Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та мехатроніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

Розробка системи підтримки прийняття рішень для завдань переміщення мобільної платформи
(тема)

Виконав:
студент 2 курсу, групи КТРСм-21-1
Мандрікін М.С.
(прізвище, ініціали)

Спеціальності 151 Автоматизація та комп`ютерно-інтегровані технології
(код і повна назва спеціальності)
Тип програми Освітньо-професійна
Освітня програма Комп'ютеризовані та робототехнічні системи
(повна назва освітньої програми)

Керівник проф. Цимбал О.М.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри КІТАМ _____ Невлюдов І. Ш.
(підпис) (прізвище, ініціали)

2022 р.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет	Автоматики і комп'ютеризованих технологій
Кафедра	Комп'ютерно-інтегрованих технологій, автоматизації та мехатроніки
Рівень вищої освіти	другий (магістерський)
Спеціальність	151 Автоматизація та комп'ютерно-інтегровані технології
Тип програми	освітньо-професійна
Освітня програма	Комп'ютеризовані та робототехнічні системи

(код і повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри Невлюдов І.Ш
(підпис)
«_____»_____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Мандрікіну Микиті Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка системи підтримки прийняття рішень для завдань переміщення мобільної платформи
затверджена наказом по університету від 07.11. 2022 р. № 1462 Ст.
2. Термін подання студентом роботи до екзаменаційної комісії 21.12. 2022 р.
3. Вихідні дані до роботи задача переміщення мобільної платформи, розробка СППР.

4. Перелік питань, що потрібно опрацювати в роботі: аналіз предметної області та технічного завдання, аналіз існуючих підходів до навігації мобільних роботів, дослідження алгоритмів пошуку за першим найкращим збігом, проведення імітаційного моделювання, порівняння показників та характеристик алгоритмів, створення СППР

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів)
демонстраційний матеріал представлений у форматі презентації PowerPoint (*.pptx) – 27 с.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на виконання кваліфікаційної роботи	15.09.2022	Виконано
2	Визначення актуальності роботи, мети, предмета, об'єкта, та розробка завдань для досягнення мети	22.09 – 30.09.22	Виконано
3	Аналіз літературних джерел	01.10 – 13.10.22	Виконано
4	Вибір середовища розробки	15.10 – 3.11.22	Виконано
5	Оформлення пояснювальної записки та документації	05.11 – 19.11.22	Виконано
6	Оформлення додатків	20.11 – 01.12.2	Виконано
7	Оформлення графічного та презентаційних матеріалів комп'ютерного захисту	01.12 – 06.12.22	Виконано
8	Представлення роботи на рецензування	14.12 – 19.12.22	Виконано

Дата видачі завдання 15.09.2022

Студент _____ Мандрікін М.С.
(підпис) (прізвище, ініціали)

Керівник роботи _____ проф. Цимбал О.М.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 155 с., 26 табл., 41 рис., 4 дод., 32 джерел.

АВТОМАТИЗАЦІЯ, МОБІЛЬНА ПЛАТФОРМА, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, СППР, ПОШУК ШЛЯХУ.

Мета роботи – розробка компонентів програмного забезпечення системи підтримки прийняття рішень для завдань переміщення мобільної платформи.

Об’єкт дослідження – процес керування мобільним роботом.

Предмет дослідження – програмна реалізація системи підтримки прийняття рішень для завдань переміщення мобільної платформи.

Методи дослідження – імітаційне моделювання, теорія графів, логічний та системний аналіз, методи і засоби збору та обробки даних, теорія алгоритмів.

Виконано огляд сучасних існуючих підходів до навігації мобільних роботів, знайдено найбільш оптимальний з них за допомогою аналізу літератури, досліджено алгоритми у рамках обраного підходу, проведено імітаційне моделювання та на основі отриманих даних створено систему підтримки прийняття рішень для завдань переміщення мобільної платформи.

ABSTRACT

Explanatory note: 155 p., 26 tabl., 41 fig., 4 app., 32 sources.

AUTOMATION, MOBILE PLATFORM, SOFTWARE, DECISION SUPPORT SYSTEM, PATH FINDING.

The purpose of the work is the development of software components of the decision support system for the tasks of mobile platform movement.

The object of research is the mobile robot control process.

The subject of research is the software implementation of the decision support system for the tasks of mobile platform movement.

Research methods – simulation modeling, graph theory, logical and system analysis, methods and techniques of data collection and processing, theory of algorithms.

The review of modern existing approaches to the navigation of mobile robots was performed, the most optimal of them was found by analyzing a literature, the algorithms within the chosen approach were reviewed, simulation was made and based on the obtained data a decision support system for the tasks of mobile platform movement was created.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ОПР – особа, що приймає рішення;

ПЗ – програмне забезпечення;

СППР – система підтримки рішень.

GPS – Global Positioning System;

ВСТУП

У зв’язку зі швидким розвитком робототехніки завдання керування роботами стають все більш актуальними та комплексними, саме тому об’єктом дослідження цієї роботи є процес керування мобільним роботом.

Планування траєкторії руху є однією з найважливіших дослідницьких проблем робототехніки. Багато проблем у різних галузях вирішуються за допомогою планування траєкторії. Воно застосовується при керуванні роботом для досягнення певної мети від дуже простого планування траєкторії до вибору відповідної послідовності дій. Планування траєкторії не завжди може бути спроектоване заздалегідь, оскільки точна інформація про глобальне середовище не завжди доступна. Запропонувавши відповідний алгоритм, планування траєкторії може бути широко застосовано в частково і невідомому структурованому середовищі.

Від правильного вибору алгоритму переміщення мобільної платформи залежить ефективність функціонування робота, та навіть його технічні характеристик, так як від обраного алгоритму залежать такі важливі показники як необхідна кількість пам’яті та швидкість роботи, та разом з тим правильний вибір прямо впливає на витрату ресурсів для створення робота а також на загальну ефективність його функціонування.

Цю ситуацію можна охарактеризувати як вибір в умовах невизначеності та складних умовах для повного й об'єктивного аналізу предметної діяльності. Така проблема може бути вирішення за допомогою СППР.

СППР – інтерактивна комп’ютерна система для підтримки різних видів діяльності під час прийняття рішень стосовно різноманітних проблем. Застосування СППР забезпечує виконання ґрунтовного та об’єктивного аналізу предметної області при прийнятті рішень у складних умовах. Її застосування дозволить більш ефективно керувати мобільною платформою наближаючи її

параметри до необхідних в наслідок чого скоротити витрати ресурсів при розробці та подальшої експлуатації, саме тому тема роботи є актуальною.

Мета роботи – розробка компонентів програмного забезпечення системи підтримки прийняття рішень для завдань переміщення мобільної платформи.

Об’єкт дослідження – процес керування мобільним роботом.

Предмет дослідження – програмна реалізація системи підтримки прийняття рішень для завдань переміщення мобільної платформи.

Методи дослідження – імітаційне моделювання, теорія графів, логічний та системний аналіз, методи і засоби збору та обробки даних, теорія алгоритмів.

Для досягнення поставленої мети необхідно:

– провести аналіз технічного завдання;

– дослідити алгоритми пошуку;

– провести імітаційне моделювання;

– базуючись на отриманих даних, створити СППР;

– програмно реалізувати систему підтримки прийняття рішень для завдань переміщення мобільної платформи;

– розглянути питання охорони праці.

Кваліфікаційну роботу виконано згідно [1], [2].

1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

1.1 Системи підтримки прийняття рішень

Єдиного визначення для СППР немає. Існує кілька поширених визначень СППР, які відображають основні особливості будови, використання та ефективності застосування цих систем [3].

СППР – інтерактивна комп’ютерна система для підтримки різних видів діяльності під час прийняття рішень стосовно різноманітних проблем.

СППР дає змогу користувачам моделювати й аналізувати інформацію у такий спосіб, який буде найефективнішим для вироблення певного специфічного рішення і буде забезпечувати підтримку в інтерактивному режимі.

Необхідно розуміти, що СППР будь-якого типу повинна містити:

– підсистему введення та аналізу запитів користувача (ПВАЗ);

– підсистему оброблення запитів користувача та генерації результатів (ПОЗГР);

– базу знань і даних (БЗД);

– підсистему подання результатів (ППР) в зручній для користувача формі.

Задачі прийняття рішень постійно виникають і розв’язують у природі, у навколишньому світі, що нас оточує, зокрема у біологічних, екологічних, соціальних і економічних системах, різноманітних процесах та явищах.

Рішенням вважають обґрунтований набір дій з боку особи, що приймає рішення (ОПР). Ці дії спрямовані на об’єкт чи систему управління, а набір дій надає можливість привести даний об’єкт чи систему до бажаного стану або досягнути поставленої мети.

Застосування СППР забезпечує виконання ґрунтовного та об’єктивного аналізу предметної області при прийнятті рішень у складних умовах.

1.1.1 Структура СППР

Можна виділити наступні основні структурні компоненти (рисунок 1.1):

– база даних – призначена для зберігання, управління, відбору, відображення та аналізу даних;

– інтерфейс користувача – інтерфейс користувача дає змогу особі, яка приймає рішення, виконувати діалог із системою, використовуючи програми введення, формати і технології виведення;

– система вироблення рекомендацій – розробляється відповідно до структури розв'язання оптимізаційних задач: інструменти формування математичної моделі, що включає критерії вибору та область допустимих рішень – альтернатив, а також інструменти для розв'язання сформованої задачі вибору.

