис компонентів системи

Назва компонента	Основні функції
RandomDataGenerator	Випадкова генерація координат цілей та баз
CSVFileManager	Експорт та імпорт файлів з розширенням .csv
AlgoSolutionProvider	Містить імплементацію алгоритмів розв'язання задачі та методів проведення експериментів

DbService	Взаємодія із сховищем збережених задач та експериментів

Після короткого опису областей відповідальності компонентів системи, зупинимося більш детально на таких модулях як AlgoSolutionProvider та DbService, так як саме ці компоненти грають найважливішу роль у роботі даного програмного продукту.

DbService – це компонет системи, що є мікросервісом, тож десктопний застосунок розроблений на рушії Unity комунікує з ним за допомогою http запитів. У таблиці 3.3 наведений опис ресурсів API даного сервісу.

Таблиця 3.3 – Опис API сервісу DBService

Назва ресурсу	Метод	Опис ресурсу	Параметри	Опис параметрів	Вихідні дані
/solutions	GET	Отримати усі збережені задачі	-	-	Список з усіма збереженими задачами
	POST	Зберегти задачу	Об’єкт задачі, що необхідно зберегти	-	Об’єкт збереженої задачі
/solutions/{id}	GET	Отримати одну збережену задачу	id	Ідентифікатор задачі	Об’єкт задачі
	DELETE	Видалити одну збережену задачу	id	Ідентифікатор задачі	-

/experiments	GET	Отримати усі збережені експерименти	-		Список з усіма збереженими експериментами
	POST	Зберегти експеримент	Об’єкт експерименту, який необхідно зберегти		Об’єкт збереженого експерименту
/experiments/{id}	GET	Отримати збережений експеримент по ідентифікатору	id	Ідентифікатор експерименту	Об’єкт експерименту
	DELETE	Видалити експеримент по ідентифікатору	id	Ідентифікатор експерименту	-

Структурна схема класів модуля AlgoSolutionProvider, що відповідає за імплементацію алгоритмів, наведена у додатку Е. У таблиці 3.4 наведемо опис класів даного модуля.

Таблиця 3.4 – Опис класів компонента AlgoSolutionProvider

Назва класу	Опис
BaseAlgo	Базовий клас алгоритму, що містить допоміжні функції, що можуть бути корисні для імплементації декількох алгоритмів, що розглядаються. Наприклад такі методи як отримати відстань між двома цілями, встановити рядок матриці часів у безкінечність, встановити стовпець матриці часів у безкінечність, дістати маршрут між базами, отримати відстань Геммінга, згенерувати випадковий розв’язок задачі, дізнатися чи досяжна ціль, випадково обрати ціль, випадково обрати депо
ExtendAlgo	Клас, що імплементує логіку алгоритму розширення маршруту між базами
FireflyAlgo	Клас, який імплементує логіку алгоритму Firefly
GreedyAlgo	Знаходження розв’язку задачі за допомогою алгоритму NNA
LocalOptimizer	Надає імплементацію алгоритмів Swap та Insertion
ProducerScroungerAlgo	Надає імплементацію алгоритму Producer-Scrounger
SweepAlgo	Знаходження розв’язку за допомогою алгоритму Sweep

AlgoEnum	Містить перерахунок усіх наявних алгоритмів
ProblemInput	Модель, що описує вхідні дані для розв’язку задачі. Містить такі поля: матриця часів, координати, час обслуговування БПЛА, час на переїзд ТЗ між базами, місткість БПЛА
ProblemOutput	Модель, що містить вихідні дані в результаті розв’язку задачі. Містить такі поля: підмаршрути, значення цільової функції, час на обслуговування
Route	Модель, що описує один підмаршрут розв’язку задачі. Містить такі поля: упорядкований список цілей, час витрачений на обліт підмаршруту
ExperimentResult	Модель, що містить результати проведення експерименту, а саме такі поля як максимальний час виконання експерименту, мінімальний час, середній час, кількість разів коли алгоритм досяг найкращого результату, кількість разів, коли алгоритм досяг найгіршого результату, сумарний час виконання експерименту та назва алгоритму
SolutionProvider	Слугує за вхідну точку взаємодії із компонентом. Отримає на вхід назву алгоритму та всі необхідні вхідні дані. Далі використовуючи наявні класи у

	компоненті повертає результат (розв’язок задачі)

3.5 Розробка бази даних

Реляційна схема бази даних наведена у Додатку Ж. У таблицях 3.5 - 3.9 представлений детальний опис таблиць бази даних.

Таблиця 3.5 – Опис таблиці Problems

Назва поля	Тип поля	Опис
Id (PK)	INTEGER	Унікальний ідентифікатор (первинний ключ)
InputCoordinatedFilePath	TEXT	Шлях до файлу із вхідними даними
ParameterSetId	INTEGER	Зовнішній ключ на таблицю ParameterSets

Таблиця 3.6 – Опис таблиці Algorithms

Назва поля	Тип поля	Опис
Id (PK)	INTEGER	Унікальний ідентифікатор (первинний ключ)
AlgorithmName	TEXT	Назва алгоритму
AlgorithmDescription	TEXT	Опис алгоритму

Таблиця 3.7 – Опис таблиці Solutions

Назва поля	Тип поля	Опис
Id (PK)	INTEGER	Унікальний ідентифікатор (первинний ключ)
Name	TEXT	Назва розв’язку
Description	TEXT	Опис розв’язку

OutputSolutionFilePath	TEXT	Шлях до файлу із знайденим розв’язком
CreatedAt	TEXT	Дата збереження розв’язку
ProblemId (FK)	INTEGER	Зовнішній ключ на таблицю Problems
AlgorithmId (FK)	INTEGER	Зовнішній ключ на таблицю Algorithms

Таблиця 3.8 – Опис таблиці ParameterSets

Назва поля	Тип поля	Опис
Id (PK)	INTEGER	Унікальний ідентифікатор (первинний ключ)
DroneSpeed	REAL	Швидкість БПЛА
DroneCapacity	REAL	Місткість БПЛА
RechargeSpeed	REAL	Час на обслуговування БПЛА
AutoTime	REAL	Час на пересування ТЗ між базами

Таблиця 3.9 – Опис таблиці Experiments

Назва поля	Тип поля	Опис
Id (PK)	INTEGER	Унікальний ідентифікатор (первинний ключ)
Name	TEXT	Назва експерименту
Description	TEXT	Опис експерименту
MinX	REAL	Мінімальне значення координати x для усіх задач експерименту

MaxX	REAL	Максимальне значення координати x для усіх задач експерименту
MinY	REAL	Мінімальне значення координати y для усіх задач експерименту
MaxY	REAL	Максимальне значення координати y для усіх задач експерименту
ProblemCount	INTEGER	Кількість задач в експериментi
OutputExperimentalFilePath	TEXT	Шлях до файлу із вихідними даними результатів експерименту
CreatedAt	TEXT	Дата збереження експерименту
ParameterSetId	INTEGER	Зовнішній ключ на таблицю ParameterSets

3.6 Інструкція користувача

Структурна схема сторінок інтерфейсу програмного продукту наведена у додатку К. Дана схема допомагає у навігації по інтерфейсу програмного продукту.

При запуску інтерфейсу програми користувач бачить головне меню, що зображене на рисунку 3.4.

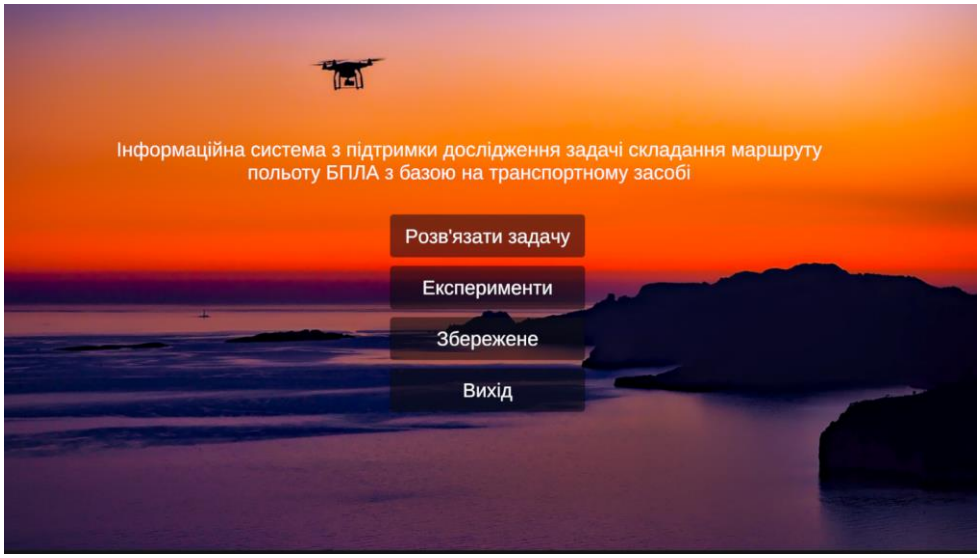


Рисунок 3.4 – Вигляд головного меню

Для того, щоб розв’язати задачу із головного меню необхідно натиснути кнопку “Розв’язати задачу”. Дана дія повинна перенаправити користувача на сторінку розв’язку задачі. На даній сторінці можна обрати яким чином вводити координати цілей та баз – із файлу з розширенням .csv або генерувати випадковим чином. Для того, щоб імпортувати csv файл необхідно натиснути кнопку “Завантажити”. Після цього відкриється вікно з вибором фалу де необхідно обрати файл csv з координатами у валідному форматі. Приклад валідного формату координат представлений на рисунку 3.5. Слід зазначити, що в даній системі використовуються прямокутні координати, а не географічні. Даний тип координат часто використовується у топографії або геодезії, так як вимірювати відстань у метрах або кілометрах іноді зручніше ніж у секундах або хвилинах. Існує безліч застосунків, де можна конвертувати географічні координати у прямокутні та навпаки. З одним із них, що називається PGC Coordinate Converter можна ознайомитися перейшовши по електронному ресурсу [15].

A	B	C	D	E
4.83	0.49	2.99	1.81	
6.07	1.27	-3.75	1.93	
5.29	3.39			
1.19	3.09			
-1.57	0.99			
-4.71	0.53			
-4.89	-1.85			
-7.23	-1.97			
-2.71	-2.93			

Рисунок 3.5 – Приклад файлу із координатами баз та цілей

Після вибору файлу бази та цілі повинні відобразитися у вікні зліва. Далі необхідно ввести параметри задачі. Після цього потрібно обрати алгоритм, яким користувач хочу розв’язувати задачу. Після цих дій можна натискати кнопку “Розрахувати”. Приклад вікна із валідними вхідними даними та результатом розв’язку зображений на рисунку 3.6.

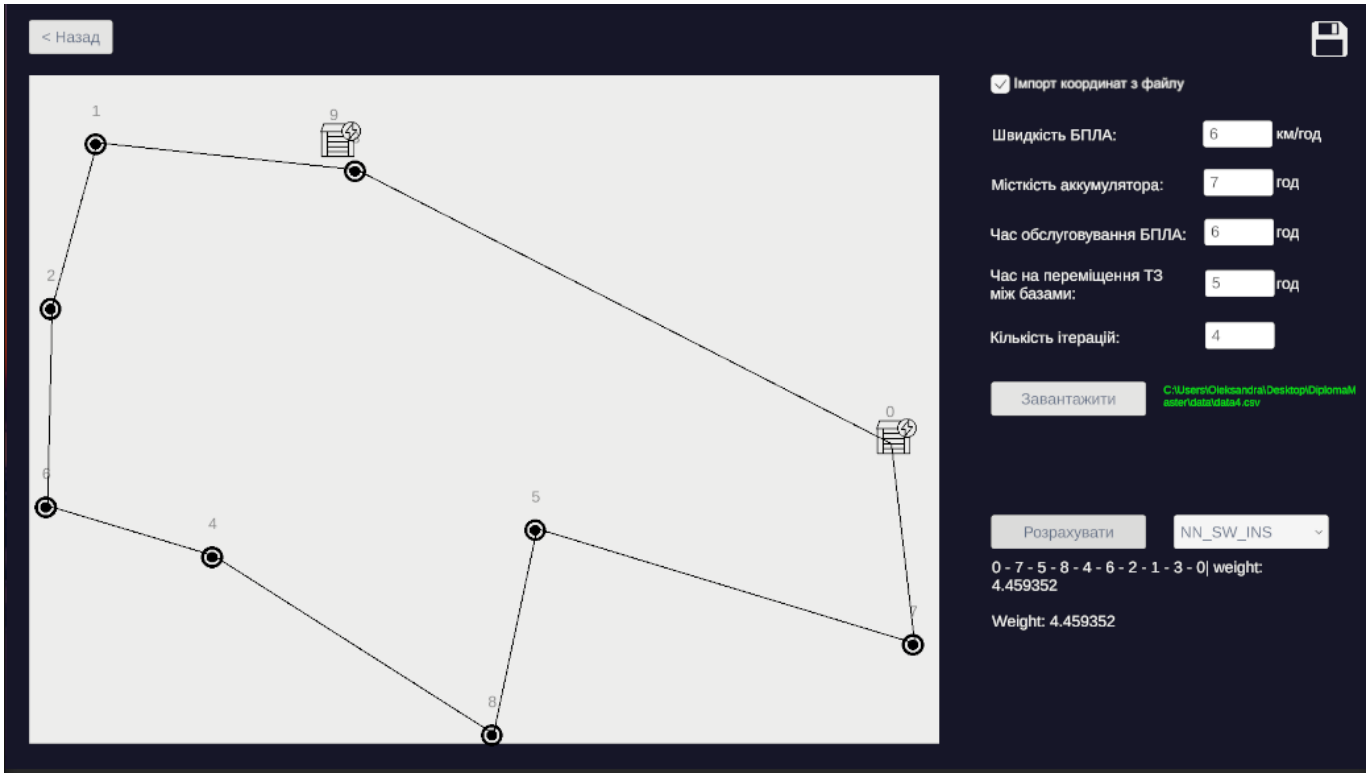


Рисунок 3.6 – Вигляд сторінки розв’язку задачі із результатом розв’язку

Щоб зберегти задачу потрібно натиснути на іконку “Зберегти” у верхньому правому кутку. Після цього з’явиться вікно, де необхідно ввести назву задачі та її опис.

Для того, щоб згенерувати координати баз та цілей випадковим чином необхідно зробити неактивним чекбокс “Імпорт координат з файлу”. Після цього необхідно ввести параметри для генерації координат, та натиснути на кнопку “Show Data”. Після цього цілі та бази повинні відобразитись у вікні ліворуч. Приклад правильно згенерованих координа. наведений на рисунку 3.7.

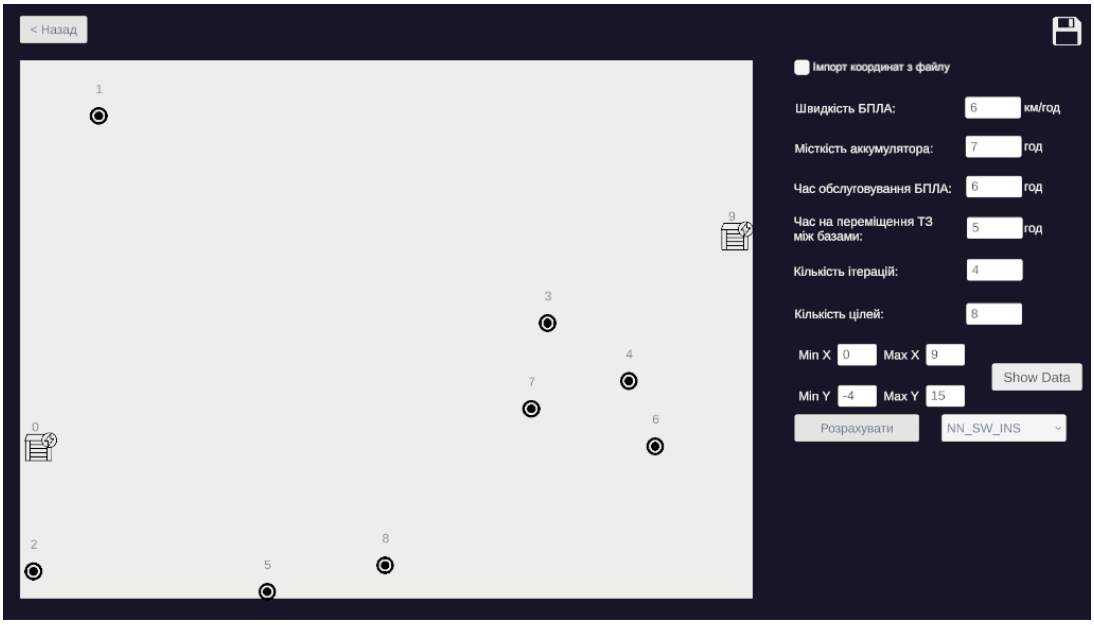


Рисунок 3.7 – Приклад графічного відображення координат баз та цілей, що були згенеровані випадковим чином

Для того, щоб провести експеримент необхідно із сторінки головного меню натиснути на кнопку “Експерименти”. Після цього з’явиться сторінка проведення експерименту на порівняння ефективності алгоритмів. Приклад цієї сторінки зображений на рисунку 3.8.