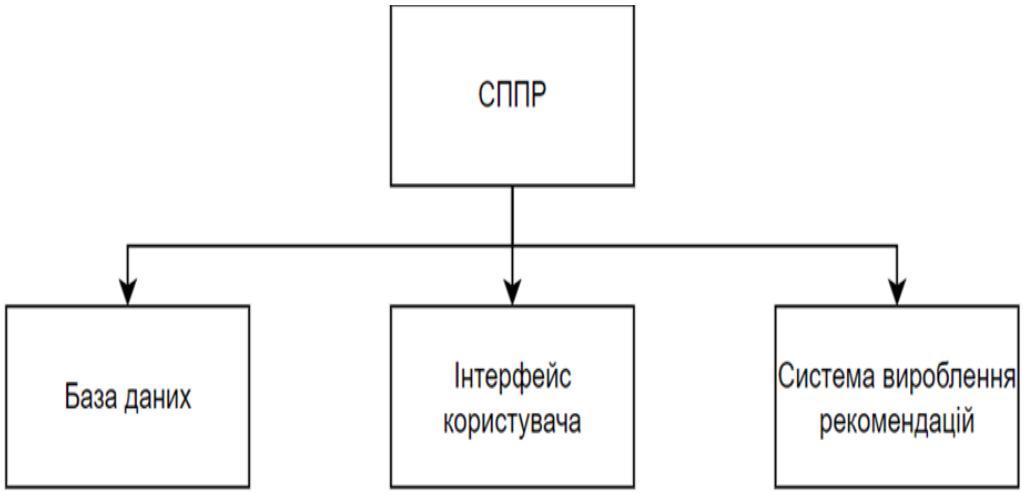


Рисунок 1.1 – Загальна структура СППР

1.1.2 Визначення структури розв'язання оптимізаційних задач

Виходячи з теми роботи можна визначити наступну структури розв'язання оптимізаційних задач: маємо задачу з декількома критеріями.

Існує три основних метода прийняття рішень в умовах векторної (багатокритеріальної) оптимізації:

– лексикографічний метод – ґрунтується на впорядкуванні критеріїв, що враховуються, в порядку убування значущості, з наданням переваги екстремуму вищих із них над таким самим значенням усіх інших. Має наступні недоліки: складність і суб'єктивність ранжування критеріїв, непридатність у разі їх рівнозначності;

– метод послідовних поступок (Парето) – вимагає ранжирування і пошуку оптимуму найважливішого з критеріїв. Потім його величину знижують на деяку поступку, і знаходять екстремальне значення наступного, і так доти, доки не буде виявлено оптимальне значення всієї сукупності критеріїв. Недоліки подібні до попередніх плюс суб'єктивізм у виборі поступок і сумніви в екстремальності знайдених критеріїв, крім першого;

– метод згортання критеріїв – пов'язаний із підсумовуванням критеріїв після надання кожному з них індивідуального вагового коефіцієнта. Недоліком є довільність у виборі ваг, що іноді призводить до помітної не екстремальності окремих критеріїв, незважаючи на оптимальне значення всієї їхньої адитивної згортки.

У рамках цієї роботи були виділені наступні критерії системи керування мобільним роботом:

– оптимальність отриманого шляху;

– споживання пам’яті;

– швидкість роботи.

У якості метода прийняття рішень в умовах векторної (багатокритеріальної) оптимізації було обрано метод згортання критеріїв.

1.2 Мобільні роботи

Мобільний робот – це робот, здатний пересуватися. Така інтелектуальна технічна система, не прив'язана до однієї локації, виконує задані дії згідно з

інтегрованою в неї базою знань. Залежно від програми, закладеної в блок управління, мобільний робот діє автономно або управляється віддалено оператором.

Мобільні роботи можуть переміщуватись у своєму середовищі та не прив'язані до одного фізичного місця розташування. Мобільні роботи можуть бути автономними, що означає, що вони здатні здійснювати навігацію в неконтрольованому середовищі без необхідності використання фізичних або електромеханічних пристроїв керування. Як альтернатива мобільні роботи можуть покладатися на пристрої керування, які дозволяють їм переміщатися за заздалегідь визначеним маршрутом навігації у відносно контрольованому просторі. Навпаки, промислові роботи зазвичай більш-менш нерухомі [4].

1.2.1 Класифікація мобільних роботів

Існує багато різноманітних типів мобільних роботів (рисунок 1.2). Типізація робототехніки побудована на параметрах автономних пристроїв і їхніх здібностях взаємодіяти з навколишнім середовищем [5].

За середовищем, у якому пересуваються:

– наземні;

– повітряні;

– морські.

За принципом пересування:

– колісні (з різною кількістю коліс) і гусеничні, що відрізняються високою прохідністю;

– крокуючі та стрибаючі, що відрізняються числом кінцівок;

– літаючі;

– плаваючі;

– лазаючі;

– зооморфні, або біоміметичні;

– спеціалізовані - на повітряній або електромагнітній подушці, з приводами на вакуумних присосках або липучках, ті, що не входять у попередні принципи.

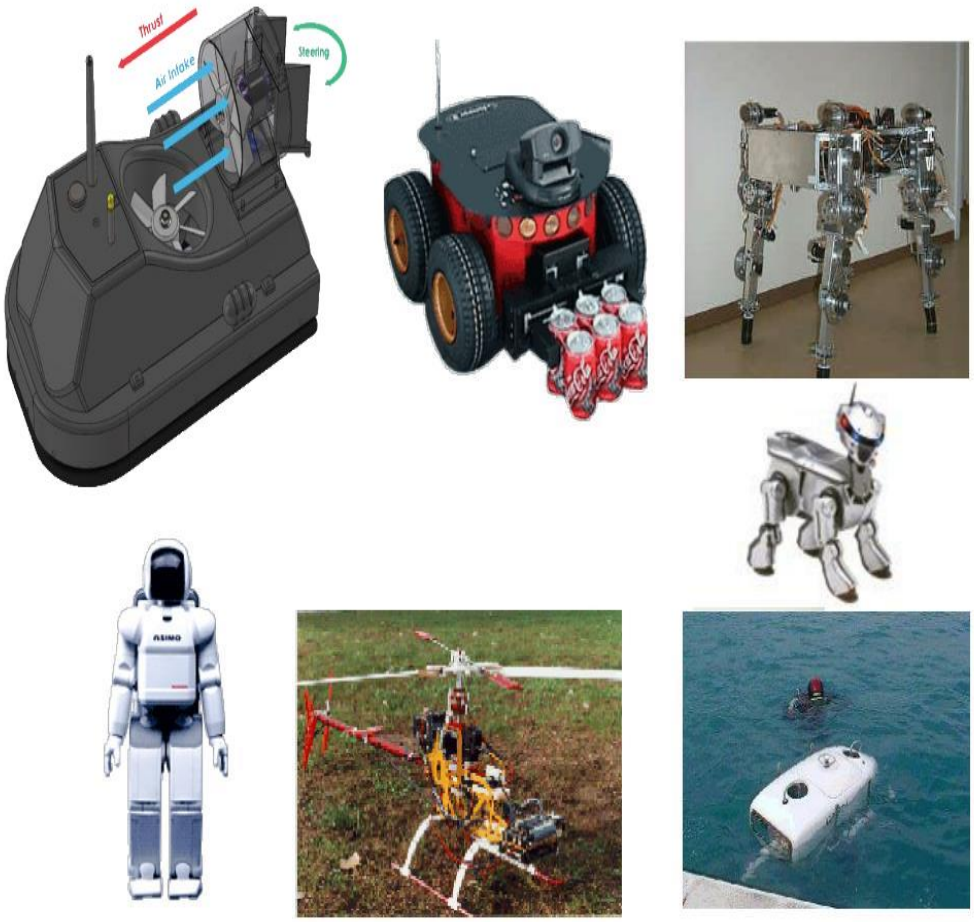


Рисунок 1.2 – Мобільних роботи для виконання різних задач

1.3 Задача керування рухом мобільного робота

Ця задача відноситься до групи транспортних задач. Її вирішення полягає у плануванні переміщення робота з врахуванням інформації яка надходить з датчиків, які в свою чергу забезпечують загальний зворотній зв'язок, надаючи інформацію про різні параметри зовнішнього середовища [6].

Вирішення цієї задачі можна описати як поєднання трьох основних функцій: картографування, локалізація та планування шляху.

Картографування вбудовано в мобільний робот з використанням підходу запам’ятовування, що дозволяє йому повністю рухатися та досліджувати навколишнє середовище.

Процес локалізації використовується для визначення поточного розташування робота в межах репрезентації оточуючого середовища у пам’яті. Прийняття рішення про певний рух для досягнення цілі за допомогою карти та процесу локалізації називається процес планування шляху.

Усі наведені функції, як правило, забезпечують роботу можливість визначити його поточне розташування, що потім дозволяє планувати шлях досягнення певної цілі.

При вирішенні наведеної задачі необхідно забезпечити можливість робота визначати своє положення в незалежності від оточуючої ситуації. Важливо забезпечити робота чутливими сенсорами та датчиками для виконання наведених вимог.

1.4 Автоматичне картографування місцевості

Проблема робототехнічного картографування полягає в отриманні просторової моделі середовища, в якому перебуває робот. Для отримання карти робот повинен мати сенсори, які дозволяють йому сприймати оточуюче середовище.

Датчики, які зазвичай використовуються для цього завдання, включають камери, далекоміри, що використовують сонар, лазерні та інфрачервоні технології, радар, тактильні датчики, компаси, і GPS. Проте всі ці датчики схильні до похибок, які часто називають шумом вимірювань.

Більш важливо, що більшість сенсорів роботів мають обмеження по дальності дії. Наприклад, світло і звук не можуть проникати крізь стіни. Ці

обмеження дальності дії роблять необхідним для робота переміщуватися в навколишньому середовищі при побудові карти.

Команди руху (управління), що видаються під час дослідження навколишнього середовища, несуть важливу інформацію для побудови карт (рисунок 1.3), оскільки вони передають інформацію про місця, в яких були зроблені різні сенсорних вимірювань.

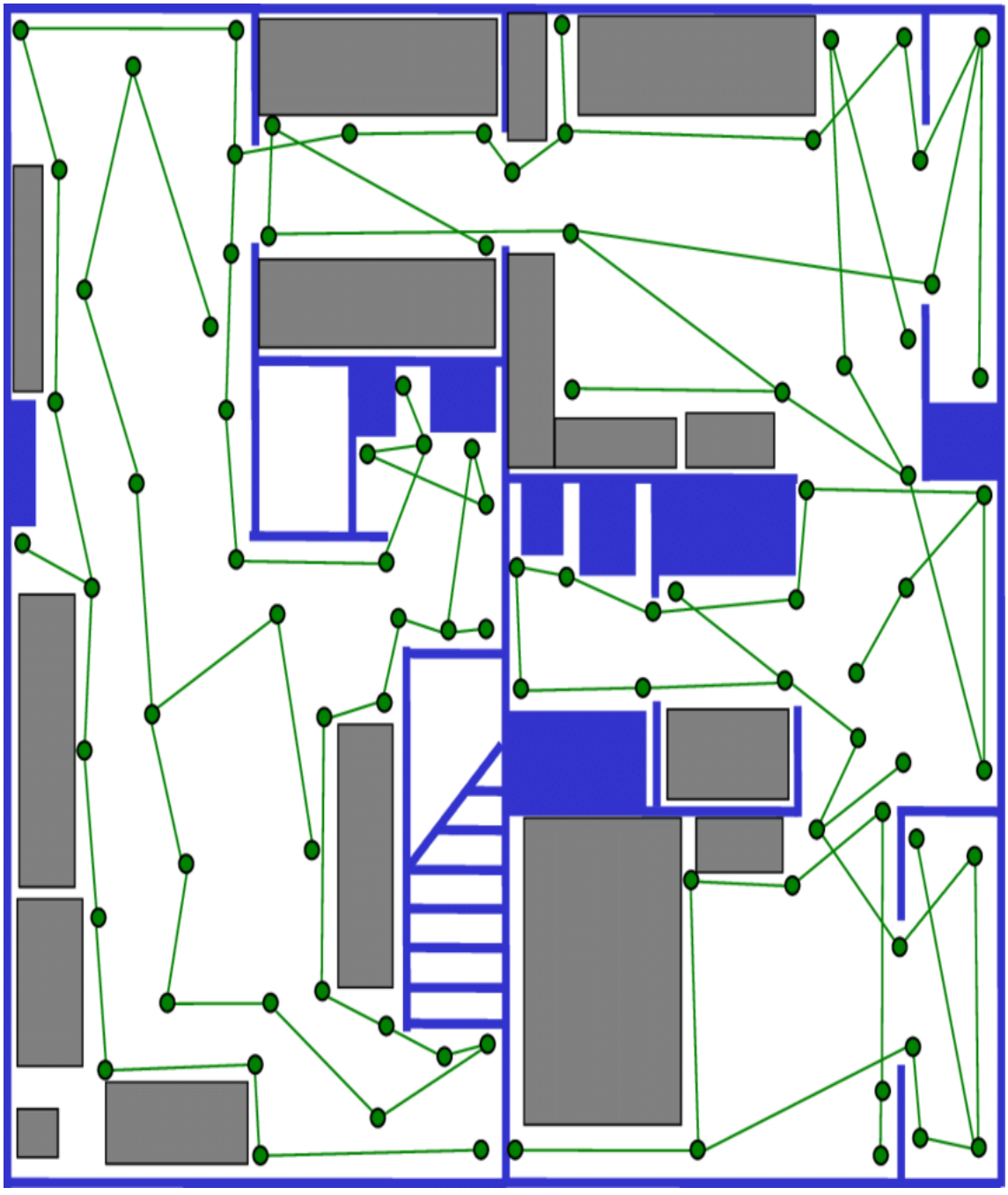


Рисунок 1.3 – Приклад автоматичного картографування місцевості

1.5 Навігація мобільних роботів

Навігація мобільних роботів є важливою проблемою в галузі робототехніки. Вони відомі своїми інтелектуальними тенденціями. Вони також охоплюють широкий спектр застосувань, таких як транспорт, промисловості, а також в роботах-рятувальниках.

Планування маршруту є однією з найбільш важливих частин навігації мобільних роботів. Протягом останніх двох десятиліть дослідники працюють над проблемою планування шляху, для якої розроблено декілька методів.

Планування шляху включає в себе визначення шляху без зіткнень з однієї точки в іншу при мінімізації загальної вартості шляху. Залежно від характеру середовища, планування шляхів можна розділити на для статичного та динамічного середовища.

Якщо перешкоди змінюють своє положення відносно часу, то це називається статичним плануванням шляху, а якщо перешкоди змінюють своє положення і орієнтацію відносно часу, то це називається динамічним плануванням шляху.

Над удосконаленням автономної навігації мобільних роботів виконано багато робіт. Існує три загальних підходи до навігації мобільних роботів

– автономна;

– телекерована;

– напвіавтономна.

Автономний підхід фокусується на розробці систем, які дозволяють роботам орієнтуватися в середовищі з дуже незначним контролем з боку людини або взагалі без нього. Цей підхід має на меті зменшити потребу в наявності кваліфікованого оператора, який керує роботом. Такі системи, однак, часто потребують декількох датчиків і передових алгоритмів, що може
бути як фінансово, так і обчислювально дорогими. Надійність також викликає занепокоєння, оскільки робот може вийти з ладу і завдати шкоди, якщо людина-оператор не вживатиме екстрених реагування на надзвичайні ситуації.

Телекерований, або "безпілотний", підхід має на меті розширити можливості здатності людини-оператора повністю контролювати мобільного робота. В даний час цей підхід використовується у безпілотних літальних апаратах військового типу (БПЛА). Перевагою такого підходу є те, що система може бути відносно недорогою, з точки зору необхідних датчиків, і недорогою в обчислювальному плані оскільки планування траєкторії здійснюється оператором. Недоліком є те, що він може вимагати високий ступінь концентрації уваги і навичок людини для ефективного управління роботом, особливо для тривалих місій.

Напівавтономний підхід забезпечує компроміс між двома вищезгаданими підходами. Робот може самостійно здійснювати навігацію, але при цьому присутня людина-оператор для спостереження, командування або безпосереднього управління роботом за необхідності.

На практиці більшість робототехнічних систем мають напівавтономний характер, що обумовлено необхідністю наявності людини-оператора для забезпечення безпечної та ефективної роботи.

У рамках магістерської роботи буде детально розглянуто автономний підхід, система підтримки прийняття рішень буде пропонувати особі, що приймає рішення саме автономні варіанти варіантів вирішення завдань переміщення мобільної платформи за визначеними параметрами. Виходячи з рамок поставленої задачі ОПР може комбінувати запропоновані автономні варіанти вирішення завдань з телекерованими і реалізовувати напівавтономний підхід.

1.6 Автономний підхід до навігації мобільних роботів

Базуючись на аналізі існуючих досліджень [7] було виявлено що загалом виділяють наступні основні підходи при розробці автономних навігаційних систем мобільних роботів:

1.6.1 Генетичні алгоритми

Генетичний алгоритм (ГА) – це еволюційний метод вирішення проблем, де рішення проблеми еволюціонує після ряду ітерацій . Генетичний алгоритм починається з популяції особин (хромосом). Кожна особина представляє можливе рішення для даної проблеми. Для навігації робота окрема особина може представляти шлях між початковою та цільовою точками. Кожному індивідууму присвоюється значення придатності, засноване на тому, наскільки добре індивідуум відповідає цілям задачі.

Використовуючи ці значення придатності, особини обираються в якості батьків. Ці батьки утворюють нових особин, або потомство, за допомогою кросинговеру і мутацій. Відбір батьків, кросинговер і мутації тривають протягом декількох ітерацій (поколінь) до тих пір, поки алгоритм не збігається до оптимального або близького до оптимального рішення [8] це є типовою схемою роботи генетичного алгоритму (рисунок 1.4).

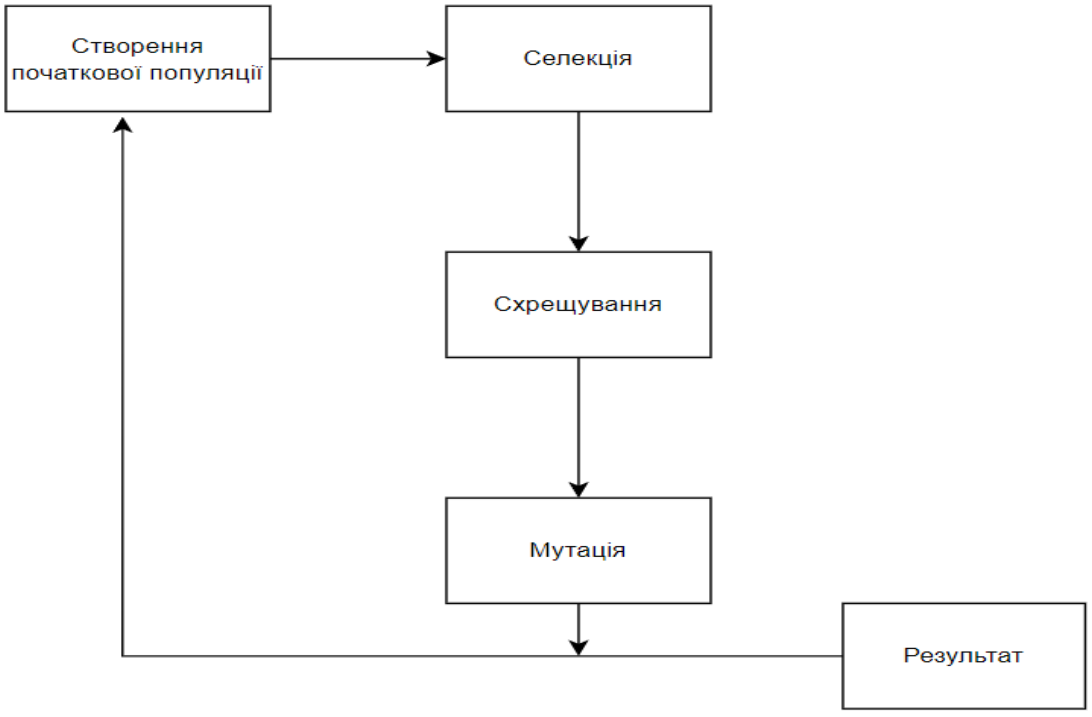


Рисунок 1.4 – Схема роботи генетичного алгоритму

Основною відмінною рисою генетичних алгоритмів є спосіб представлення простору рішень для обраної проблеми. Кожна особина в популяції повинна представляти дійсну точку в просторі рішень. Рішення кодуються в генотипи - це моделі представлення дизайну, якими можна маніпулювати за допомогою генетичного алгоритму.

Для оцінки кожного генотипу по відношенню до цілей задачі використовується функція пристосованості. Таким чином, домінуючим завданням при розробці генетичного алгоритму для вирішення конкретної задачі є розробка структури генотипу і операторів, які обробляють цю структуру, таких як фітнес-функція, оператори кросинговеру і мутації.