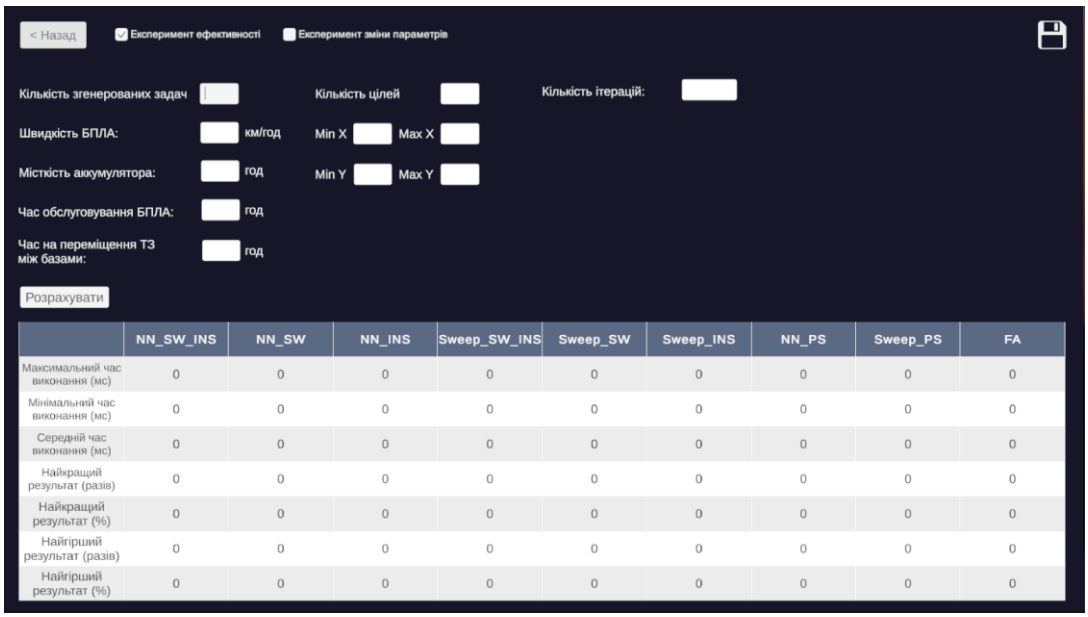
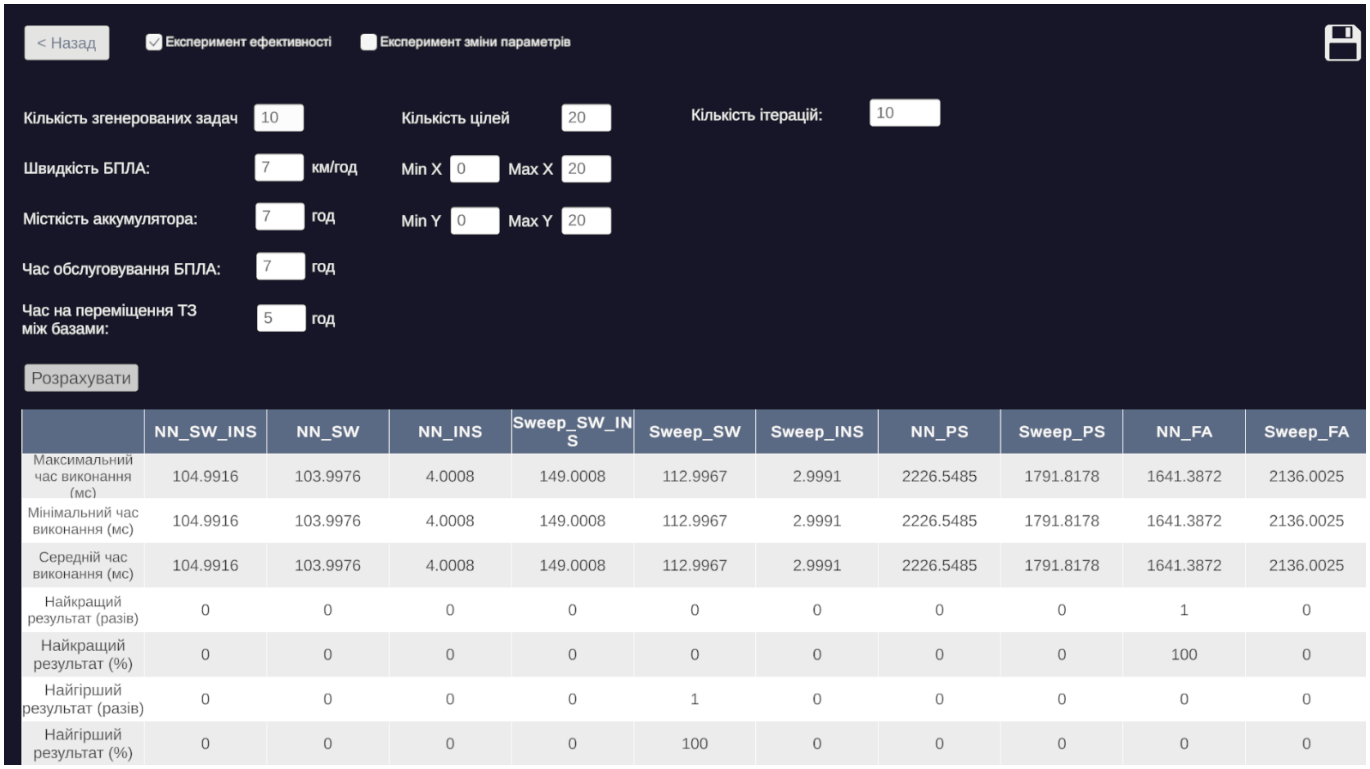


Рисунок 3.8 – Приклад сторінки для проведення експерименту ефективності алгоритмів

Після введення усіх необхідних вхідних даних необхідно натиснути кнопку “Розрахувати”, щоб провести експеримент. Приклад результату експерименту можна побачити на рисунку 3.9.



	NN_SW_INS	NN_SW	NN_INS	Sweep_SW_INS	Sweep_SW	Sweep_INS	NN_PS	Sweep_PS	NN_FA	Sweep_FA
Максимальний час виконання (мс)	104.9916	103.9976	4.0008	149.0008	112.9967	2.9991	2226.5485	1791.8178	1641.3872	2136.0025
Мінімальний час виконання (мс)	104.9916	103.9976	4.0008	149.0008	112.9967	2.9991	2226.5485	1791.8178	1641.3872	2136.0025
Середній час виконання (мс)	104.9916	103.9976	4.0008	149.0008	112.9967	2.9991	2226.5485	1791.8178	1641.3872	2136.0025
Найкращий результат (разів)	0	0	0	0	0	0	0	0	1	0
Найкращий результат (%)	0	0	0	0	0	0	0	0	100	0
Найгірший результат (разів)	0	0	0	0	1	0	0	0	0	0
Найгірший результат (%)	0	0	0	0	100	0	0	0	0	0

Рисунок 3.9 – Приклад виведення результатів експерименту

Для того, щоб провести експеримент, що включає у себе покрокову зміну одного або багатьох параметрів задачі необхідно на сторінці проведення експериментів зробити активним чекбокс “Експеримент зміни параметрів”. Після цього повинна з’явитися сторінка зображена на рисунку 3.10, де необхідно ввести параметри для налаштування експерименту.

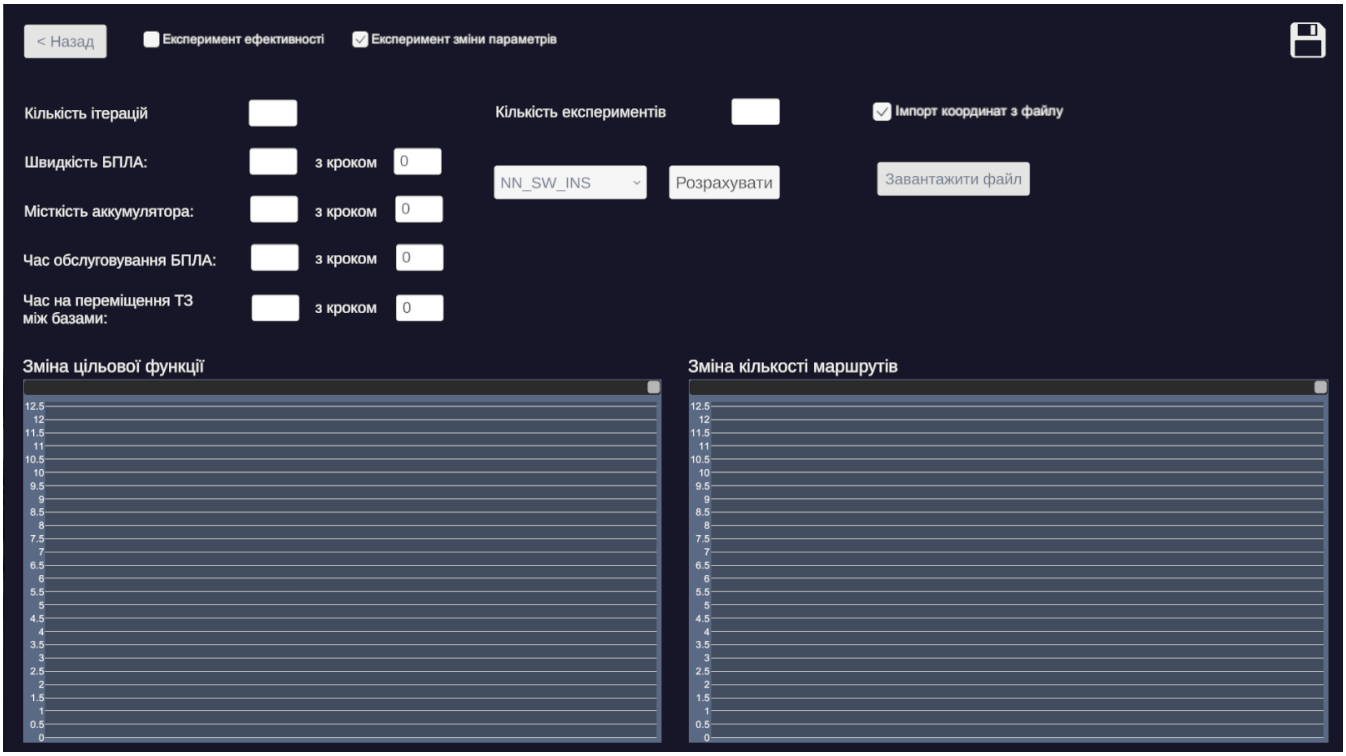


Рисунок 3.10 – Вигляд сторінки проведення експерименту за допомогою зміни параметрів задачі

3.7 Тестування системи

Окрім юніт та інтеграційних тестів системи було проведено також мануальне тестування. Результати цього тестування представлені у вигляді тестових кейсів у таблиці 3.10. Тестові кейси представляють собою набір дій, що повинні бути зроблені у певному порядку для того щоб перевірити коректність роботи певного функціоналу.

Таблиця 3.10 – Тестові кейси системи

Номер	Опис	Кроки	Очікуваний результат	Висновок
1	Розв’язання задачі з валідними вхідними даними експортованими із csv файлу	1. перейти на сторінку розв’язання задачі; 2. ввести валідні параметри задачі; 3. натиснути кнопку “Завантажити”; 4. обрати csv файл з валідними координатами у вікні для вибору файлу; 5. обрати один із запропонованих алгоритмів у полі з випадаючим меню; 6. натиснути кнопку “Розрахувати”.	Розв’язок задачі відображається на графічному екрані у лівій частині сторінки. Розв’язок задачі відображається у текстовій формі під кнопкою “Розрахувати”	Успіх
2	Розв’язання задачі з валідними вхідними даними згенерованими випадковим чином	1. перейти на сторінку розв’язання задачі; 2. ввести валідні параметри задачі; 3. ввести валідні налаштування випадкової генерації координат для цілей та баз;		Успіх

		4. обрати один із запропонованих алгоритмів у полі з випадаючим меню; 5. натиснути кнопку “Розрахувати”.		
3	Розв’язання задачі з невалідними вхідними даними	1. перейти на сторінку розв’язання задачі; 2. ввести невалідні параметри задачі; 3. натиснути кнопку “Завантажити”.	Зверху графічного вікна з’явилося повідомлення, що вхідні дані є невалідними	Успіх
4	Збереження задачі з валідними даними	1. перейти на сторінку розв’язання задачі; 2. розв’язати задачу; 3. натиснути на кнопку у вигляді іконки “Зберегти”; 4. ввести ім’я та опис задачі у вікні, що з’явилося; 5. натиснути кнопку “Зберегти”.	Користувач успішно зберіг задачу	Успіх

5	Збереження задачі з невалідними даними	1. перейти на сторінку розв’язання задачі; 2. розв’язати задачу; 3. натиснути на іконку у вигляді “Зберегти”; 4. залишити порожніми ім’я та опис задачі; 5. натиснути кнопку “Зберегти”.	Користувач отримав повідомлення, що введені дані не валідні	Успіх
6	Проведення експерименту на ефективність	1. перейти на сторінку проведення експериментів на ефективність; 2. заповнити усі необхідні поля валідними даними; 3. натиснути на кнопку “Розрахувати”.	З’явилися результати проведення експерименту у вигляді таблиці	Успіх
7	Проведення експерименту на дослідження зміни параметрів	1. перейти на сторінку проведення експериментів за допомогою зміни параметрів; 2. заповнити валідними даними усі необхідні поля;	Результати проведення експерименту відобразилися у вигляді графіків	Успіх

		3. натиснути на кнопку “Розрахувати”.		
8	Збереження експерименту на ефективність з валідними даними	1. перейти на сторінку проведення експериментів на ефективність; 2. провести експеримент; 3. натиснути на кнопку “Зберегти”, що має вид іконки у верхньому правому кутку. 4. У вікні, що з’явилося ввести валідне ім’я експерименту та опис. 5. Натиснути кнопку “Ок”.	Експеримент був успішно збережений	Успіх
9	Пошук збережених задач	1. Перейти на сторінку “Збережене”;	На екрані у списку збережених	Успіх

		2. Обрати варіант “Задачі” на панелі фільтрації 3. Увести в пошук назву заздалегідь збереженої задачі; 4. Натиснути кнопку “Пошук”, що має вигляд іконки праворуч від поля пошуку.	відобразилася шукана задача	
10	Пошук збережених експериментів	1. Відкрити сторінку “Збережене”; 2. Обрати варіант “Експерименти” на панелі фільтрації зверху 3. Ввести у поле пошуку назву збереженого експерименту; 4. Клацнути на кнопку “Пошук”.	На сторінці у списку збережених об’єктів видно шуканий експеримент	Успіх

Висновки до розділу

У рамках третього розділу були визначені функціональні вимоги до системи, що були поділені на користувацькі історії, а також нефункціональні вимоги. У рамках опису бізнес-логіки системи були представлені сценарії використання та діаграма

активності для опису процесу розв’язання задачі. Після формування вимог були обрані технології за допомогою яких буде розроблятися система. Було прийняте рішення розроблювати інтерфейс системи на ігровому рушії Unity, оскільки він надає широкі можливості у роботі з 2D та 3D графікою, що може бути корисним при імплементації графічного відображення вхідних та вихідних даних задачі. Імплементація бекенд частини системи була створена за допомогою мови програмування C#. Для сховища даних, а саме збережених задач та експериментів було вирішено використовувати базу даних SQLite .

4 РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТАЛЬНИХ ДОСЛІДЖЕНЬ

У таблиці 4.1 наведені назви алгоритмів, та використані у них методи розв’язання задачі. Тобто кожен алгоритм являє собою комбінацію декількох методів, що описані у підпункті 2.2. Назва алгоритму включає у себе скорочену назву кожного із використаних у ньому методів. Виключенням є алгоритм розширення підмаршруту між базами, оскільки він використовується у всіх алгоритмах, то його назва не фігурую у назва жодного алгоритму, що представлені у таблиці 4.1.

Таблиця 4.1 – відповідності між розробленими алгоритмами та використаними у них методами

Назва алгоритму	Назва методу						
	NNA	Sweep	Swap	Insertion	Алгоритм розширення підмаршруту між базами	Producer-Scrounger	Firefly
NN_SW_INS	+		+	+	+		
NN_SW	+		+		+		
NN_INS	+			+	+		
Sweep_SW_INS		+	+	+	+		
Sweep_SW		+	+		+		
Sweep_INS		+		+	+		
NN_PS	+				+	+	
Sweep_PS		+			+	+	
NN_FA	+				+		+
Sweep_FA		+			+		+

4.1 Експеримент 1

Експеримент 1 полягає у порівнянні швидкодії та ефективності роботи алгоритмів один з одним, а також у аналізі характеру зміни часу виконання алгоритму при зміні розмірності задачі (кількості цілей для обльоту). В даному контексті міра ефективності визначається як відсоток кількості разів, де алгоритм дав найкраще значення цільової функції у порівняння з іншими алгоритмами. Вхідні дані для проведення даного експерименту наведені у таблиці 4.2. Величину кількості цілей будемо змінювати від 10 до 50 із кроком 10.

Таблиця 4.2 – Вхідні дані для Експерименту 1

Назва поля	Значення поля
Кількість прогонів	5
Кількість випадкового згенерованих задач	50
Швидкість БПЛА	10
Місткість акумулятора БПЛА	9
Час обслуговування БПЛА	0.2
Час на переміщення ТЗ між базами	0.4
Min X	0
Min Y	0
Max X	20
Max Y	20
Кількість ітерацій	10

У таблиці 4.3 наведені результати Експерименту 1, де для кожного експерименту представлені такі метрики як максимальний час роботи алгоритму, мінімальний час роботи алгоритму, середній час роботи алгоритму, найкращий та найгірший результат (значення цільової функції) у кількісній та відсотковій величині.

Таблиця 4.3 – Результати Експерименту 1

Назва алгоритму	К-сть цілей	Досліджувані метрики						
		Макс. час вик.	Мін. час вик.	Серед. час вик.	Найк. рез. (разів)	Найк. рез. (%)	Найг. рез. (разів)	Найг. рез. (%)
NN_SW_INS	10	26	6.96	10.29	26	52	0	0
	20	67.99	13	29.6	16.5	33	0	0
	30	147.51	32.45	62.30	19	38	0	0
	40	193.24	54.09	91.85	21	42	0	0
	50	227	61.64	111.57	18	38	0	0
NN_SW	10	18	6.95	9.69	1	2	0	0
	20	62.99	15.99	0.99	1.5	3	0	0
	30	139.31	33.99	60.13	3.5	7	0	0
	40	276.19	44.04	95.51	4	8	0	0
	50	320.51	62.54	109.27	3.5	7	0	0
NN_INS	10	13	8.08	11.03	6.66	13.32	0	0
	20	21.97	12.16	19.01	0.5	1	0	0
	30	51.63	31.2	45.6	1.5	3	0	0
	40	95.01	42.09	63.6	4.5	9	0	0
	50	121.9	83.5	92.6	4	8	0	0
Sweep_SW_INS	10	29	6.99	16.63	2.66	5.32	4.33	8.66
	20	87	21.81	35.53	0.5	1	4	8
	30	94.96	43	58	0	0	6	12
	40	206.04	61.93	108.73	0	0	1	2
	50	368.47	93.96	149.48	0	0	1	2
Sweep_SW	10	40	6	17	0	0	20.66	41.32
	20	68	21.77	35.12	0	0	26.5	53
	30	124.99	33.99	59.97	0	0	19.5	39
	40	193	61.54	104.09	0	0	27	54

	50	385.42	81.99	159.41	0	0	27	54
Sweep_INS	10	14	8.49	11.02	0.33	0.66	12.66	25.32
	20	17.09	13.4	15.76	0	0	17.5	35
	30	32.45	21.3	29.4	0	0	24.5	29
	40	59.8	39.6	43.9	0	0	22	44
	50	120.54	66.7	81.34	0	0	22	44
NN_PS	10	703.15	243.78	407.03	1.66	3.32	8.33	16.66
	20	2985.3	859.55	1643.94	8	16	0	0
	30	5868.9	1703.5	3579.05	6.5	13	0	0
	40	14340	2698.2	6354.19	5	10	0	0
	50	15248	4759.4	8134.4	3.5	7	0	0
Sweep_PS	10	1368.9	220.74	513.75	8.33	16.64	1	2
	20	1991.9	700.17	1048.61	0.5	1	1.5	3
	30	2916.4	1390.3	1830.45	0	0	0	0
	40	5241.7	2093.4	3222.81	0	0	0	0
	50	7487.7	3395.7	4741.42	0	0	0	0
NN_FA	10	434.89	243.41	309.41	1.66	3.32	2.66	5.32
	20	1093.5	395.96	554.48	21	42	0	0
	30	1931	803.12	1233.59	19.5	39	0	0
	40	2744.8	1490.5	1748.14	15.5	31	0	0
	50	3757.8	1741.6	2225.34	21	42	0	0
Sweep_FA	10	943.01	238.04	555.85	4	8	0.33	0.66
	20	1990.9	782.3	1154.21	1.5	3	0.5	1
	30	3826.9	1161.9	1929.78	0	0	0	0
	40	11652	2264.8	3563.39	0	0	0	0
	50	8795.5	2669.3	5058.32	0	0	0	0

4.1.1 Дослідження швидкодії алгоритмів у рамках Експерименту 1

На рисунках 4.1 – 4.10 представлена залежність мінімального, максимального та середнього часу виконання кожного алгоритму від розмірності задачі (кількості цілей для обльоту), а на рисунку 4.11 – порівняння середнього часу виконання алгоритмів при різних розмірностях задачі. Легко бачити, що алгоритм NN_PS є найповільнішим. Він розв'язує задачу за 8600 мілісекунд при розмірності у 50 цілей. Sweep_FA займає другу позицію з кінця після NN_PS та розв'язує задачу за 5100 мілісекунд при тій же самій розмірності. У порівнянні з іншими алгоритмами, відносно повільними також можна вважати алгоритми Sweep_PS та NN_FA. Варто зауважити, що і алгоритм Producer-Scrounger і алгоритм Firefly використовують ідею застосування популяцій для пошуку розв'язку і, оскільки, не завжди вдається з першого разу програмним шляхом згенерувати популяцію (підмаршрут), що є придатним для використання, це сповільнює процес розв'язку задачі цими алгоритмами. У контексті порівняння швидкодії фаворитами стали алгоритми, що використовують оператори Sweep або Insertion для покращення розв'язку. Як бачимо із результатів алгоритми пошуку початкового розв'язку задачі майже не впливають на час виконання алгоритмів порівнюючи із методами локального пошуку.

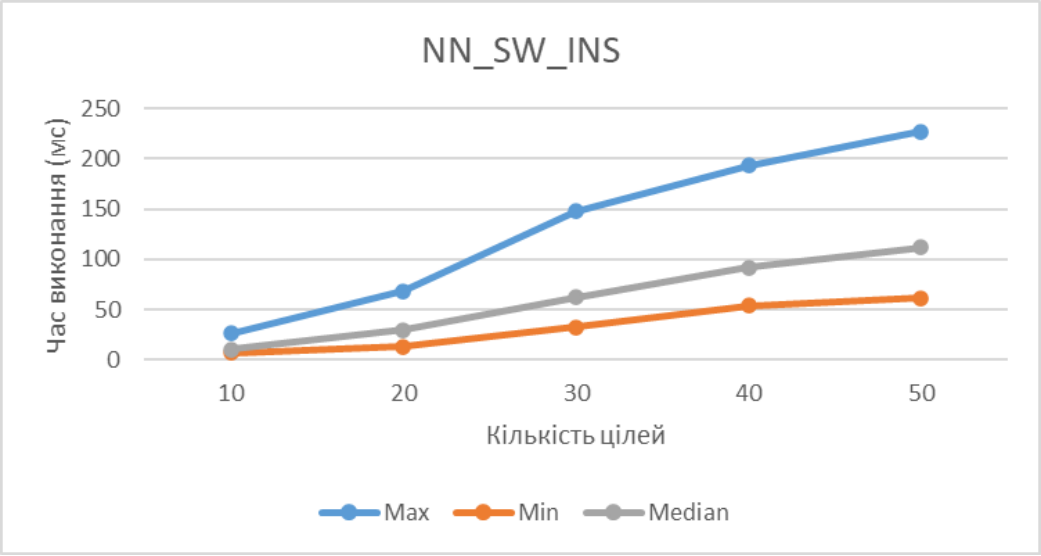


Рисунок 4.1 - Максимальний, мінімальний та середній час виконання алгоритму NN_SW_INS у рамках Експерименту 1

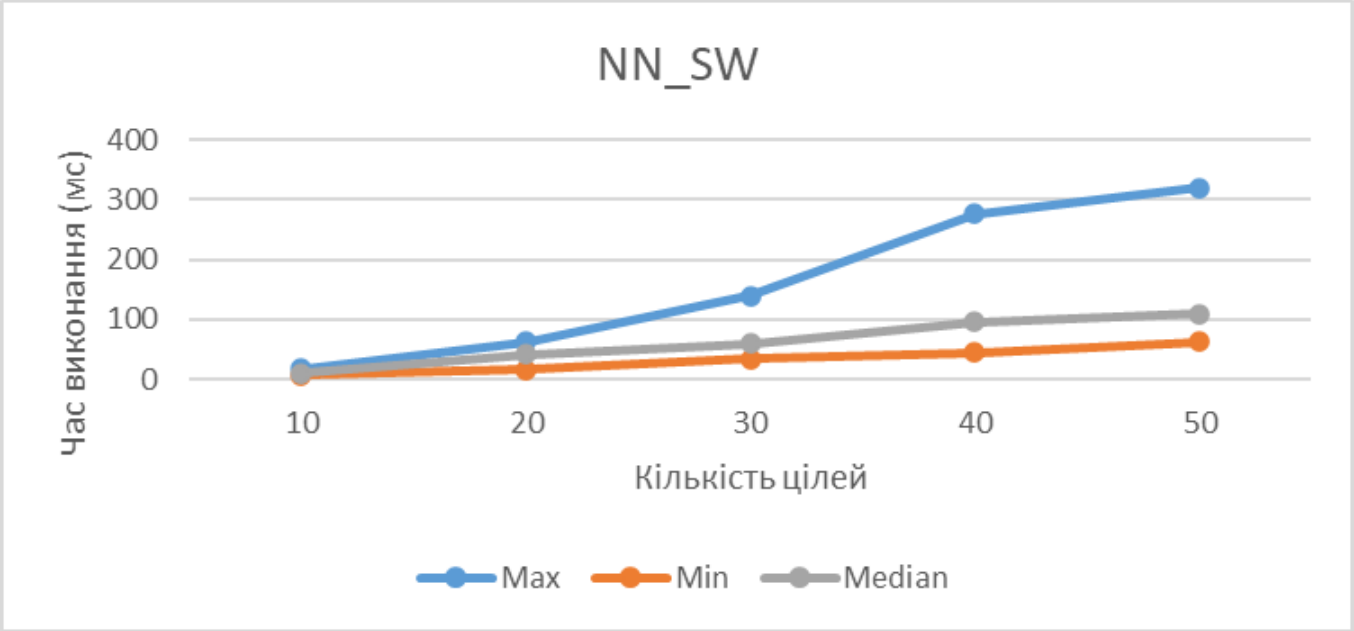


Рисунок 4.2 - Максимальний, мінімальний та середній час виконання алгоритму NN_SW у рамках Експерименту 1

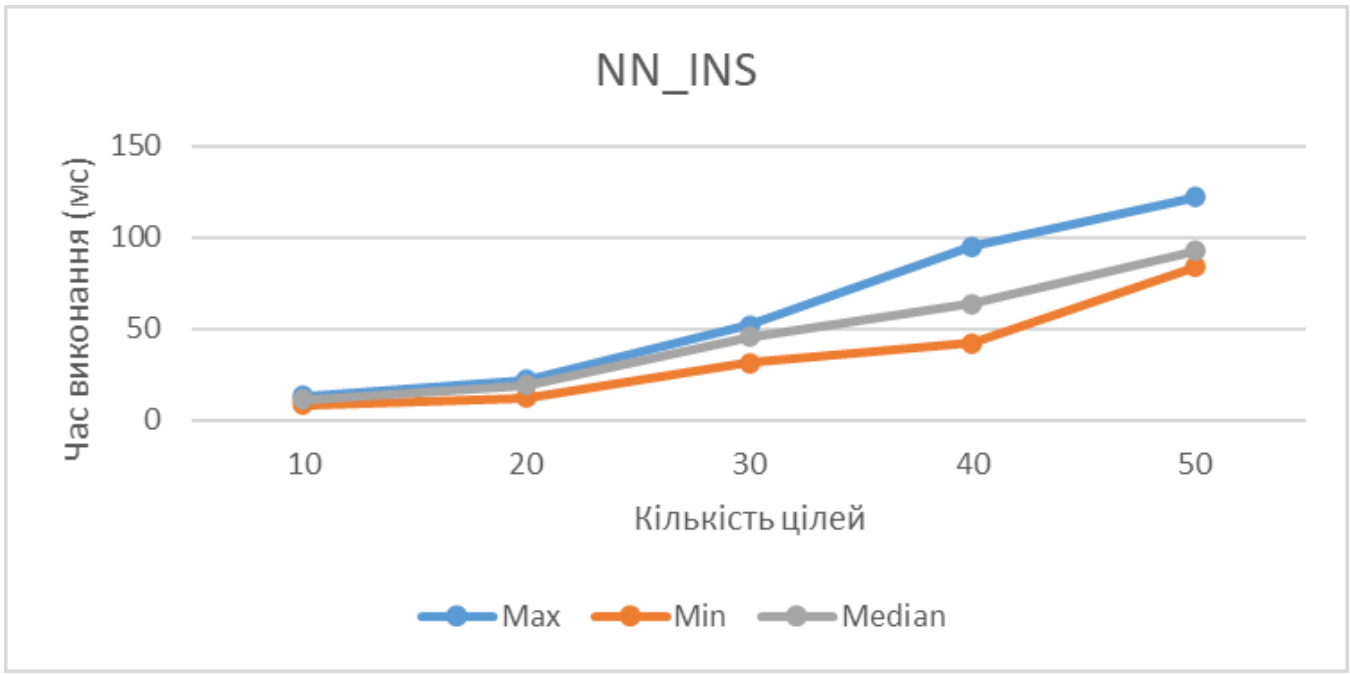


Рисунок 4.3 - Максимальний, мінімальний та середній час виконання алгоритму NN_INS у рамках Експерименту 1

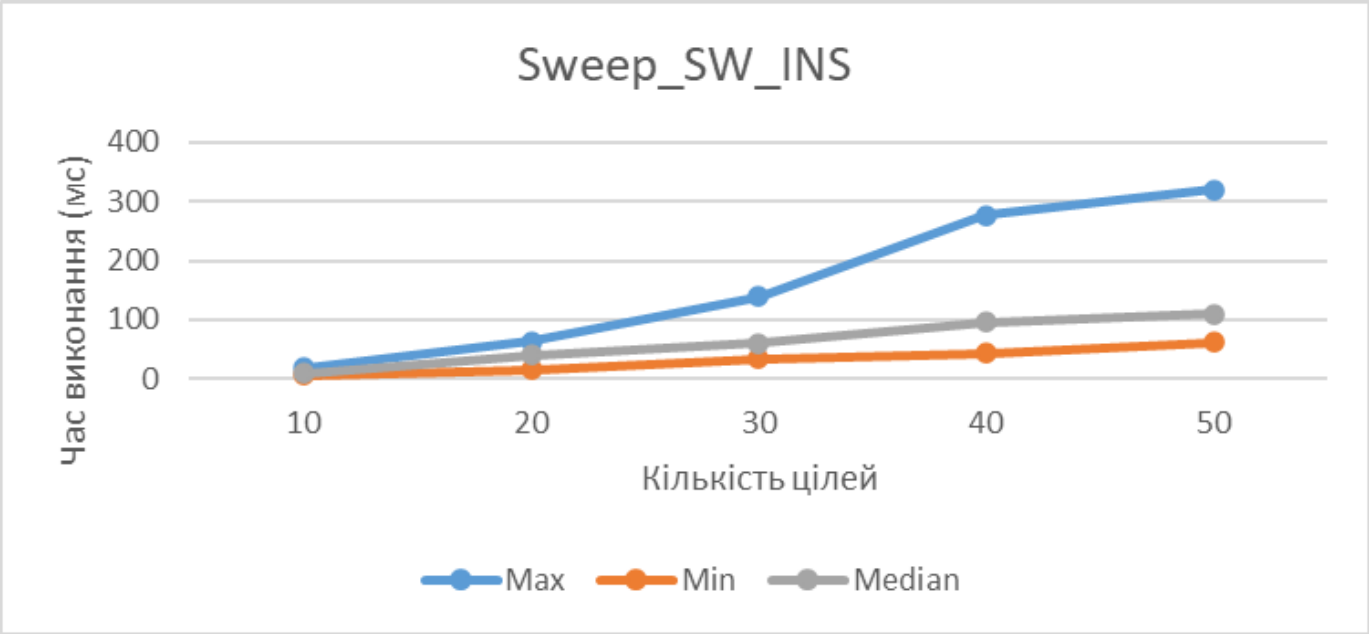


Рисунок 4.4 - Максимальний, мінімальний та середній час виконання алгоритму Sweep_SW_INS у рамках Експерименту 1