Для навігації роботів метою є якнайшвидше визначення можливого шляху через невизначене середовище. Тому генотип повинен бути здатним представляти дійсний шлях в навігаційному просторі.

Крім того, структури генотипу і функції пристосованості повинні бути відносно простими, щоб їх можна було швидко обробляти.

1.6.2 Штучні нейроні мережі

Нейронні мережі мають справу з такими когнітивними завданнями, як навчання, адаптація, узагальнення та оптимізація. Дійсно, розпізнавання, навчання, прийняття рішень та дії є основними проблемами навігації. Для вирішення цих завдань можна викроистовувати нейронні мережі. Вони покращують можливості навчання та адаптації, пов'язані з варіаціями в навколишньому середовищі, де інформація є якісною, неточною, невизначеною або неповною.

Сукупності з'єднаних вузлів, що називають штучними нейронами (аналогічно до біологічних нейронів у головному мозку тварин). Кожне з'єднання (аналогічне синапсові) між штучними нейронами може передавати сигнал від одного до іншого. Штучний нейрон, що отримує сигнал, може обробляти його, й потім сигналізувати штучним нейронам, приєднаним до нього.

Обробка неточних або зашумлених даних за допомогою нейронних мереж є більш ефективною, ніж класичні методи, оскільки нейронні мережі мають високу толерантність до шумів.

Нейронна мережа - це масивна система паралельно працюючих розподілених обчислювальних елементів (нейронів), з'єднаних у графовій топології. При цьому підході до навігації може використовуватися багато видів штучних нейронних структур (рисунок 1.5).

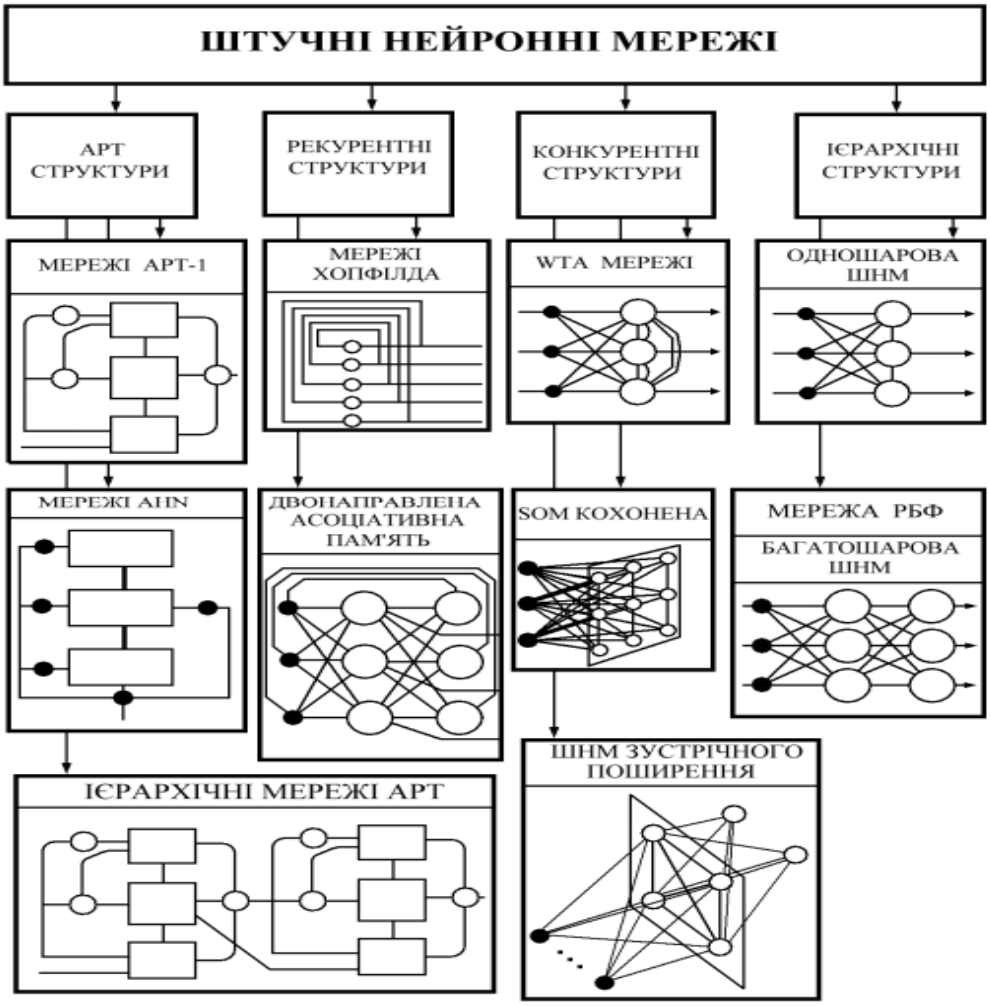


Рисунок 1.5 – Штучні нейронні структури [9]

Навчання в нейронній мережі може бути контрольованим або неконтрольованим. Контрольоване навчання використовує засекречену інформацію про моделі, тоді як неконтрольоване навчання використовує лише

мінімум інформації без попереднього засекречування. Алгоритми неконтрольованого навчання пропонують меншу обчислювальну складність і меншу точність, ніж алгоритми контрольованого навчання. Перші навчаються швидко, часто на одному проході зашумлених даних. Після навчання нейронна мережа може виражати знання неявно у вагах.

Використовуються конкурентні нейронні мережі для керування та генерації траєкторій для роботів, а також нейронної мережі для обробки сенсорних даних при оновленні карт та траєкторій роботів.

Для цілей об'їзду перешкод використовувався рекурентний тип нейронної мережі з методом градієнтного зворотного поширення для навчання мережі.

1.6.3 Системи нечіткої логіки

Надійна навігація в динамічному або невідомому середовищі залежить від здатності роботів рухатися серед невідомих перешкод без зіткнень та швидкої реакції на невизначеності.

Нечітка логіка використовує евристичні знання для представлення та реалізації методології розробки стратегій сприйняття-дії для навігації мобільних роботів. Крім того, ця методологія є дуже корисною при роботі з невизначеностями в реальному світі, і точна модель навколишнього середовища не є абсолютно необхідною для навігації.

Було розроблено багато підходів до вирішення навігаційних задач мобільних роботів відстеження цілі, відстеження шляху, об'їзду перешкод, координації поведінки та моделювання середовища.

Для підвищення загальної продуктивності навігаційної системи складні навігаційні завдання розбиваються на ряд більш простих і менших підсистем (поведінок), що називається поведінковою системою (рисунок 1.6).

У системі, що базується на поведінці, кожна поведінка отримує певну сенсорну інформацію і трансформує її у заздалегідь визначену реакцію. Поведінка включає відстеження шляху, уникнення перешкод, відстеження цілі,

досягнення цілі тощо. Нарешті, на основі командних виходів активної поведінки робот виконує дію.

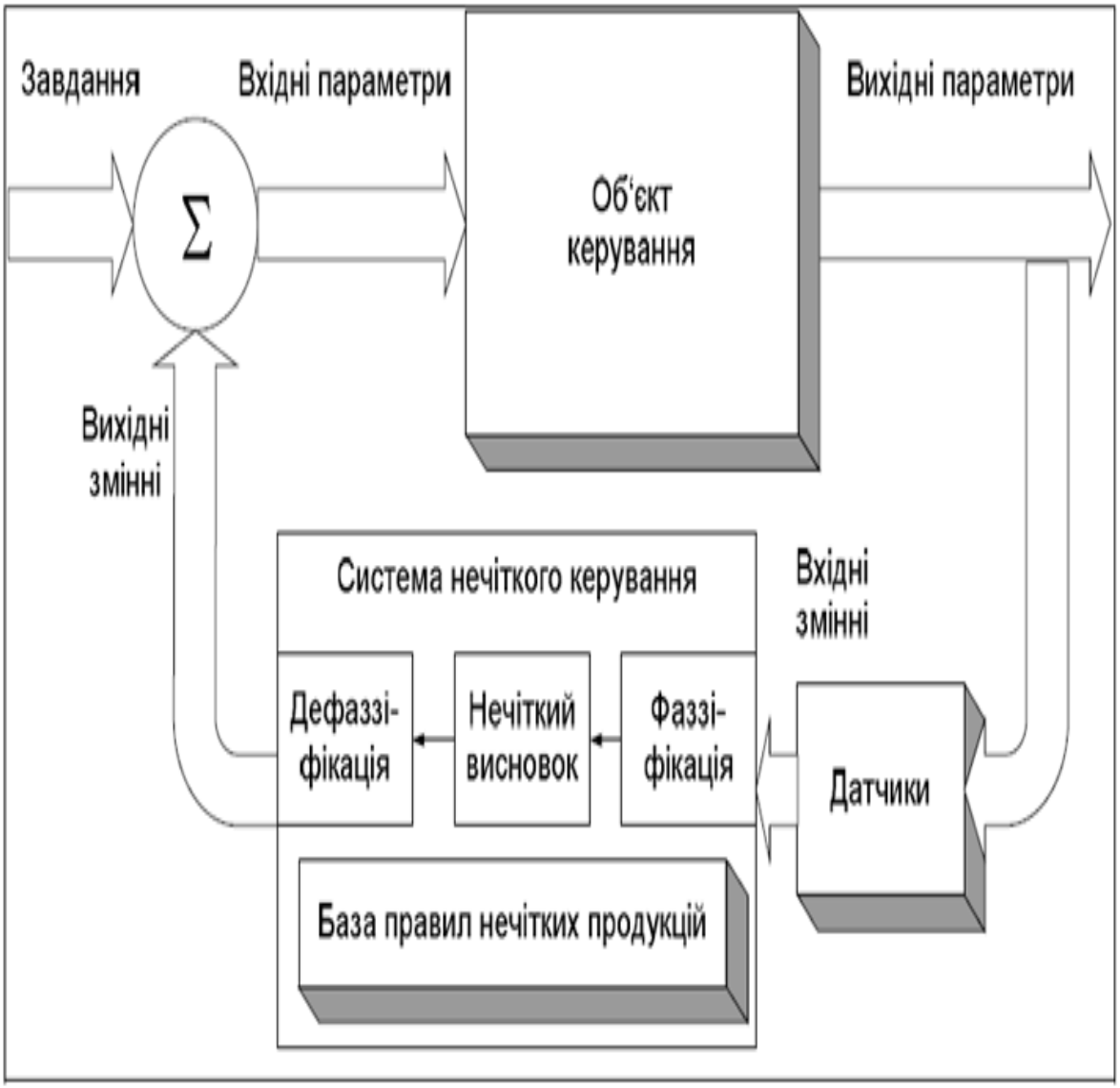


Рисунок 1.6 – Загальна схема нечіткого керування [10]

Проблеми, пов'язані з навігаційними системами, заснованими на поведінці, полягають у координації поведінки або виборі дій. Різноманітна поведінка може викликати одночасно декілька командних виходів, що може призвести до руху робота в непередбачених напрямках або до повного виходу системи з ладу.

Надійна і стійка робота системи залежить від рішення про те, як інтегрувати поведінку високого рівня планування і низького рівня виконання,

яка поведінка повинна бути активована (арбітраж) і як вихідні команди повинні бути об'єднані в одну команду для управління роботом (злиття команд).

1.6.4 Пошук за першим найкращим збігом

Це алгоритми пошуку, які використовують інтерпретацію середовища у вигляді графа та досліджують його шляхом розширення найперспективніших вузлів, які обираються відповідно до визначеного правила [11].

Пошук за цим принципом виконується шляхом вибору вузла з найкращим значенням f у відкритому списку для розширення. Вузол видаляється зі списку Open і створюються всі його дочірні вузли.

Для кожної дочірньої вершини, що генерується, метод пошуку найкращого варіанту перевіряє списки Open і Closed, щоб переконатися, що вона не є дублікатом раніше згенерованої вершини, або, якщо це так, що вона представляє кращий шлях, ніж раніше згенерований дублікат. Якщо це так, то вузол вставляється в закритий список і вибирається новий вузол для розширення. Це продовжується до тих пір, поки для розширення не буде обраний цільовий вузол.

Різні алгоритми пошуку найкращого з перших будуються шляхом вибору різних оціночних функцій. Часто функція оцінки будується шляхом об'єднання вимірювання вартості, понесеної на частковому шляху рішення, що позначається функцією $g(n)$, і оптимістичної оцінки вартості будь-якого шляху, що випливає з вузла, що позначається функцією $h(n)$. Алгоритм пошуку найкращого з перших використовує функцію оцінки (1.1).

$$f(n) = g(n) + h(n), \tag{1.1}$$

де $f(n)$ – функція оцінки;

$g(n)$ – функція вимірювання вартості на частковому шляху рішення;

$h(n)$ – функція оптимістичної вартості шляху.

У просторі пошуку в порядку виключення вартістю повного шляху розв'язку є ширина відповідного порядку. Вона обчислюється шляхом взяття максимальної вартості усунення вздовж шляху розв'язку, де вартість конкретного усунення є ступенем вершини, що усувається. Таким чином, розумною g-функцією буде максимальна вартість при досягненні вершини. Оптимальна h-функція буде оптимістичною оцінкою максимальної вартості на деякому шляху від поточного стану до цільового стану. Вони можуть бути об'єднані в оціночну функцію (1.2).

$$f(n) = max\, g(n), h(n), \tag{1.2}$$

де $f(n)$ – оціночна функція;

$g(n)$ – оптимальна функція вартості при досягненні вершини;

$h(n)$ – оптимальна оптимістична функція оцінки максимальної вартості.

1.7 Висновки до першого розділу

Проаналізувавши наведені основні підходи при розробці автономних навігаційних систем мобільних роботів та ґрунтуючись на існуючих дослідженнях [10], [11] можна зробити висновок, що базуючись на простоті реалізації, швидкості обчислення, адаптивності до різноманітних видів мобільних роботові та універсальності застосування у різних ситуаціях найкращим підходом по сумі наведених вище критерієві є саме пошук за першим найкращим збігом, у наведених дослідженнях він був представлений алгоритмом A*.

У рамках цієї роботи існуючи дослідження будуть доповненні аналізом характеристик алгоритму A* в залежності від параметрів евристичної функції. Буде проведено аналіз інших алгоритмів пошуку за першим найкращим збігом на основі алгоритму A* на основі чого можна буде зробити порівняння показників таких як:

– оптимальність отриманого шляху;

– споживання пам’яті;

– швидкість роботи;

Буде проаналізовано залежність цих показників від ступеню дослідження мапи навколишнього середовища на початок роботи засобами комп’ютерного моделювання, та реалізована система підтримки прийняття рішень для завдань переміщення мобільної платформи на основі отриманих результатів.

2 ДОСЛІДЖЕННЯ АЛГОРИТМІВ ПОШУКУ ЗА ПЕРШИМ НАЙКРАЩИМ ЗБІГОМ

2.1 Евристичні функції визначення відстані

Усі алгоритми, які розглянуті в цьому розділі використовують евристичну функцію визначення відстані. Вибір евристичної функції визначення відстані впливає на час роботи алгоритму та на оптимальність знайденого шляху.

2.1.1 Евклідова відстань

У математиці евклідова відстань між двома точками (рисунок 2.1) в евклідовому просторі - це довжина відрізка прямої між двома точками. Її можна обчислити з декартових координат точок за допомогою теореми Піфагора.

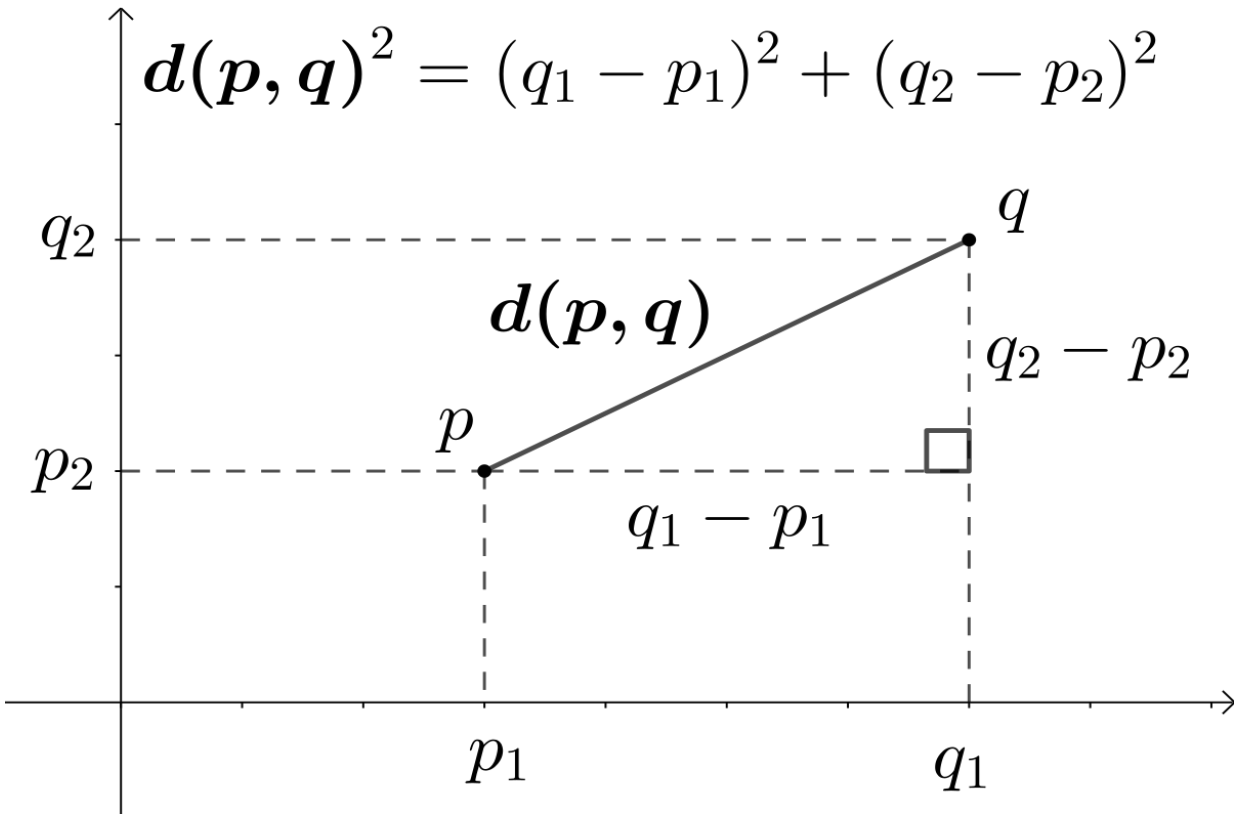


Рисунок 2.1 – Евклідова відстань між двома точками

Відстань між двома об'єктами, які не є точками, зазвичай визначається як найменша відстань між парами точок цих двох об'єктів. Відомі формули для обчислення відстаней між різними типами об'єктів, наприклад, відстань від точки до прямої.

Використання Евклідової відстані надає змогу у більшості ситуацій знайти наймень короткий шлях (рисунок 2.2).

			3			
	2.8	2.2	2	2.2	2.8	
	2.2	1.4	1	1.4	2.2	
3	2	1	0	1	2	3
	2.2	1.4	1	1.4	2.2	
	2.8	2.2	2	2.2	2.8	
			3			

Рисунок 2.2 – Відстані вирахуванні за допомогою Евклідової відстані

2.1.2 Манхеттенська відстань

Манхеттенська відстань – метрика відстані між двома точками в N-вимірному векторному просторі. Являє собою суму довжин проекцій відрізка прямої між точками на осі координат. Простіше кажучи, це сума абсолютної різниці між мірами у всіх вимірах двох точок.

Манхеттенською відстанню між двома векторами в n-вимірному дійсному векторному просторі з фіксованою декартовою системою координат називається сума довжин проекцій відрізка прямої між точками на осі координат за формулою (2.1).

$$d(p,q) = \|p - q\| = \sum_{i=1}^{n} |p_i - q_i|, \tag{2.1}$$

де $d(p,q)$ – манхеттенська відстань;

n – вимірність векторного простору;

q, r – вектори, між якими обраховується відстань.

Відстані, вирахуванні за допомогою Манхеттенської відстані можна побачити на рисунку 2.3.