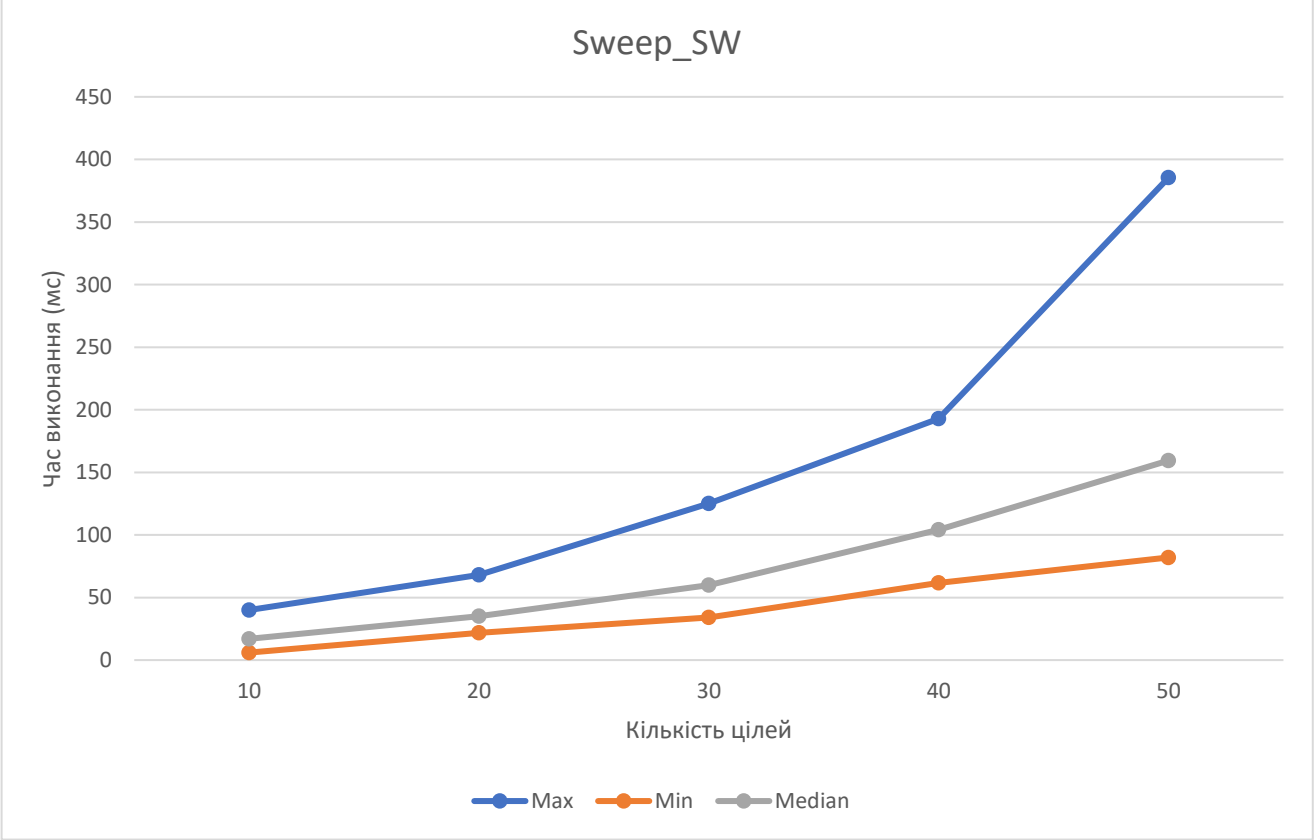


Рисунок 4.5 - Максимальний, мінімальний та середній час виконання алгоритму Sweep_SW у рамках Експерименту 1

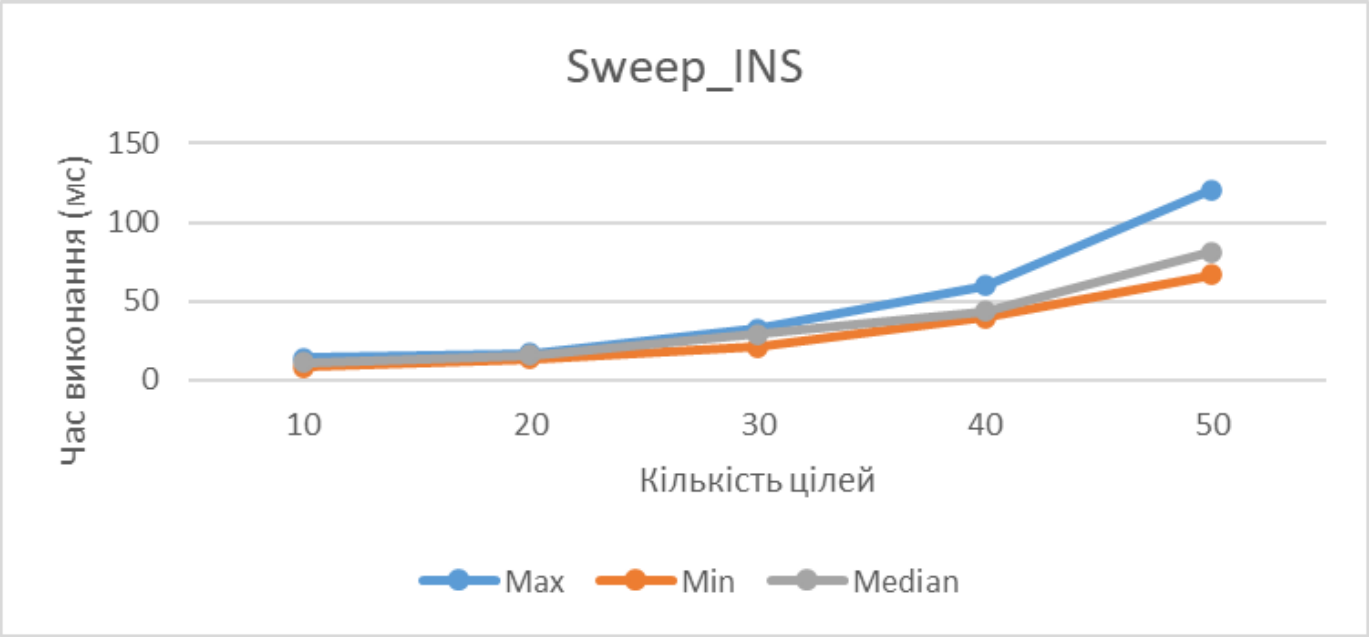


Рисунок 4.6 - Максимальний, мінімальний та середній час виконання алгоритму Sweep_INS у рамках Експерименту 1

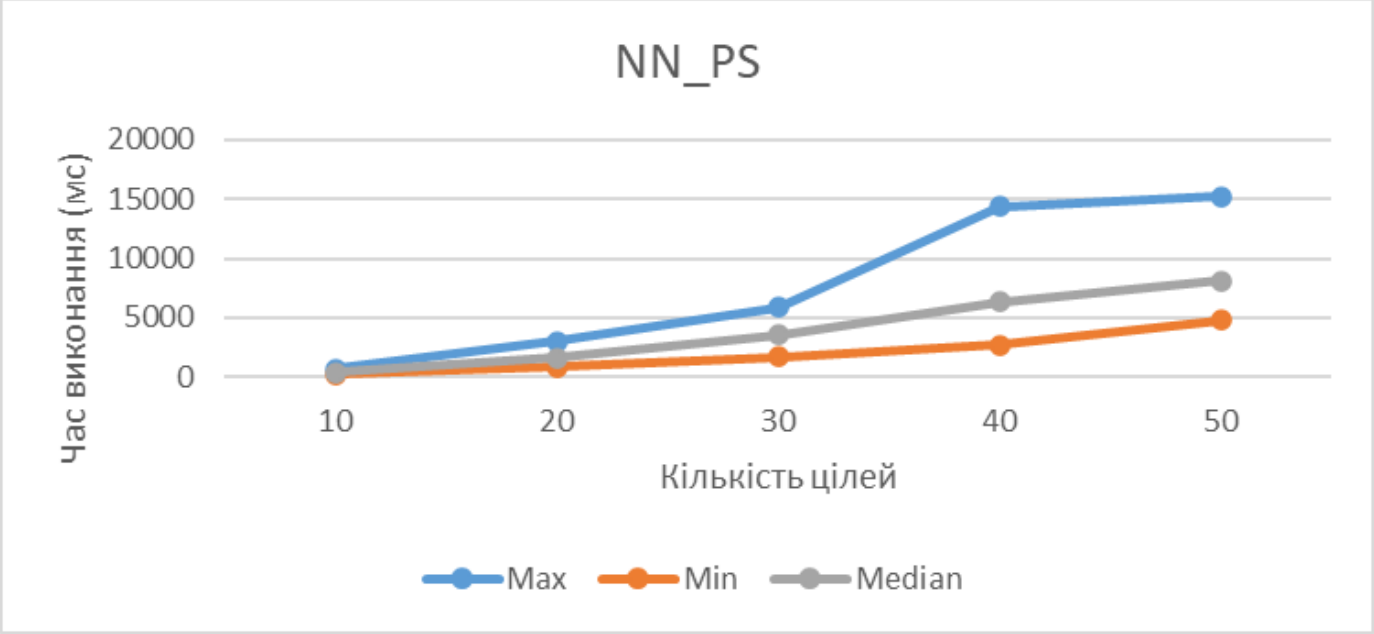


Рисунок 4.7 - Максимальний, мінімальний та середній час виконання алгоритму NN_PS у рамках Експерименту 1

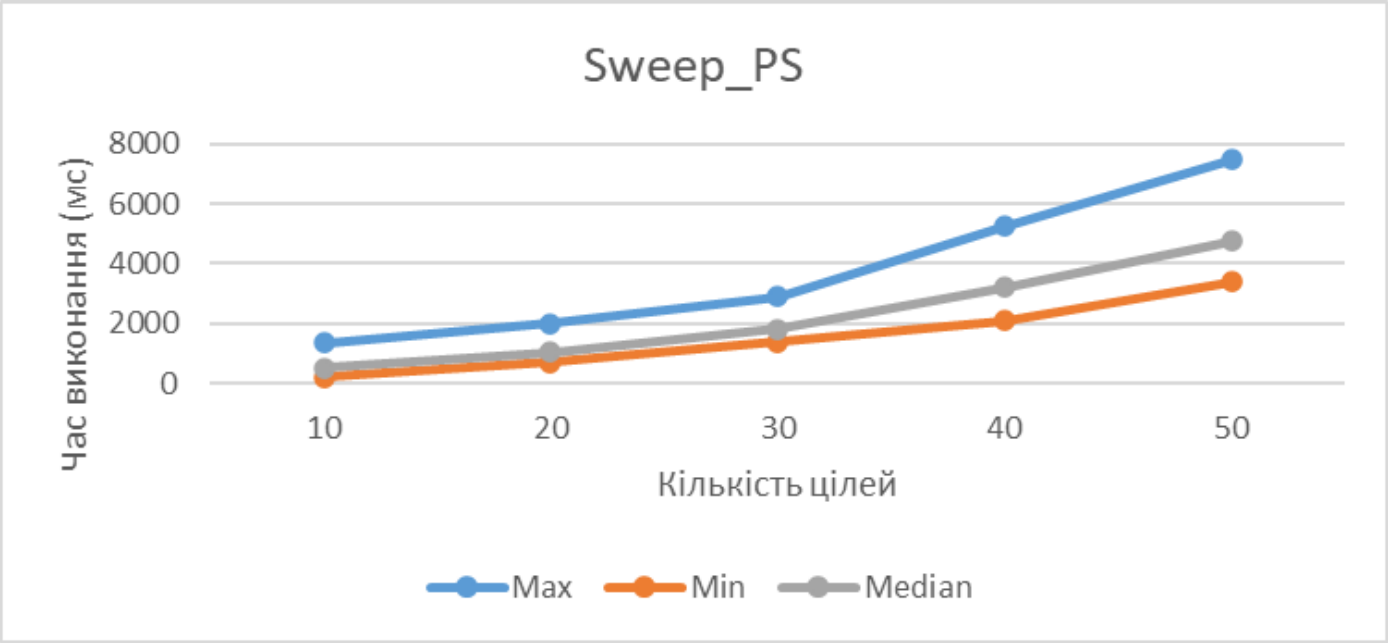


Рисунок 4.8 - Максимальний, мінімальний та середній час виконання алгоритму Sweep_PS у рамках Експерименту 1

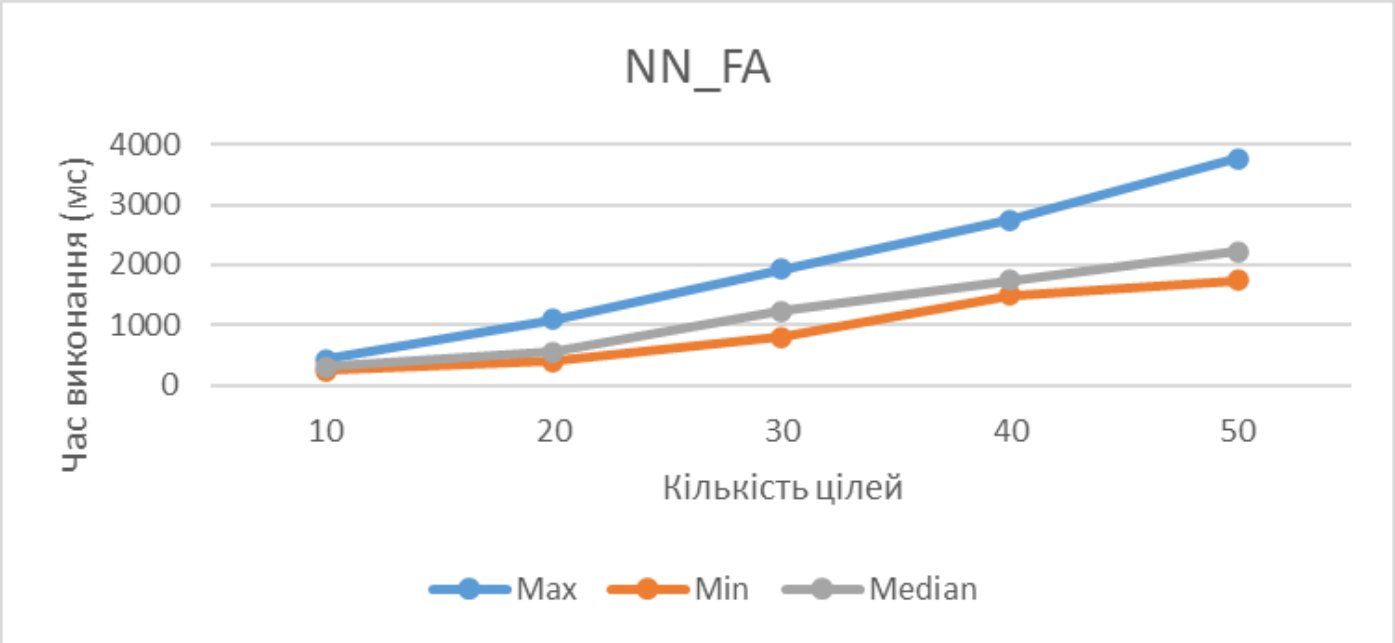


Рисунок 4.9 - Максимальний, мінімальний та середній час виконання алгоритму NN_FA у рамках Експерименту 1