			3			
		3	2	3		
	3	2	1	2	3	
3	2	1	0	1	2	3
	3	2	1	2	3	
		3	2	3		
			3			

Рисунок 2.3 – Відстані вирахуванні за допомогою Манхеттенської відстані

2.1.3 Відстань Чебишова

Відстань Чебишова – метрика, визначена на векторному просторі, де відстань між двома векторами дорівнює найбільшій з їх різниць вздовж будь-якого координатного виміру.

Відстань Чебишова також відома як відстань на шахівниці, оскільки в грі в шахи мінімальна кількість ходів, необхідна королю для переходу з однієї клітини шахівниці на іншу, дорівнює відстані Чебишова між центрами клітин, якщо клітини мають довжину сторони одиниця, зображеної в двовимірних просторових координатах з осями, вирівняними по краях дошки.

Математично, відстань Чебишова є метрикою, індукованою супремум-нормою або рівномірною нормою. Вона є прикладом ін'єктивної метрики. У двовимірному просторі, якщо точки p та q мають декартові координати, їх відстань Чебишова обчислюється за формулою (2.2).

(2.2)

$$d = max_i(|x_i - y_i|),$$

де d – відстань Чебишова;

x, y – координати точок.

Чебишова, як правило, використовується в дуже специфічних випадках, що ускладнює його використання в якості універсальної метрики відстані, наприклад, Евклідової відстані або Манхеттенської. З цієї причини рекомендується використовувати її тільки тоді, коли ви абсолютно впевнені, що вона підходить для вашого випадку використання.

Як згадувалося раніше, відстань Чебишова може бути використана для визначення мінімальної кількості ходів, необхідних для переходу з однієї клітини в іншу. Крім того, вона може бути корисною коли дозволяється необмежений рух у 8 напрямках.

На практиці відстань Чебишова часто використовується у складській логістиці, оскільки вона дуже нагадує час, який потрібен мостовому крану для

переміщення об'єкта. Відстані, вирахуванні за допомогою відстані Чебишова можна побачити на рисунку 2.4.

3	3	3	3	3	3	3
3	2	2	2	2	2	3
3	2	1	1	1	2	3
3	2	1	0	1	2	3
3	2	1	1	1	2	3
3	2	2	2	2	2	3
3	3	3	3	3	3	3

Рисунок 2.4 – Відстані вирахуванні за допомогою відстані Чебишова

2.2 Алгоритм A*

2.2.1 Загальний опис алгоритму

Це алгоритм пошуку, який використовується для знаходження найкоротшого шляху між початковою та кінцевою точками.

Це зручний алгоритм, який часто використовується для обходу карти, щоб знайти найкоротший шлях. Спочатку A* був розроблений як задача обходу

графів, щоб допомогти побудувати робота, який може знайти свій власний курс. Він все ще залишається широко популярним алгоритмом для обходу графів.

Алгоритм спочатку шукає найкоротші шляхи, що робить його оптимальним і повним алгоритмом. Оптимальний алгоритм знайде найменш витратний результат для проблеми, в той час як повний алгоритм знаходить всі можливі вирішення проблеми.

Ще одним аспектом, який робить A* таким потужним, є використання зважених графів при його реалізації. Зважений граф (рис 2.5) використовує числа для представлення вартості кожного шляху або способу дій. Це означає, що алгоритми можуть обрати шлях з найменшою вартістю і знайти найкращий маршрут з точки зору відстані та часу.

Алгоритм A* – це пошуковий алгоритм, який використовується для пошуку найкоротшого шляху між початковою та кінцевою точками. Алгоритм є оптимальним і повним, оскільки спочатку шукає найкоротші спочатку шукає найкоротші шляхи. Оптимальний алгоритм знаходить найменш витратний результат для проблеми, в той час як повний алгоритм знаходить всі можливі результати [14].

При проходженні графа A* слідує шляхом з найменшою відомою вартістю, зберігаючи відсортовану чергу пріоритетів альтернативних відрізків шляху.

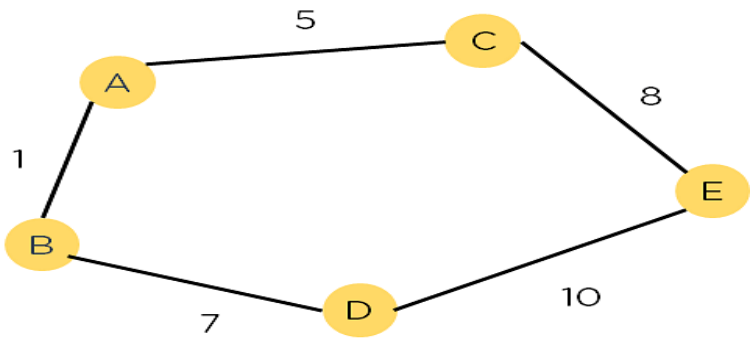


Рисунок 2.5 – Зважений граф

Алгоритм пошуку A* – це простий і ефективний алгоритм пошуку, який може бути використаний для пошуку оптимального шляху між двома вершинами графа. Він буде використовуватися для пошуку найкоротшого шляху. Він є розширенням алгоритму найкоротшого шляху Дейкстри (рисунок 2.6).

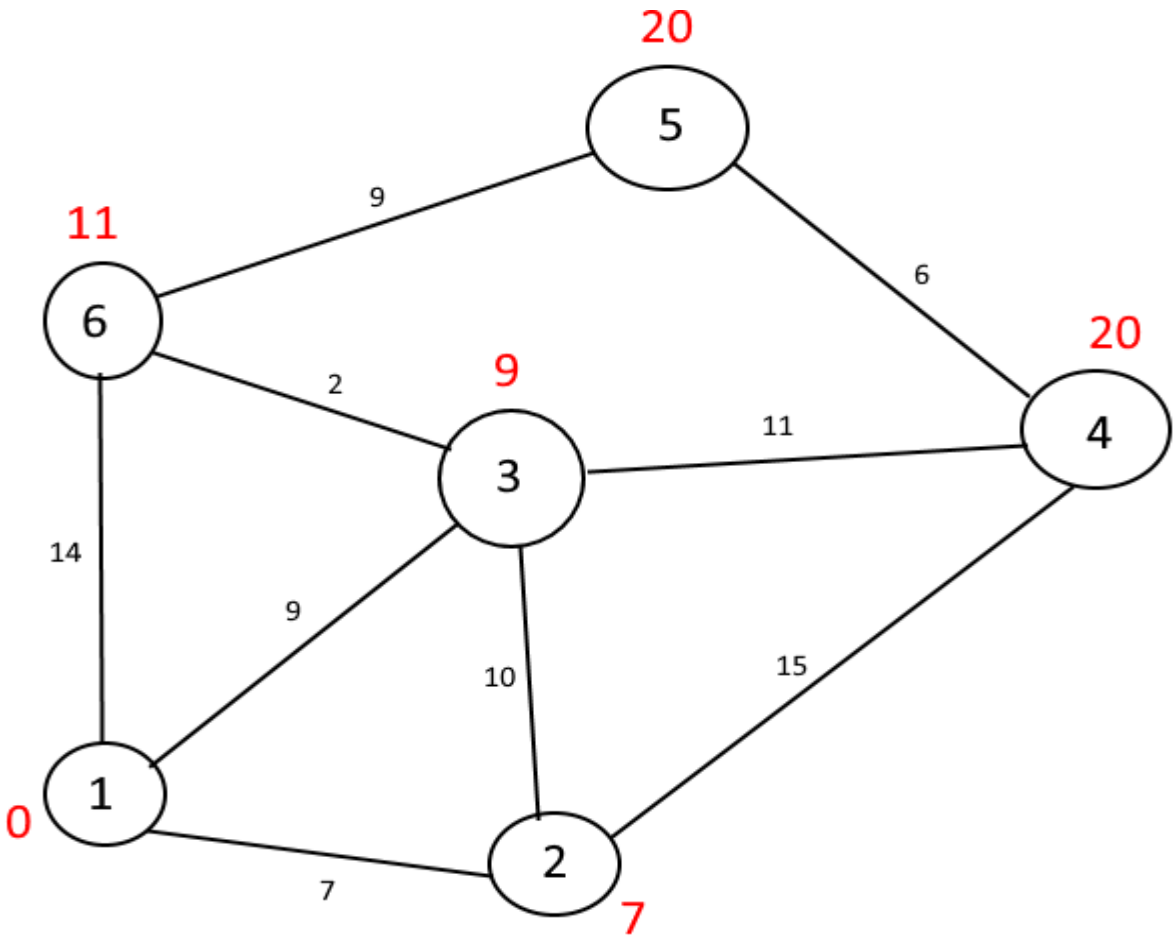


Рисунок 2.6 – Результат роботи алгоритму Дейкстри

У порівнянні з алгоритмом Дейкстри, алгоритм A* знаходить лише найкоротший шлях від заданого джерела до заданої мети, а не дерево найкоротшого шляху від заданого джерела до всіх можливих цілей. Це необхідний компроміс для використання евристики, орієнтованої на конкретну мету. Для алгоритму Дейкстри, оскільки генерується все дерево найкоротшого шляху, кожен вузол є метою, і не може бути евристики, орієнтованої на конкретну мету.

Типові реалізації A* використовують чергу пріоритетів (рисунок 2.7) для виконання повторного вибору вузлів з мінімальною (оціночною) вартістю для розширення [15]. Ця черга пріоритетів називається відкритою множиною.