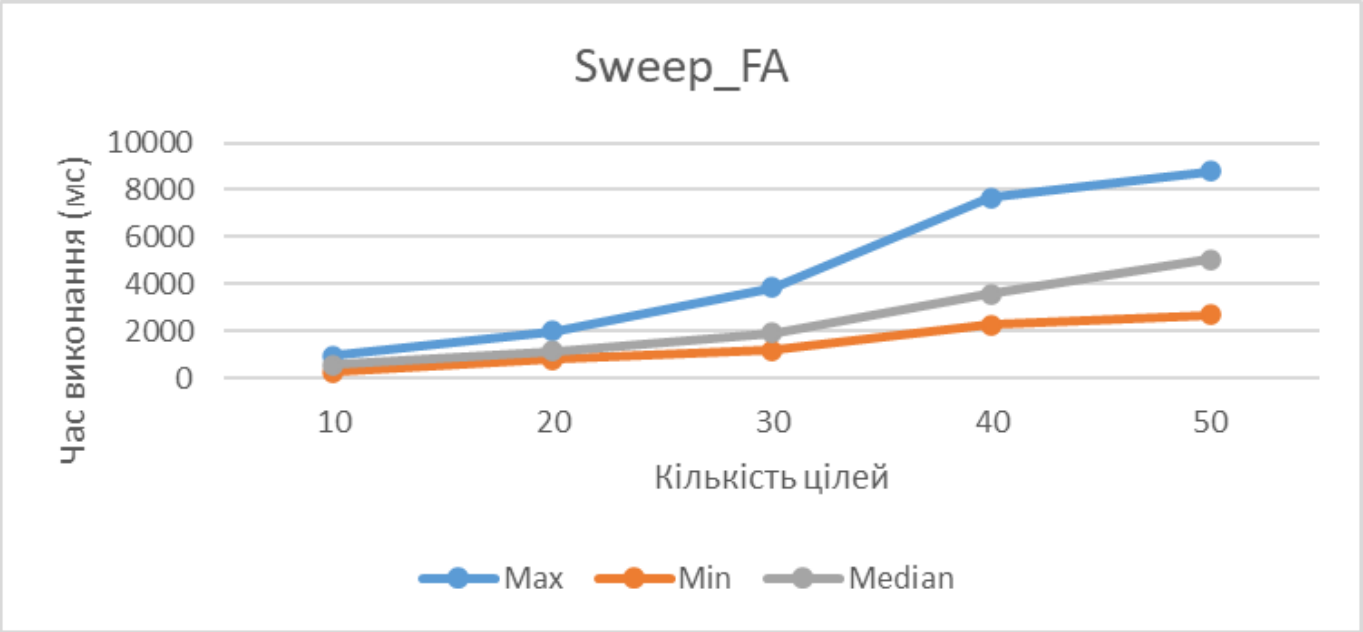


Рисунок 4.10 – Максимальний, мінімальний та середній час виконання алгоритму Sweep_FA у рамках Експерименту 1

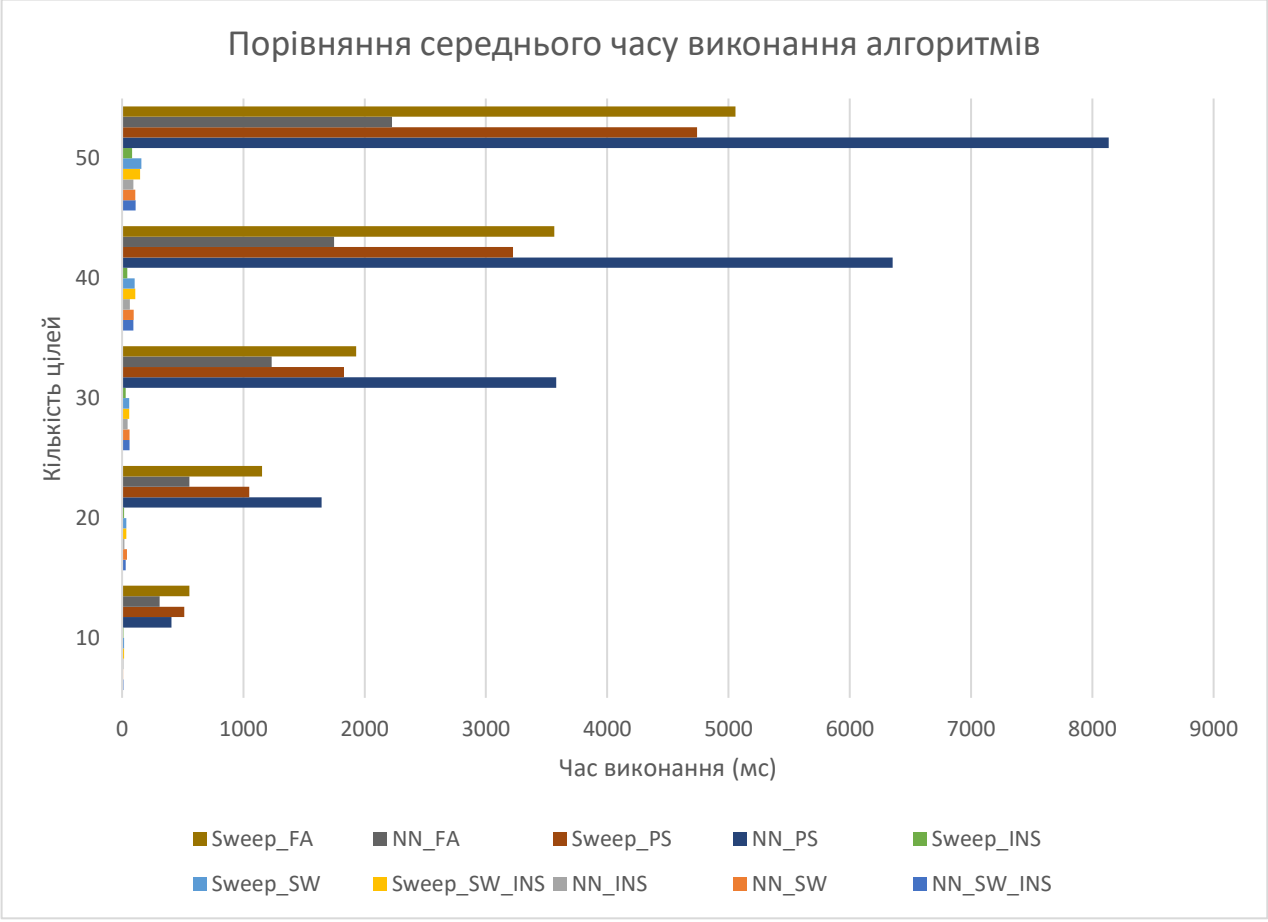


Рисунок 4.11 – Порівняння середнього часу виконання алгоритмів у рамках Експерименту 1

4.1.2 Дослідження ефективності алгоритмів у рамках Експерименту 1

У рамках дослідження ефективності алгоритмів наведені рисунки 4.12 та 4.13, де порівнюються найкращі та найгірші алгоритми у сенсі значення цільової функції відповідно. Як видно із результатів експерименту найбільш вдалим можна вважати алгоритм NN_SW_INS, що показує стабільно хороший результат навіть при максимальній розмірності задачі у 50 цілей. Також гарні результати з невеликим відривом показує алгоритм NN_FA. Як видно, обидва ці алгоритми використовують алгоритм на основі методу найближчого сусіда для пошуку початкового розв'язку задачі. Тож цей метод у комбінації із методом Firefly або операторами Swap + Insertion надає частіше найкращі значення цільової функції ніж інші представлені алгоритми. Також можна зробити висновок, що такі оператори як Swap та Insertion варто використовувати у парі, так як використання лише одного із двох операторів знижує шанс отримати найкращу цільову функцію у 4 рази. Найгірші значення цільової функції частіше усього давали алгоритми, що використовували метод Sweep для побудови початкового розв'язку задачі.

Рисунок 4.12 – Порівняння ефективності алгоритмів у контексті найкращого значення цільової функції у рамках Експерименту 1

Рисунок 4.13 – Порівняння ефективності алгоритмів у контексті найгіршого значення цільової функції у рамках Експерименту 1

4.2 Експеримент 2

У рамках Експерименту 2 досліджується як зміна кількості ітерацій для розв'язання задачі впливає на час виконання алгоритмів та і їх ефективність. У таблиці 4.3 представлені вхідні дані для експерименту при зміні кількості ітерацій від 10 до 50 із кроком 10.

Таблиця 4.3 – Вхідні дані для Експерименту 2

Назва поля	Значення поля
Кількість прогонів	5
Кількість випадкового згенерованих задач	50
Швидкість БПЛА	10
Місткість акумулятора БПЛА	9
Час обслуговування БПЛА	0.2

Час на переміщення ТЗ між базами	0.4
Min X	0
Min Y	0
Max X	20
Max Y	20
Кількість цілей	20

У таблиці 4.4 наведені результати Експерименту 2. Досліджувані метрики співпадають із метриками із Експерименту 1.

Таблиця 4.4 – Результати Експерименту 2

Назва алгоритму	К-сть ітерацій	Досліджувані метрики						
		Макс. час вик.	Мін. час вик.	Серед. час вик.	Найк. рез. (разів)	Найк. рез. (%)	Найг. рез. (разів)	Найг. рез. (%)
NN_SW_INS	10	67.89	13	29.7	16.7	33	0	0
	20	127.5	25.6	59.4	9.5	19	0	0
	30	256.7	40.72	84.36	13	26	0	0
	40	589	54.8	118.19	14.5	29	0	0
	50	322.37	67.9	134.59	12.5	25	0.5	1
NN_SW	10	62.93	15.97	0.91	1.5	3	0	0
	20	127	24.06	57.25	3	6	0	0
	30	252.9	42.13	89.82	0.5	1	0.5	1
	40	443.6	61.1	122.9	4.5	9	0.5	1
	50	399.96	72.9	141.44	3	6	0	0
NN_INS	10	21.91	12.18	19.01	0.6	1	0	0
	20	5.99	0.95	2.33	1	2	0	0
	30	8.25	0.89	2.91	1.5	3	0.5	1
	40	7.9	0.99	2.93	0.5	1	1	2
	50	19	1.68	3.9	2.5	5	0	0

Sweep_SW_INS	10	87	21.84	35.54	0.4	1	4	8
	20	166.4	46.9	80	0	0	4.5	9
	30	295	68.4	109.1	0.5	1	5	10
	40	595.1	93	156.9	0	0	2	4
	50	492.3	115.02	183.27	1	2	3	6
Sweep_SW	10	68	21.78	35.13	0	0	26.4	53
	20	266.04	43.5	78.1	0	0	22	44
	30	264.54	68.42	109.16	0	0	26	52
	40	470.78	87.9	154.32	0	0	29	58
	50	350.6	111.5	179.8	0	0	30	60
Sweep_INS	10	14	8.47	11.03	0.33	0.67	12.67	25.32
	20	7.02	0.98	2.04	0	0	21.5	43
	30	3.93	0.99	2.06	0	0	15	30
	40	6.04	1	2.68	0	0	16.5	33
	50	5.99	1	2.72	0	0	15	30
NN_PS	10	2985.5	859.52	1643.91	8	16	0	0
	20	7974	2080	3850	5.6	11	0	0
	30	9570.7	3169	5180	9.5	19	0	0
	40	25192	3429	7304	9	18	0	0
	50	13907	5129	8589	4	8	0	0
Sweep_PS	10	1991.6	700.16	1048.62	0.5	1	1.5	3
	20	6547	1550	2524	1	2	2	4
	30	4980	2159	3256	1	2	2.5	5
	40	9943	2807	4561	2.5	5	1	2
	50	13469	3523	5729	2	4	1.5	3
NN_FA	10	1093.4	395.94	554.49	21	42	0	0
	20	3356	840.6	1483.3	28.5	57	0	0
	30	3554	1213.2	1747.1	18	36	0	0
	40	5783.2	1653.1	2358.7	16.5	33	0	0

	50	6997.8	2074.8	3156.5	19.5	39	0	0
Sweep_FA	10	1990.7	782.2	1154.22	1.5	3	0.5	1
	20	4398.8	1621.9	2611.9	1.5	3	0	0
	30	5598.9	2529.4	3476.5	6	12	0.5	1
	40	8924.9	3258.3	4815.7	2.5	5	0	0
	50	9935.1	4085.5	5878.7	5.5	11	0	0

4.2.1 Дослідження швидкодії алгоритмів у рамках Експерименту 2

На рисунках 4.14 – 4.23 наведені максимальний, мінімальний та середній час виконання кожного із алгоритмів. На рисунку 4.24 представлене порівняння середнього часу виконання алгоритмів між собою. На основі отриманих результатів усі алгоритми можна умовно розбити на дві групи – “швидкі” та “повільні”. До “повільних” входять такі алгоритми як Sweep_FA, NN_FA, Sweep_PS, NN_PS, а до “швидких”, відповідно, усі інші. Як видно, усі алгоритми, що мають низьку швидкодію використовують методи ройового інтелекту та принцип популяцій для локального покращення.

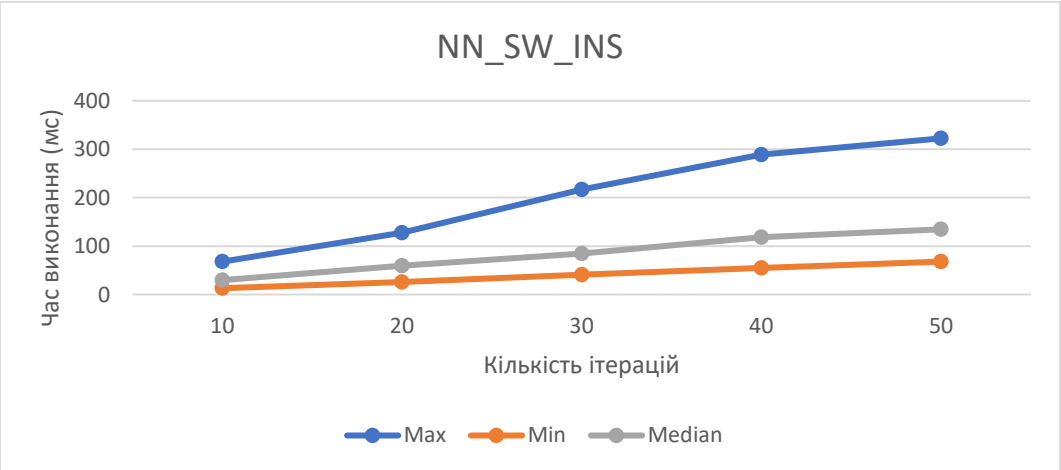


Рисунок 4.14 - Максимальний, мінімальний та середній час виконання алгоритму NN_SW_INS у рамках Експерименту 2

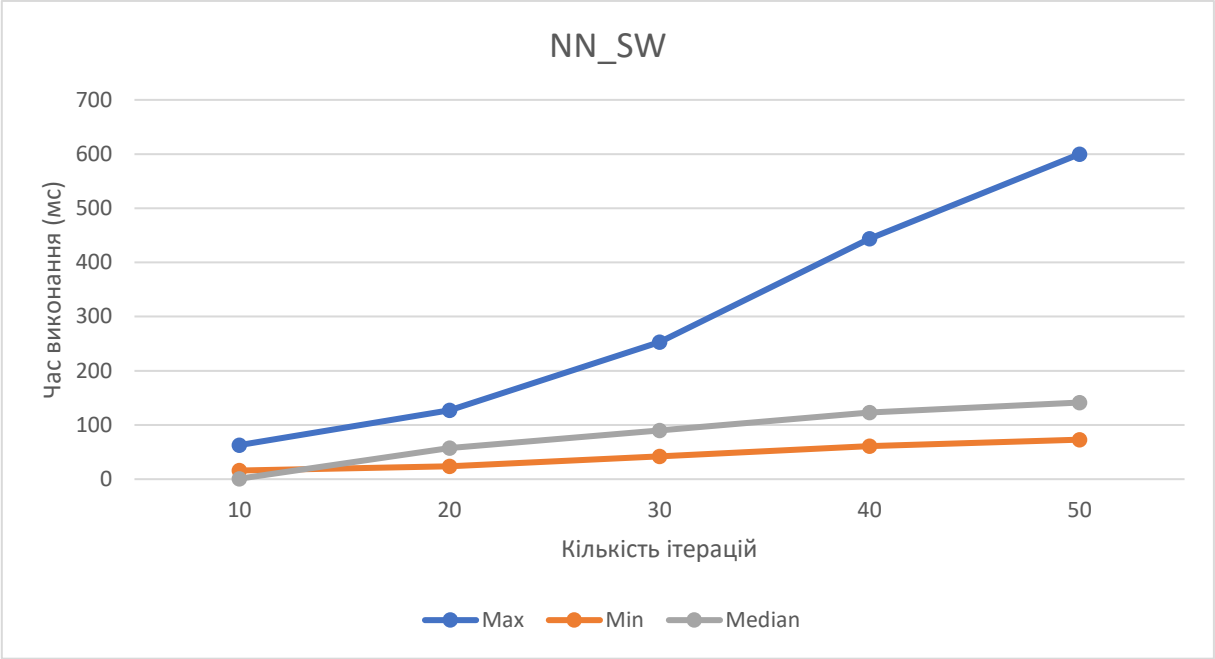


Рисунок 4.15 - Максимальний, мінімальний та середній час виконання алгоритму NN_SW у рамках Експерименту 2