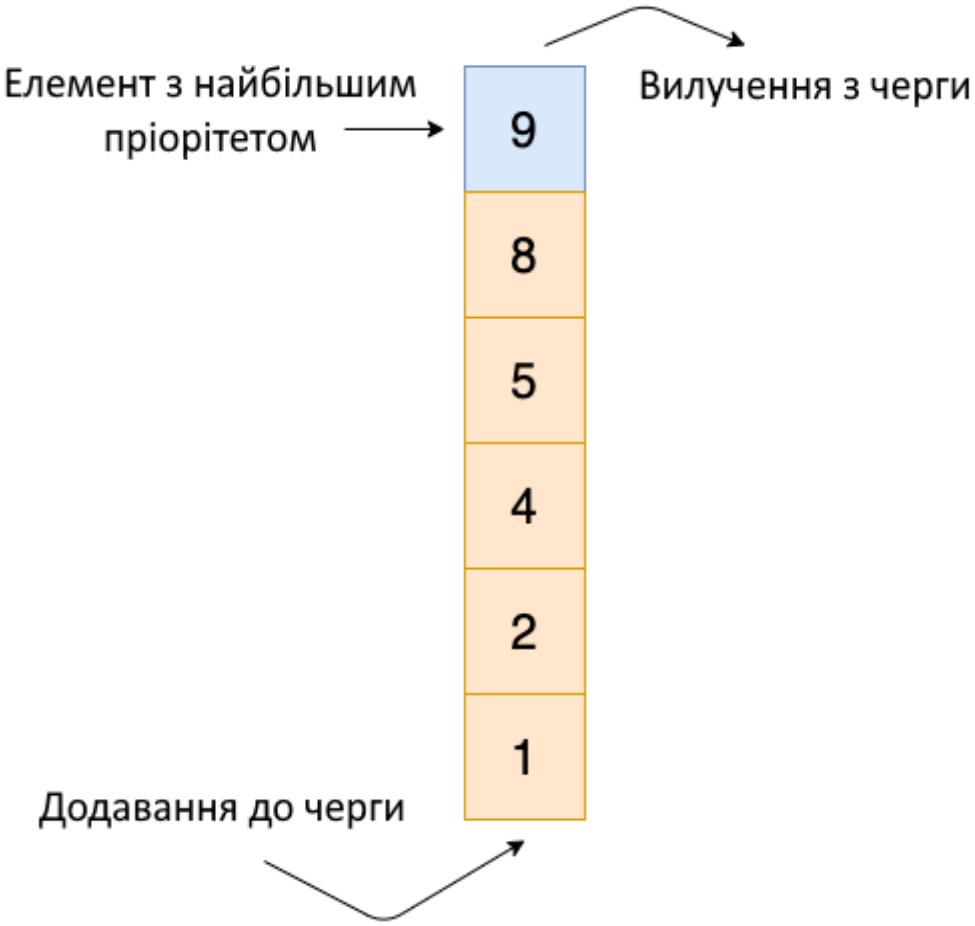


Рисунок 2.7 – Черга пріоритетів

На кожному кроці алгоритму з черги видаляється вершина з найменшим значенням f(x), відповідно оновлюються значення f і g її сусідів, і ці сусіди додаються до черги. Алгоритм продовжується до тих пір, поки видалений вузол (тобто вузол з найменшим значенням f з усіх вузлів периферії) не стане цільовим вузлом. Значення f цього вузла є також вартістю найкоротшого шляху, оскільки h в цільовому вузлу дорівнює нулю в допустимій евристиці [16].

Описаний алгоритм дає нам лише довжину найкоротшого шляху. Для знаходження дійсної послідовності кроків алгоритм може бути легко

переглянутий таким чином, щоб кожен вузол на шляху відстежував свого попередника. Після запуску такого алгоритму кінцевий вузол буде вказувати на свого попередника, і так до тих пір, поки попередник деякого вузла не стане початковим вузлом.

Наприклад, при пошуку найкоротшого маршруту на карті, h(x) може представляти відстань по прямій до цілі, оскільки це фізично найменша можлива відстань між будь-якими двома точками. Для сітчастої карти краще використовувати Евклідову відстань, залежно від набору доступних рухів (4-сторонній або 8-сторонній).

Алгоритм A* називається оптимально ефективним відносно множини альтернативних алгоритмів Alts на множині задач P, якщо для кожної задачі P в P і кожного алгоритму A′ в Alts множина вершин, розширених A при розв'язанні P, є підмножиною (можливо, рівною) множини вершин, розширених A′ при розв'язанні P. Остаточне дослідження оптимальної ефективності A* належить Ріні Дехтер і Джуді Перл [15]. Вони розглядали різні визначення Alts і P в поєднанні з евристикою A*, яка була просто допустимою або була одночасно і послідовною, і допустимою.

Найцікавіший позитивний результат, який вони довели, полягає в тому, що A* з послідовною евристикою є оптимально ефективним по відношенню до всіх допустимих A*-подібних алгоритмів пошуку на всіх проблемах пошуку.

Цей результат не зберігається, якщо евристика A* допустима, але не є послідовною. У цьому випадку існують допустимі A*-подібні алгоритми, які можуть розширити як завгодно меншу кількість вершин, ніж A*, на деяких непатологічних задачах.

Оптимальна ефективність залежить від набору розширених вершин, а не від кількості розширень вершин (кількості ітерацій основного циклу A*). Коли евристика, що використовується, є припустимою, але непослідовною, можливо, що вузол може бути розширений A* багато разів, у найгіршому випадку - експоненціальну кількість разів.

За таких обставин алгоритм Дейкстри міг би перевершити A* з великим відривом від нього. Однак більш пізні дослідження виявили, що цей патологічний випадок виникає лише в певних надуманих ситуаціях, коли вага ребра пошукового графа експоненціально залежить від розміру графа, і що певні непослідовні (але допустимі) евристики можуть призвести до зменшення кількості розширень вузлів у пошуках A*.

2.2.2 Переваги і недоліки

Як перевагу алгоритму можна виділити те, що він найбільш простий алгоритм за кількістю інструкцій, теоретично має найменший час виконання для повністю відомої мапи.

Як недолік алгоритму можна виділити не адаптованість алгоритму до умови динамічного навколишнього середовища, якщо на спланованому шляху буде виявлена перешкода – алгоритм має відпрацьовувати заново не базуючись на попередніх обчисленнях.

2.3 Алгоритм Theta*

2.3.1 Загальний опис алгоритму

Theta* - алгоритм планування траєкторії під будь-яким кутом. Як випливає з назви, алгоритми пошуку шляху під будь-яким кутом дозволяють своїм шляхам прямувати під будь-яким кутом [18]. Для алгоритмів Theta* це означає, що навіть якщо вони засновані на A* і, отже, поширюють інформацію вздовж ребер сітки поширюють інформацію вздовж ребер сітки, на відміну від A*, вони не обмежують свої шляхи цими ребрами сітки. Дозвіл на відхилення шляху від сітки дозволяє знаходити коротші і більш більш природні шляхи [19].

Алгоритм майже ідентичний A* і відрізняється лише оновленням g-значення та предків нерозширених сусідніх вершин n' з n, при розширенні n. A*

розглядає лише шлях вздовж ребер сітки від n_{start} до n, та від n' до n (рисунок 2.8 шлях 1). Theta* додатково розглядає шлях від n_{start} до предку n, предок(n) та від батька n до n' по прямій лінії (рисунок 2.8 шлях 2).

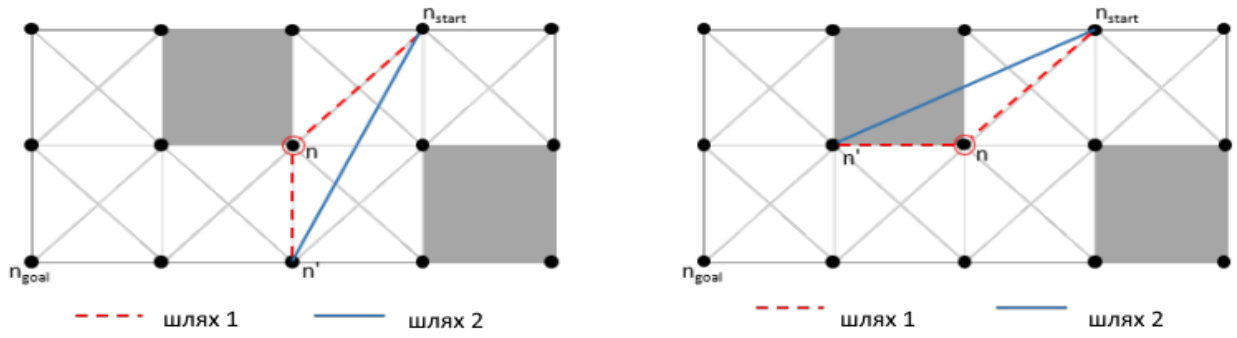


Рисунок 2.8 – Шляхи розглянуті Theta*

Шлях 2 завжди коротший за шлях 1 і буде обраний Theta*, якщо він не заблокований. Іншими словами, коли є пряма видимість між n' та предком n. В іншому випадку буде обрано шлях 1.

На рисунку 2.9 показаний приклад розгортки Theta*. Для простоти несуттєві вилучення вузлів не показані. Червоні кола вказують на те, який вузол в даний момент вилучається.

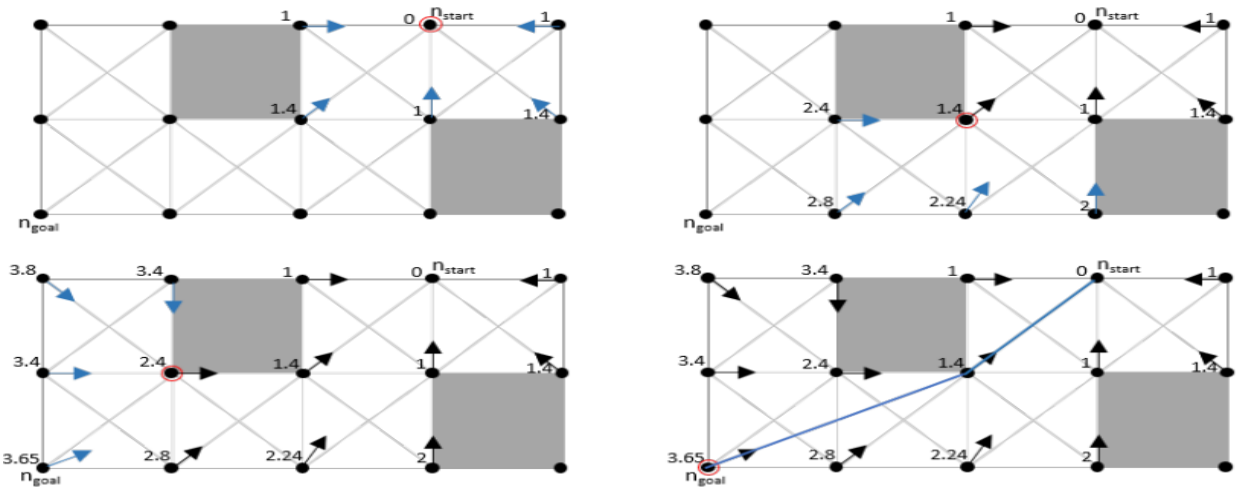


Рисунок 2.9 – Приклад роботи Theta*

2.3.2 Переваги і недоліки

Як перевагу алгоритму можна виділити, що завдяки пошуку шляху під будь-яким кутом дозволяє знайти шлях коротший ніж шлях знайдений за допомогою A*.