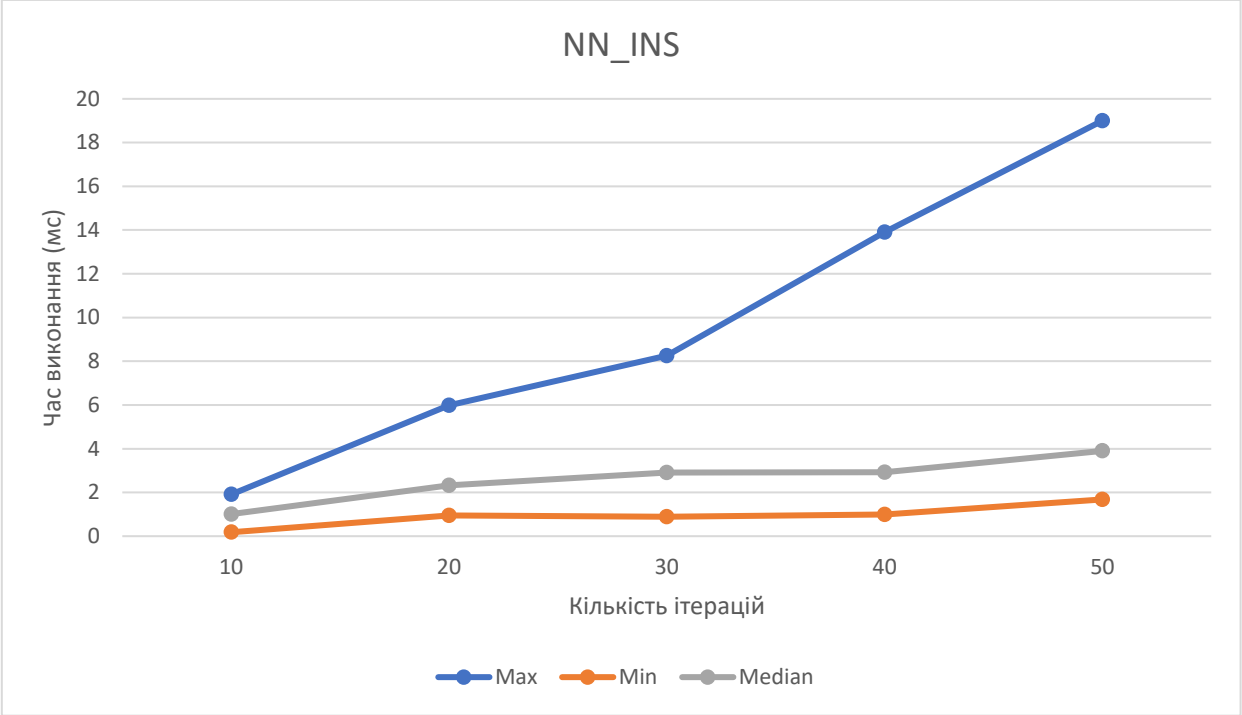


Рисунок 4.16 - Максимальний, мінімальний та середній час виконання алгоритму NN_INS у рамках Експерименту 2

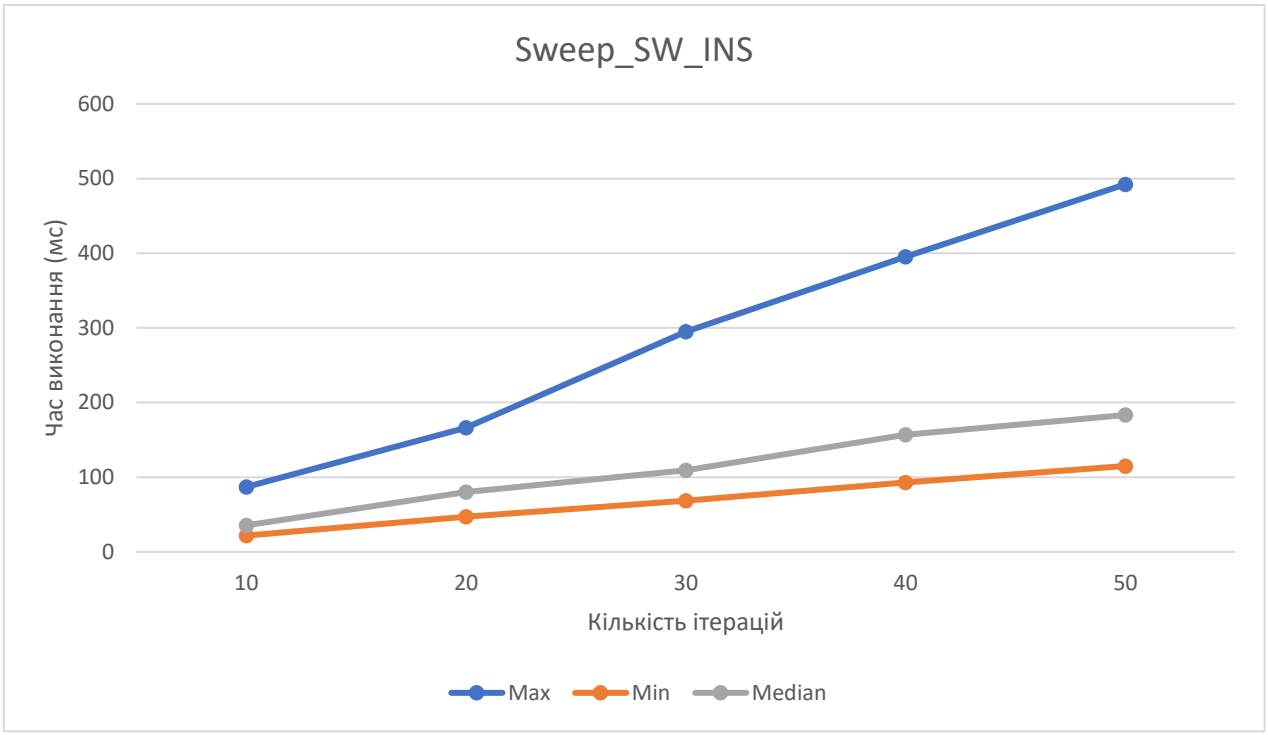


Рисунок 4.17 - Максимальний, мінімальний та середній час виконання алгоритму Sweep_SW_INS у рамках Експерименту 2

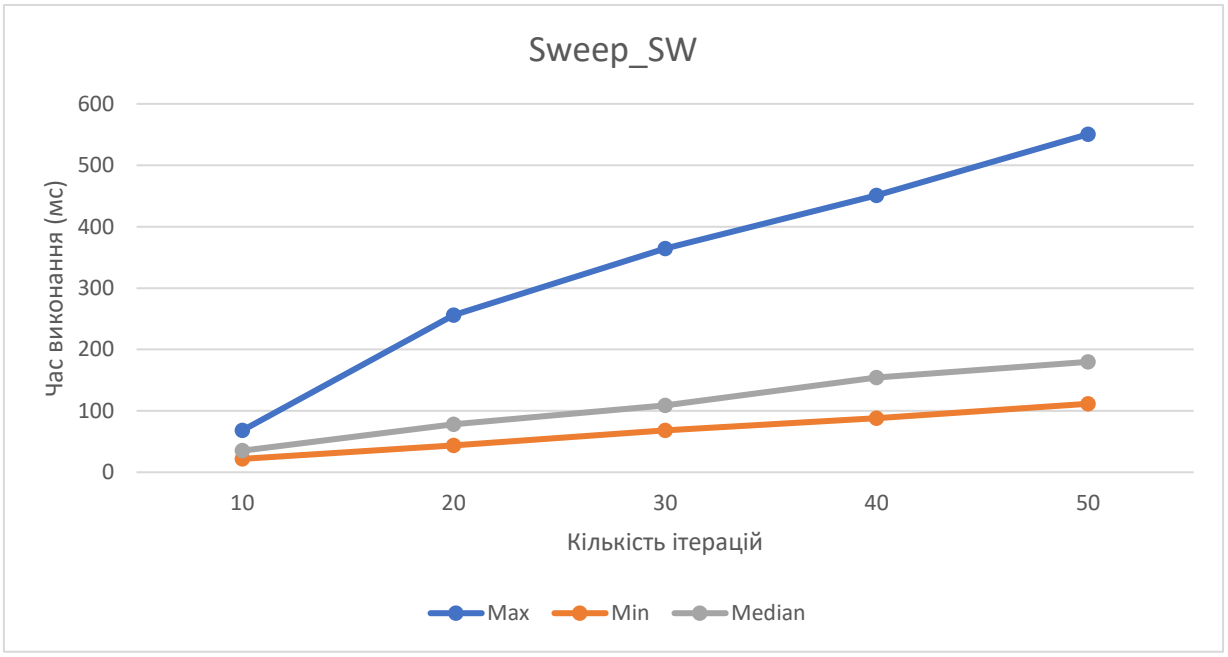


Рисунок 4.18 - Максимальний, мінімальний та середній час виконання алгоритму Sweep_SW у рамках Експерименту 2

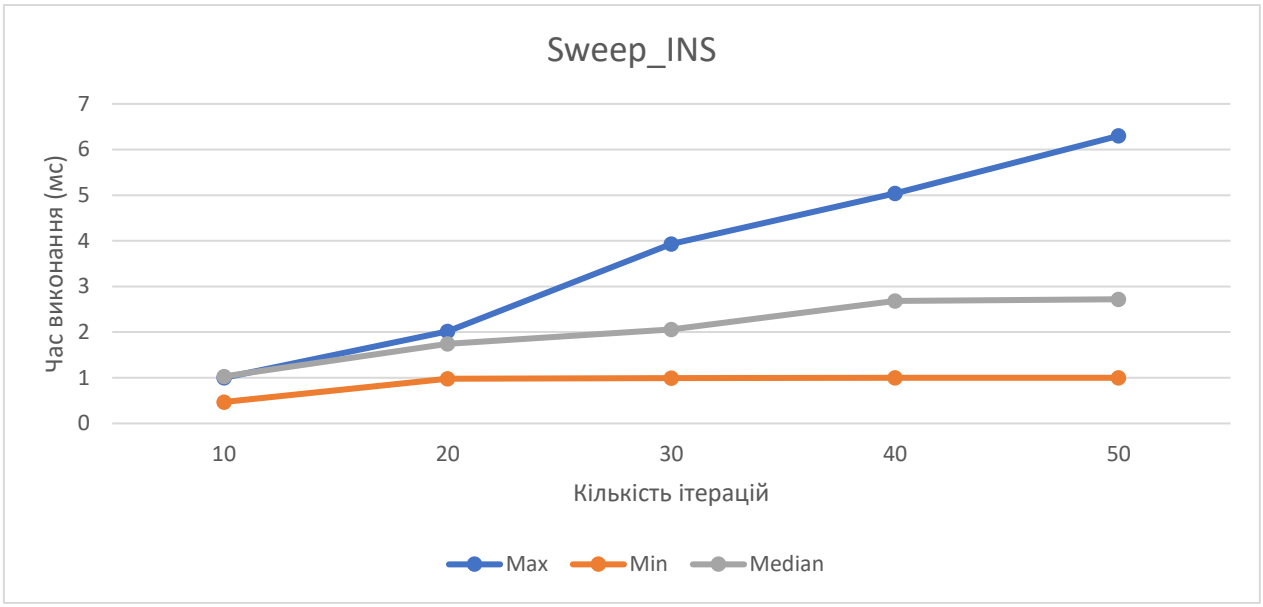


Рисунок 4.19 - Максимальний, мінімальний та середній час виконання алгоритму Sweep_INS у рамках Експерименту 2

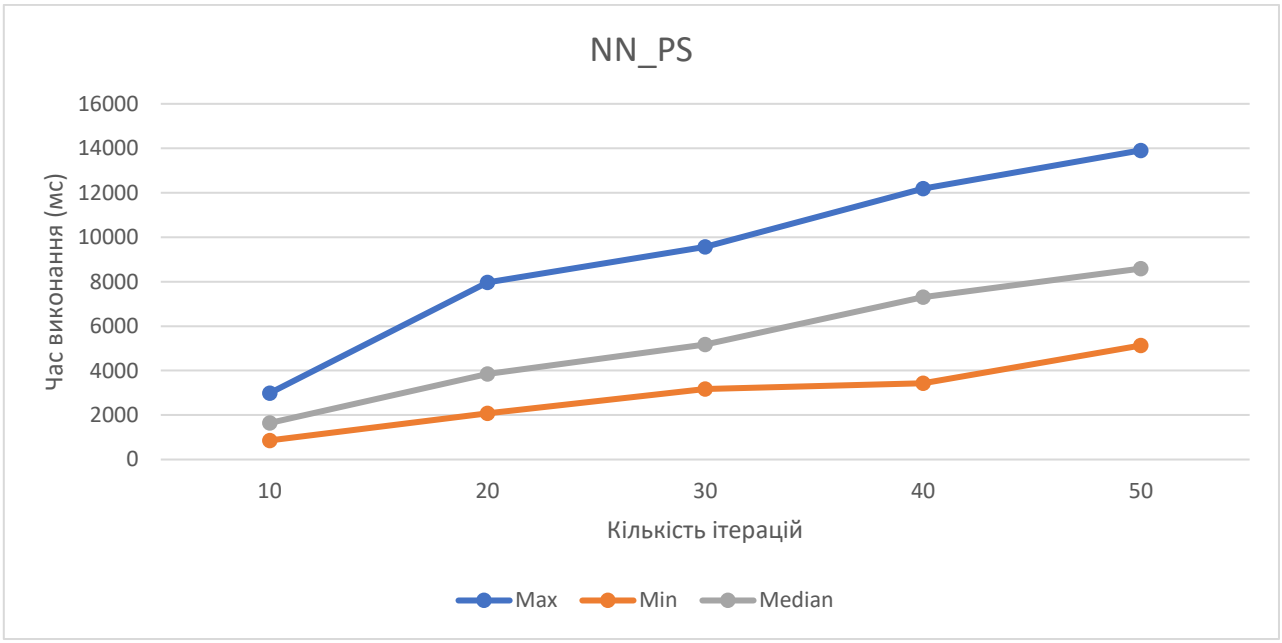


Рисунок 4.20 - Максимальний, мінімальний та середній час виконання алгоритму NN_PS у рамках Експерименту 2

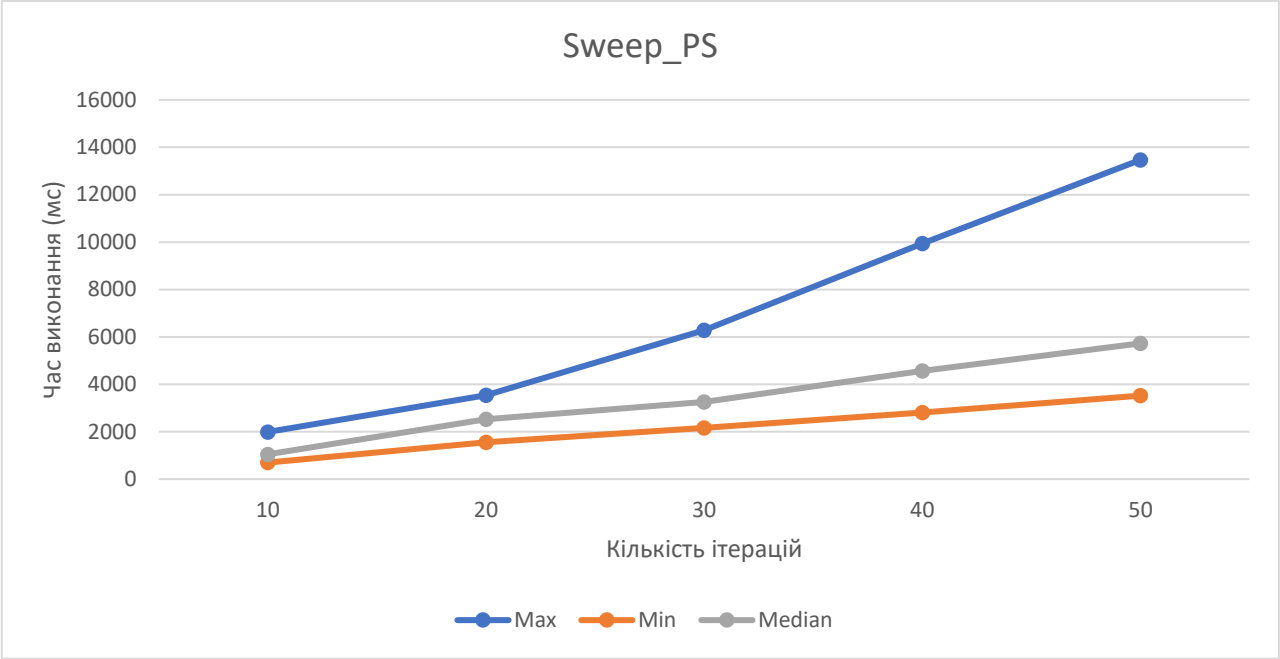


Рисунок 4.21 - Максимальний, мінімальний та середній час виконання алгоритму Sweep_PS у рамках Експерименту 2

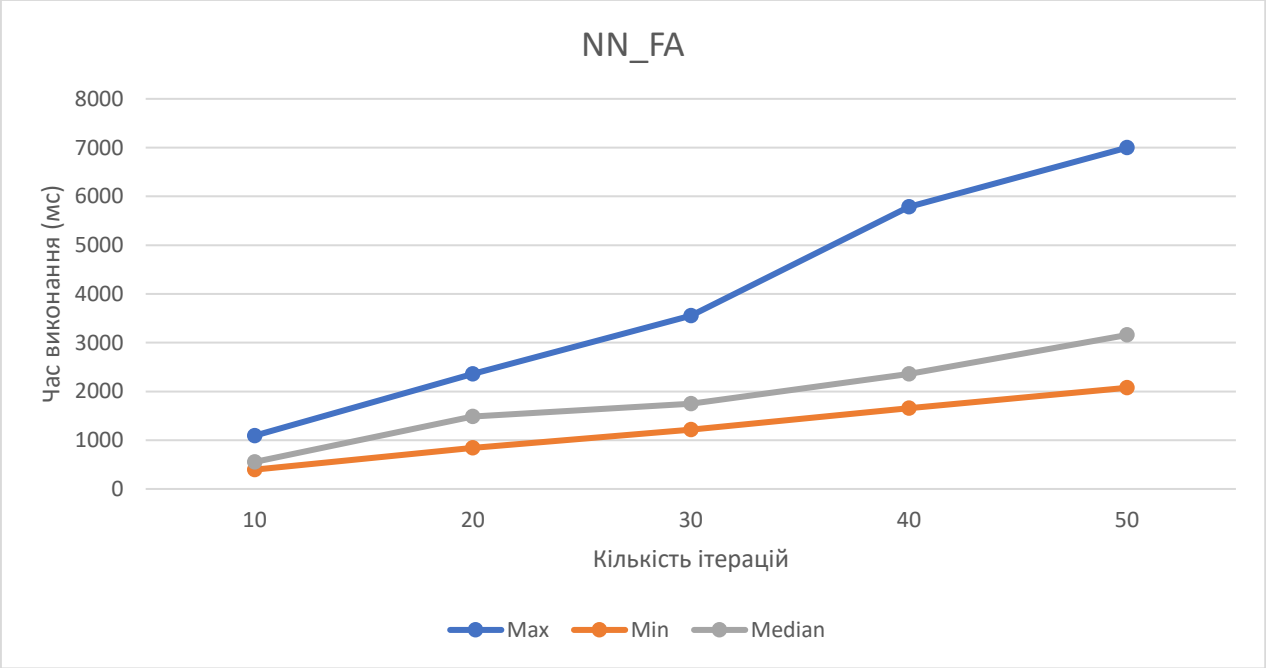


Рисунок 4.22 - Максимальний, мінімальний та середній час виконання алгоритму NN_FA у рамках Експерименту 2

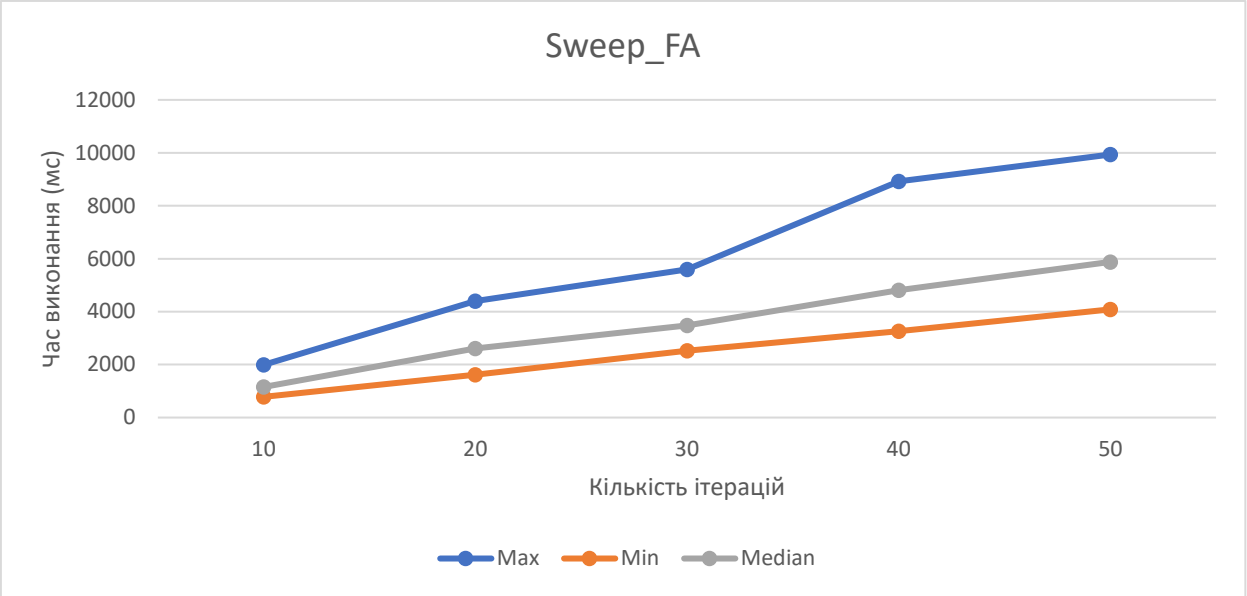


Рисунок 4.23 - Максимальний, мінімальний та середній час виконання алгоритму Sweep_FA у рамках Експерименту 2

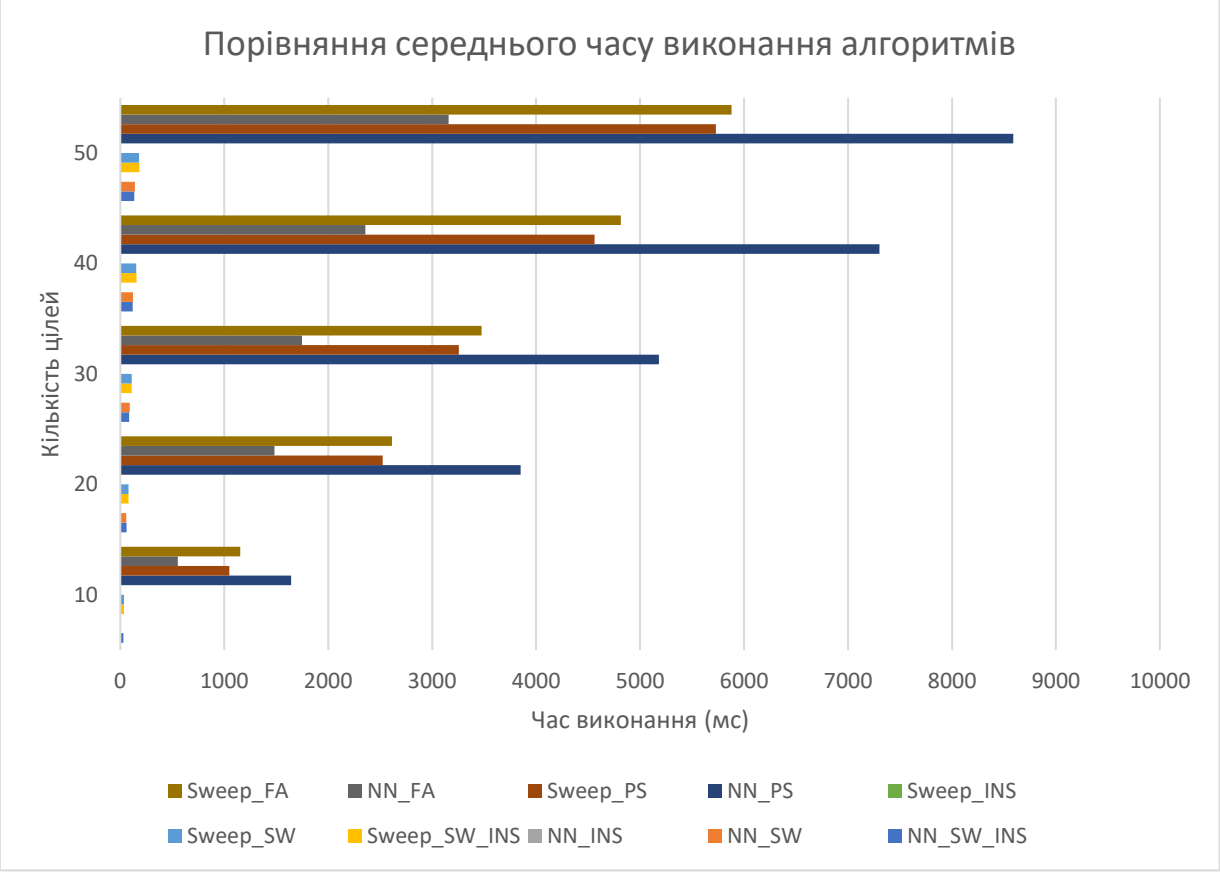


Рисунок 4.24 – Порівняння середнього часу виконання алгоритмів у рамках Експерименту 2

4.2.2 Дослідження ефективності алгоритмів у рамках Експерименту 2

На рисунках 4.25 та 4.26 представлене порівняння ефективності алгоритмів при зміні кількості ітерацій для розв'язку задачі. Найкращий результат у рамках даного експеримента має алгоритм NN_FA, так як він найчастіше надавав найкращий розв'язок задачі у контексті значення цільової функції. Також середню процентну частку кількості найкращих результатів мають такі алгоритми як NN_SW_INS, NN_PS та Sweep_FA. Найгірші значення цільової функції показували алгоритми, що використовують метод Sweep для пошуку початкового розв'язку задачі. Але слід зауважити, що з рисунку 4.26 на прикладі алгоритму Sweep_SW_INS видно, що використання пари операторів Swap та Insertion разом значно вигідніше ніж окремо, бо це зменшує вірогідність отримати найгіршу цільову функцію приблизно у 6 разів.

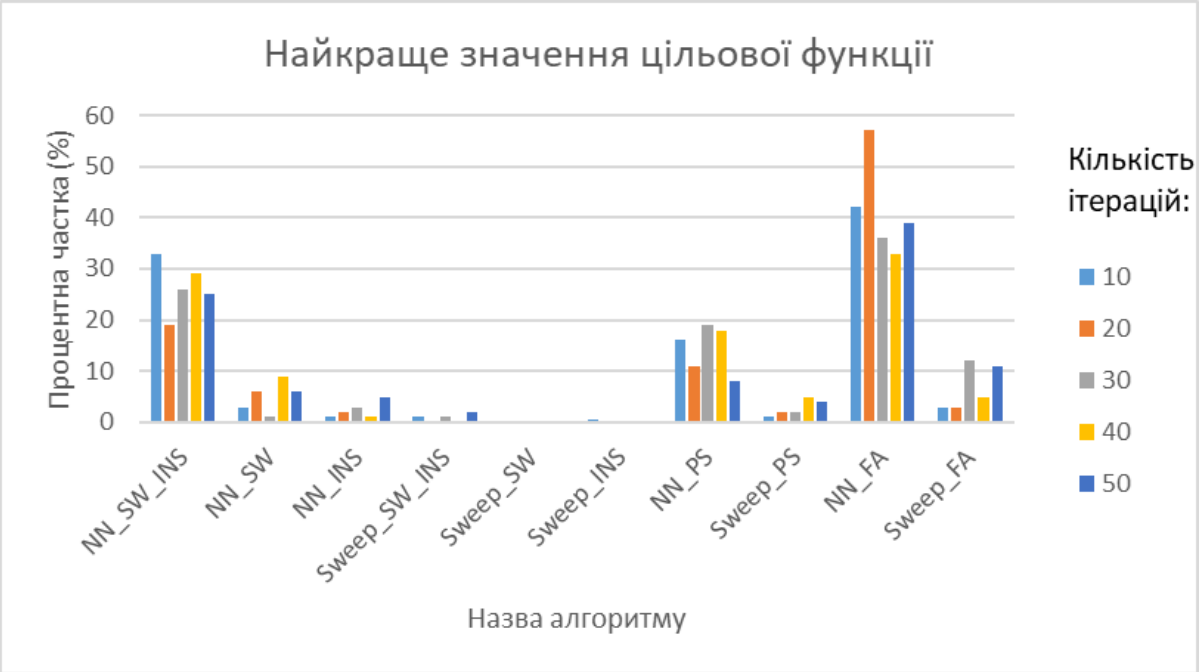


Рисунок 4.25 – Порівняння ефективності алгоритмів у контексті найкращого значення цільової функції у рамках Експерименту 2

Рисунок 4.26 - Порівняння ефективності алгоритмів у контексті найгіршого значення цільової функції у рамках Експерименту 1

Висновки до розділу

У рамках даного розділу були проведені експериментальні дослідження алгоритмів за допомогою Експерименту 1 та Експерименту 2, де досліджувалося як розмірність задачі та кількість ітерацій впливають на швидкодію та ефективність алгоритмів. Тож можна зробити висновок, що при будь-якій розмірності задачі та невеликій кількості ітерацій найкращим варіантом буде використання алгоритму NN_SW_INS. При використанні більшої кількості ітерацій ніж 10 найкращий результат у більшості випадків надає алгоритм NN_FA, але при цьому він працює у 50 разів повільніше ніж NN_SW_INS за рахунок додаткових витрат часу на генерацію валідних популяцій. З даних результатів також випливає, що вибір метод пошуку початкового розв’язку задачі напряму впливає як на швидкодію так і на ефективність алгоритмів. При наявних налаштуваннях для експериментів видно, що метод Sweep значно поступається методу найближчого сусіда, тобто ініціалізує менш вигідний маршрут.

5 СТАРТАП ПРОЕКТ

5.1 Опис ідеї проєкту

Ідеєю даного проєкту є надання програмного забезпечення для підтримки дослідження задачі складання маршруту польоту БПЛА з базою на транспортному засобі.

Таблиця 5.1 – Опис ідеї проєкту

Зміст ідеї	Напрямки застосування
Надання програмного забезпечення для підтримки дослідження задачі складання маршруту польоту БПЛА з базою на транспортному засобі	Вищі технічні освітні заклади
	Військові структури логістики та розвідки
	Наукові установи
	Приватні логістичні компанії

5.2 Технологічний аудит ідеї проекту

У межах даного підрозділу був проведений аудит технології створення програмного продукту та визначена технологічна здійсненність проєкту у таблиці 5.2.

Таблиця 5.2 – Технологічна здійсненність ідеї проєкту

№ п/п	Ідея проєкту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Візуалізація маршрутів	Бібліотека LineRenderer	Наявна	Доступна
2	Збереження розв’язку задач та	SQLite	Необхідно розробити сервіс для	Доступна

	результатів експериментів		роботи зі сховищем	
3	Алгоритми побудови та оптимізації маршрутів	Бібліотека AlgoSolutionProvider	Необхідно розробити	Доступна
4	Візуалізація результатів експериментів у вигляді таблиць	Бібліотека SimpleTableUI	Наявна	Доступна
5	Візуалізація експериментів у вигляді графіків	Бібліотека Dynamic Line Chart	Наявна	Доступна
6	Імпорт вхідних даних та експорт вихідних даних за допомогою csv файлів	Бібліотека CSVFileManager	Необхідно розробити	Доступна
7	Випадкова генерація вхідних даних	Бібліотека RandomDataGenerator	Необхідно розробити	Доступна

Як видно усі ідеї є доступними, тож є можливість для повної реалізації проекту за умови успішної розробки необхідних модулів.

5.3 Аналіз ринкових можливостей запуску стартап-проекту

Для того, щоб спланувати напрями у якому повинен розвиватися продукт необхідно визначити ринкові можливості. Наявність попиту, його обсяг та динаміка розвитку наведені у таблиці 5.3.

Таблиця 5.3 – Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	5
2	Загальний обсяг продаж, грн/ум.од	Невідомий
3	Динаміка ринку	Зростає
4	Наявність обмежень для входу	Немає
5	Специфічні вимоги до стандартизації та сертифікації	Юридичні
6	Середня норма рентабельності в галузі (або по ринку), %	50 %

Беручи до уваги отриману середню норму рентабельності та банківський відсоток на вкладення можна зробити висновок, що ринок є привабливим для входження.

У таблиці 5.4 визначено потенційні групи клієнтів, їх характеристики та орієнтовний перелік вимог до товару для кожної групи.

Таблиця 5.4 - 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	Складання оптимальних	Приватні компанії	Оскільки продукт будуть	Зручний інтерфейс,

	маршрутів для БПЛА		зазвичай використовувати цілі установи, то необхідна наявність підписки на декілька осіб	гнучке налаштування параметрів задачі, зручний ввід даних, можливість збереження отриманих результатів

Після того як потенційні групи клієнтів були визначені перейдемо до аналізу ринкового середовища. У таблиці 5.5 наведені фактори загроз, що заважають впровадженню проекту. А у таблиці 5.6 Наведені фактори можливостей, що навпаки сприяють цьому.

Таблиця 5.5 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Політична ситуація у країнах розробки та впровадження	Війни або воєнний стан, революції, зміни у державному устрої країни	Впровадження нового функціоналу, регулювання цінової політики
2	Епідеміологічна ситуація у країнах розробки та впровадження	Введення надзвичайного стану, введення обмежень на пересування	

3	Низький технічний рівень користувачів	Для того, щоб користуватися технологіями щодо оптимізації маршрутів зазвичай потрібно базове розуміння змісту проблеми	
4	Зростання конкуренції	Поява нових конкурентів на ринку з подібним функціоналом	

Таблиця 5.6 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Розвиток технологій	Поява нових технологій або розвиток вже існуючих дозволить покращити програмний продукт – зробити його більш швидким та зручним для клієнта	Розширення команди для подальшого розвитку продукту
2	Розвиток наукового напрямку у сфері дослідження задач маршрутизації та методів її розв’язання	Розвиток наукового напрямку буде сприяти створенню нових наукових робіт про задачі маршрутизації. В свою чергу поява нових типів задач та методів їх розв’язку дозволить програмному	

		продукту розширюватися, впроваджуючи їх	
3	Розвиток бізнесу та воєнної сфери	За рахунок розвитку бізнесу та воєнних технологій є можливість зростання попиту на продукти для дослідження задач маршрутизації	

Результати ступеневого аналізу конкуренції на ринку наведені в таблиці 5.7.

Таблиця 5.7 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Особливості конкурентного середовища
Тип конкуренції - чиста	Крім програмного продукту, що розглядається на ринку вже існує різне програмне забезпечення, що вирішує схожі проблеми	Впровадити такий функціонал, що наразі є відсутнім у конкурентів
За рівнем конкурентної боротьби - глобальний	Глобальність полягає у розповсюдженні продукту не лише на території України	Охопити всесвітній ринок
За галузевою ознакою - міжгалузева	ІТ галузь	Розвивати ІТ технології
Конкуренція за видами товарів: товарно-видова	Наразі вже існують схожі програмні продукти для оптимізації, але зазвичай вони створенні для	Дослухатися до бажань користувачів

	розв’язання конкретних специфічних типів задач маршрутизації	
За характером конкурентних переваг - нецінова	Клієнти звертають уваги не на ціну, а на якість та зручність у користуванні програмного продукту	Підтримувати якість програмного продукту на високому рівні
За інтенсивністю - не марочна	Програмні продукти такого плану не є широко впізнаваними	Популяризувати програмний продукт

Далі у таблиці 5.8 наведемо аналіз конкуренції у галузі за методом Портера.

Таблиця 5.8 - Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Routific Route Optimization API, Google Or-Tools	AIMMS	Відсутні	Наукові дослідники, приватні компанії	Часткова заміна присутня
Висновки			Конкуренти не мають постачальників		Конкуренти мають схожий функціонал, але є доволі вузькими у виборі

					методів розв’язання задачі та дослідження

Провівши аналіз конкуренції, можна зробити висновок, що можливості для впровадження продукту на ринок високі. Хоча дана сфера вже має конкурентів, проте вони не покривають увесь спектр можливостей, що можна було б реалізувати у контексті задач транспортної маршрутизації та методів їх розв’язання. Далі наведемо фактори конкурентоспроможності у таблиці 5.9.

Таблиця 5.9 - Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Цінова політика	Для потенційних користувачів програмного продукту цінова політика є важливим фактором
2	Дизайн інтерфейсу	Продуманий та зручний дизайн інтерфейсу допомагає заощадити час, що витрачає користувач при роботі з програмним продуктом
3	Швидкодія роботи програмного продукту	Швидкодія ще один фактор, що допомагає заощадити час
4	Впізнаваність компанії	Чим вища впізнаванність компанії, тим більше потенційних клієнтів
5	Наявність додаткового функціоналу	Даний фактор допомагає відрізнитися у позитивну сторону від конкурентів

6	Якість комунікації між користувачем та розробником	Висока якість комунікації допомагає швидко виявляти проблеми, що існують у програмному продукті та виправляти їх

За факторами, що були визначені у попередній таблиці наведемо порівняльний аналіз існуючих слабких та сильних сторін продукту у таблиці 5.10.

Таблиця 5.10 - Порівняльний аналіз сильних та слабких сторін продукту

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів						
			-3	-2	-1	0	+1	+2	+3
1	Цінова політика	15	+						
2	Дизайн інтерфейсу	16			+				
3	Швидкодія роботи програмного продукту	19					+		
4	Впізнаваність компанії	18						+	
5	Наявність додаткового функціоналу	17				+			
6	Якість комунікації між користувачем та розробником	13		+					

Як фінальний етап ринкового аналізу складемо SWOT-аналіз, представлений у таблиці 5.11.

Таблиця 5.11 – SWOT – аналіз стартап-проекту

Сильні сторони: лояльна цінова політика, зручний інтерфейс	Слабкі сторони: відсутність кросплатформенності, мала кількість вбудованих алгоритмів

Можливості: поява інвесторів для розвитку стартапу	Загрози: стрімкий розвиток конкурентів

Провівши SWOT-аналіз необхідно визначити які є альтернативи ринкової поведінки, для того щоб вивести стартап на ринок. Дані альтернативи, представлені на рисунку 5.12, аналізують у контексті визначення строків на ринкову реалізацію, а також ймовірності отримання ресурсів.

Таблиця 5.12 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива	Ймовірність отримання ресурсів	Строки реалізації
1	Поширення програмного продукту на інші платформи	Високий	6 місяців

5.4 Розроблення ринкової стратегії проекту

Зазвичай при розробці ринкової стратегії першим кроком визначають стратегії для охоплення ринку, а саме визначення потенційних споживачів та поділ їх на групи, що наведені у таблиці 4.13.

Таблиця 5.13 - 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	Логістичні компанії	90%	100%	Висока	Низька
2	Військові логістичні та розвідувальні підрозділи	70%	100%	Низька	Висока
3	Наукові структури	80%	100%	Середня	Середня
Які цільові групи обрано: усі (1, 2, 3)					

За результатами визначення цільових груп потенційних споживачів було прийняте рішення працювати з усіма сегментами одночасно, тобто використовувати масовий маркетинг. Далі визначимо базову стратегію розвитку у таблиці 5.14.

Таблиця 5.14 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Зосередження на науковому сегменті	Масовий маркетинг	Висока швидкодія, зручний інтерфейс, різноманіття методів	Стратегія диференціації

Далі у таблиці 5.15 визначимо базову стратегію конкурентної поведінки.

Таблиця 5.15 - Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати	Чи буде компанія копіювати основні характеристики	Стратегія конкурентної поведінки

		існуючих у конкурентів?	товару конкурента, і які?	
1	Ні	Компанія буде шукати нових споживачів та забирати існуючих у конкурентів	Ні	Стратегія лідера

У таблиці 5.16 визначенні, які системи позиціювання будуть використовуватися для просування програмного продукту. Для цього були сформульовані вимоги до товару з точки зору цільової аудиторії, базова стратегія розвитку, ключові конкурентоспроможні позиції та асоціації.

Таблиця 5.16 - Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які повинні сформувати комплексну позицію власного проекту
1	Зручний інтерфейс	Стратегія диференціації	Слідування правилам UI/UX дизайну, швидкодія, широкий спектр функціоналу	Лояльна цінова політика
2	Різноманіття алгоритмів			

3	Зручний функціонал аналізу алгоритмів			
4	Швидкодія програмного продукту			

5.5 Розробка маркетингової програми стартап-проекту

У рамках даного підрозділу необхідно сформувати концепцію продукту, який буде використовувати користувач. Для цього у таблиці 5.17 наведені ключові переваги концепції потенційного продукту.

Таблиця 5.17 - Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами
1	Підтримка дослідження задачі складання маршруту польоту БПЛА з базою на транспортному засобі	Вивчення задач маршрутизації та методів їх розв’язання	Наявність інтерфейсу, візуалізація результатів, наявність декількох алгоритмів розв’язання
2	Складання маршруту польоту БПЛА з базою на	Зменшення часу на обслуговування клієнтів	

	транспортному засобі		

У таблиці 5.18 наведені три рівні, їх сутність та складові.

Таблиця 5.18 – Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>I. Товар за задумом</td><td colspan="3">Програмний продукт надає функціонал з підтримки дослідження задачі складання маршруту польоту БПЛА з базою на транспортному засобі</td></tr>
<tr><td rowspan="5">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх /Тл/Е/Ор</td></tr>
<tr><td>1. зручний інтерфейс;
2. низька ціна;
3. складання маршрутів БПЛА;
4. дослідження задачі транспортної оптимізації;
5. швидкодія.</td><td>-</td><td>-</td></tr>
<tr><td colspan="3">Якість: перевірка якості здійснюється за рахунок різних видів тестування, а саме: мануальне, юніт-тести, інтеграційні тести</td></tr>
<tr><td colspan="3">Пакування: електрона документація (інструкція користувача)</td></tr>
<tr><td colspan="3">Марка: UAV Route</td></tr>
<tr><td rowspan="2">III. Товар із підкріпленням</td><td colspan="3">До продажу: пробна версія з обмеженням на розмірність задачі</td></tr>
<tr><td colspan="3">Після продажу: підтримка користувачів та розвиток програмного продукту за рахунок додавання нового функціоналу</td></tr>
<tr><td colspan="4">Товар буде захищений від копіювання за рахунок ліцензійних ключів.</td></tr>
</table>

Формування систем збуту описане у таблиці 5.19.

Таблиця 5.19 – Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	немає	Продаж ліцензійних ключів програмного продукту	Однорівневий	Власна та залучена система збуту

Як останній крок у таблиці 5.20 представлена концепція маркетингових комунікацій.

Таблиця 5.20 – 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	Орієнтація на зручність та швидкодію програмного продукту	Месенджери, соціальні мережі, електроні скриньки	Створення довіри до компанії та продукту за рахунок комунікації	Донести до клієнтів які проблеми може вирішувати даний продукт	Зробити акцент на переваги програмного продукту серед конкурентів

Висновки до розділу

У межах даного розділу був проведений маркетинговий аналіз проєкту, а саме були проаналізовані ринкові можливості та напрямки для реалізації програмного продукту, розроблена ідея проєкту, напрями використання та відмінності від конкурентів. Також був зроблений технологічний аудит та проаналізовані можливості щодо впровадження стартап-проєкту. Проаналізувавши наявний попит, рентабельність, динаміку ринку, потенційних клієнтів та конкурентів можна зробити висновок, що можливість ринкової комерціалізації проекту наявна та перспективи для впровадження є доволі високими оскільки конкуренція у сфері, що розглядається не є наразі великою, а попит на такі продукти буде зростати з кожним роком. З цього випливає, що подальша імплементація проекту є цілком доцільною. У розробці маркетингових комунікацій було вирішено робити акцент на широкі можливості програмного продукту та акцентувати увагу на спектр проблем, які він здатен вирішити.

ВИСНОВКИ

У рамках даної дисертації були сформульовані дві математичні моделі задач транспортної маршрутизація, у тому числі запропонована нова постановка задачі, що включає в себе дві бази, транспортний засіб, БПЛА та деякі обмеження на пересування та місткість БПЛА. Було розроблено десять алгоритмів для розв’язання даної задачі, де кожен алгоритм комбінує у собі методи, що надають можливість пошуку початкового розв’язку задачі або його локального покращення. Була розроблена багатокомпонентна система з імплементацією цих алгоритмі, сховищем даних та інтерфейсом. Були проведені експериментальні дослідження розроблених алгоритмів у контексті порівняння їх ефективності та швидкодії при різних розмірностях задачі або кількості ітерацій покращення розв’язку. У результаті цих досліджень був зроблений висновок, що при даному налаштуванні проведення експериментів найкращий результат надає алгоритм, що на першому кроці виконує пошук початкового розв’язку задачі за допомогою методу найближчого сусіда, а на другому виконує локальне покращення за допомогою Swap та Insertion операторів. Також слід зазначити, що локальне покращення тільки оператором Swap або тільки оператором Insertion дає значно гірші результати, тобто такі методи необхідно використовувати саме у парі. Щодо алгоритмів Firefly та Producer-Scrounger, де використовуються популяції для покращення розв’язку, то хоча при деяких параметрах дані алгоритми дають непоганий у порівнянні з іншими алгоритмами результат їх середній час виконання є найбільшим серед усіх за рахунок витрат часу на генерацію певної кількості валідних популяцій. Тож дані алгоритми потребують подальшого удосконалення розрізі швидкодії.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Shi-Yi Tan and Wei-Chang Yeh, “The vehicle routing problem: state-of the-art classification and review.” - Appl. Sci. 2021, 11, 10295.

2. Rajeev Goel and Raman Maini, “Vehicle routing problem and its solution methodologies: a survey” - Int. J. Logistics Systems and Management, Vol. 28, No. 4, 2017.

3. Routific [Електронний ресурс] – Режим доступу до ресурсу: https://routific.com/.

4. Google Developers [Електронний ресурс] – Режим доступу до ресурсу: https://developers.google.com/.

5. Aimms: Capacitated Vehicle Routing Problem formulation [Електронний ресурс] – Режим доступу до ресурсу: https://how-to.aimms.com/Articles/332/332-Formulation-CVRP.html/.

6. Zahrul Jannat Peya, “Solving Capacitated Vehicle Routing Problem through Clustering with Variant Sweep Algorithm and Route Optimization using Swarm Intelligence.” - Khulna University of Engineering & Technology, May 2016.

7. Yin-Mou Shen, Ruey-Maw Chen and M. M. Hafizur Rahman “Optimal multi-depot location decision using particle swarm optimization.” - Advances in Mechanical EngineeringVolume 9, Issue 8, August 2017.

8. M. A. H. Akhand, “Producer-Scrounger Method to Solve Traveling Salesman Problem” - International Journal of Intelligent Systems and Applications 7(7):29-36, February 2015.

9. Jati, G.K., “Evolutionary discrete firefly algorithm for travelling salesman problem.” – Volume 6943. Springer (2011).

10. Zhou, L., Ding, L., Qiang, X., “A multi-population discrete firefly algorithm to solve tsp.” - Bio-Inspired Computing-Theories and Applications. Springer (2014) 648–653.

11. Eneko Osaba, Fernando Diaz, Xin-She Yang and Enrique Onieva, “A Discrete Firefly Algorithm to Solve a Rich Vehicle Routing Problem Modelling a Newspaper Distribution System with Recycling Policy” - Soft Computing · September 2017.

12. A Guide to Functional Requirements (with Examples) [Електронний ресурс] – Режим доступу до ресурсу: https://www.nuclino.com/articles/functional-requirements#how-to-write-a-functional-requirements-document.

13. What makes unity so popular in game development [Електронний ресурс] – Режим доступу до ресурсу: https://www.arnia.com/what-makes-unity-so-popular-in-game-development/.

14. What Is SQLite? [Електронний ресурс] – Режим доступу до ресурсу: https://www.sqlite.org/index.html.

15. PGC Coordinate Converter [Електронний ресурс] – Режим доступу до ресурсу: https://www.pgc.umn.edu/apps/convert/.