Серед недоліків алгоритму можна виділити наступні:

– перевірки прямої видимості є затратними, теоретично алгоритм більш повільний ніж A*;

– не адаптованість алгоритму до умови динамічного навколишнього середовища, якщо на спланованому шляху буде виявлена перешкода – алгоритм має відпрацьовувати заново не базуючись на попередніх обчисленнях.

2.4 Алгоритм D* Lite

2.4.1 Загальний опис алгоритму

Розглянемо задачу навігації робота на невідомій місцевості, в якій робот завжди дивиться, які з восьми сусідніх клітинок є прохідними, а потім переміщується з вартістю одиниця в одну з них. Робот стартує у початковій клітині і повинен перемістити до кінцевої клітини. Він завжди обчислює найкоротший шлях від своєї поточної клітинки до цільової клітинки, припускаючи, що клітинки з невідомим статусом блокування є прохідними. Потім він слідує цим шляхом до тих пір, поки не досягне цільової клітини, в цьому випадку він успішно зупиняється, або він спостерігає непрохідну клітину, в цьому випадку він заново обчислює найкоротший шлях від своєї поточної клітини до цільової клітини [21].

D* Lite - це алгоритм пошуку шляху, який дозволяє знайти оптимальний шлях між початковою і кінцевою точкою у відомому або частково відомому середовищі. Цей алгоритм був розроблений для роботи на графовій структурі даних, що складається з вузлів, з'єднаних ребрами. Вузли, які вказують на

поточну позицію, на якій знаходиться робот, називаються вузлами-попередниками, а вузли, на які вказує поточний вузол, називаються його наступниками. Алгоритм працює, відстежуючи дві оцінки для кожного вузла на заданому графі. Перша оцінка - це оцінка G. Це значення відстежує вартість досягнення позиції на графі. Її можна розглядати як найкоротший шлях, який з'єднує початкову точку з поточною точкою. Для початкової точки показник G дорівнює нулю, тому що робот вже знаходиться там [22].

Оцінка RHS, з іншого боку, є більш корисною оцінкою. Ця оцінка діє як погляд на один крок вперед, і в алгоритмі вона використовується для пошуку наступної оптимальної позиції, до якої слід рухатися.

Якщо наступна позиція є перешкодою, витрати на з'єднання будуть встановлені на нескінченність, тому що навігатор ніколи не зможе досягти цієї точки. В результаті, оцінка RHS також дорівнює нескінченності. Використовуючи ці дві оцінки, алгоритм поширює шлях від кінцевої точки до початкової, врешті-решт знаходячи оптимальний шлях від початку до кінця за формулою (2.3).

(2.3)

$$\mathrm{rhs(n)} = min_{n`\epsilon_{succ(n)}}(g(n`) + c(n, n`),$$

де $rhs(n)$ – rhs значення;

$g(n)$ – оптимальна функція вартості при досягненні вершини;

$\mathrm{c}(n, n`)$ – функція вартості переміщення.

Оцінка G та оцінка RHS є основою для планування та перепланування маршруту Dstar Lite.

У випадку, якщо алгоритм Dstar Lite стикається з непередбаченими перешкодами і змушений перепланувати маршрут, оцінка RHS просто перераховується з урахуванням нової вартості з'єднання для всіх необхідних вузлів.

Значення g та rhs визначають, чи є вузол узгодженим, чи ні. Якщо *g(n) = rhs(n)*, то вершина є узгодженою, інакше вважається, що вона є неузгодженою.

Крім того, якщо *g(n) > rhs(n)*, то вважається, що n є надмірно узгодженим, тоді як якщо *g(n) < rhs(n)*, то n є недостатньо узгодженим. На відміну від більшості інших алгоритмів пошуку шляху, D* Lite використовує відкритий список, а не закритий [23]. Відкритий список містить усі неузгоджені вершини.

Порядок, в якому вузли сортуються у відкритому списку, визначається значенням їх ключа. Ключове значення k(n) вузла визначається за формулою (2.4) з використанням його g та rhs-значення, а також евристики фокусування і складається з двох частин, аналогічно первинному та вторинному ключам. Якщо первинні ключи двох вузлів рівні, то їх вторинні ключі будуть використовуватися як розривач зв'язку.

$$k(n) = \min\big(g(n), rhs(n)\big) + h(n_{start}, n) + k_m; \min\,(g(n), rhs(n)), \qquad (2.4)$$

де $k(n)$ – ключове значення;

$g(n)$ – оптимальна функція вартості при досягненні вершини;

$rhs(n)$ – оптимальна функція rhs вартості при досягненні вершини;

k_m – додатковий коефіцієнт;

$h(n)$ – оптимальна оптимістична функція оцінки максимальної вартості.

Коефіцієнт k_m потрібен для того, щоб уникнути необхідності переупорядкування відкритого списку при кожній зміні початкового вузла, наприклад, через те, що робот виявив зміну вартості дуги після переміщення. Оновлюючи вартість шляху, D* Lite розглядає поточне місцезнаходження робота як початковий вузол алгоритму пошуку шляху [24].

Припускаючи, що робот перемістився з вузла *n* у вузол *n`*, де виявляє змінену вартість дуги, до $k_m$ додається *h(n, n`)*, яка в свою чергу потім враховується при обчисленні нових значень ключів.

Це необхідно, оскільки після переміщення перші ключові елементи (первинні ключі) вузлів, що знаходяться у відкритому списку, можуть зменшитися максимум на *h(n, n` )*. Аналогічно, при обчисленні нових значень ключів, їх перші елементи будуть занадто малими *h(n, n`)* по відношенню до

значень ключів, що вже є у відкритому списку. Це врівноважується додаванням k_m до всіх нових ключових значень [25].

Приклад роботи алгоритму D* Lite зображено на рисунках 2.10 – 2.11.

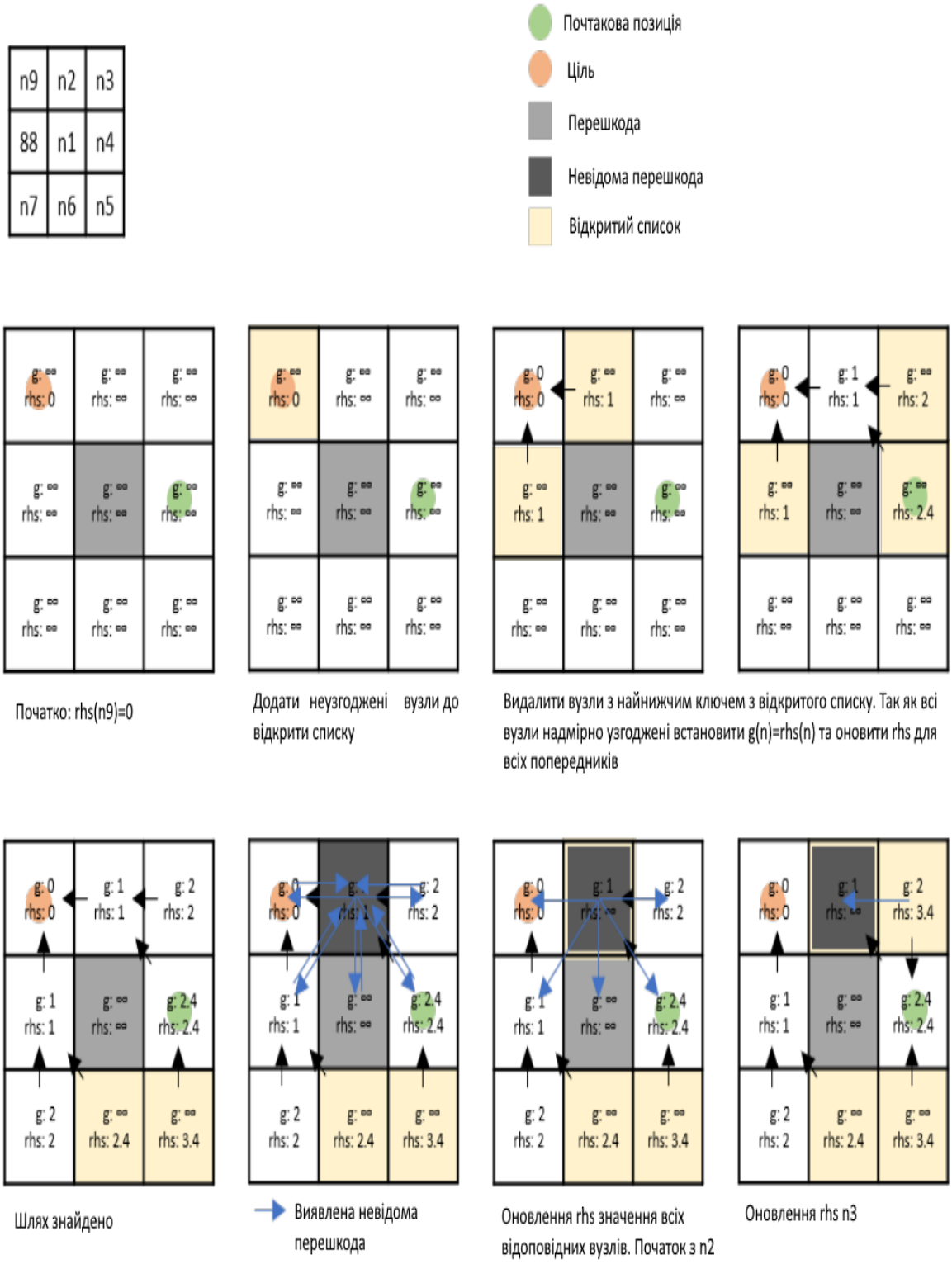


Рисунок 2.10 – Приклад роботи D* Lite

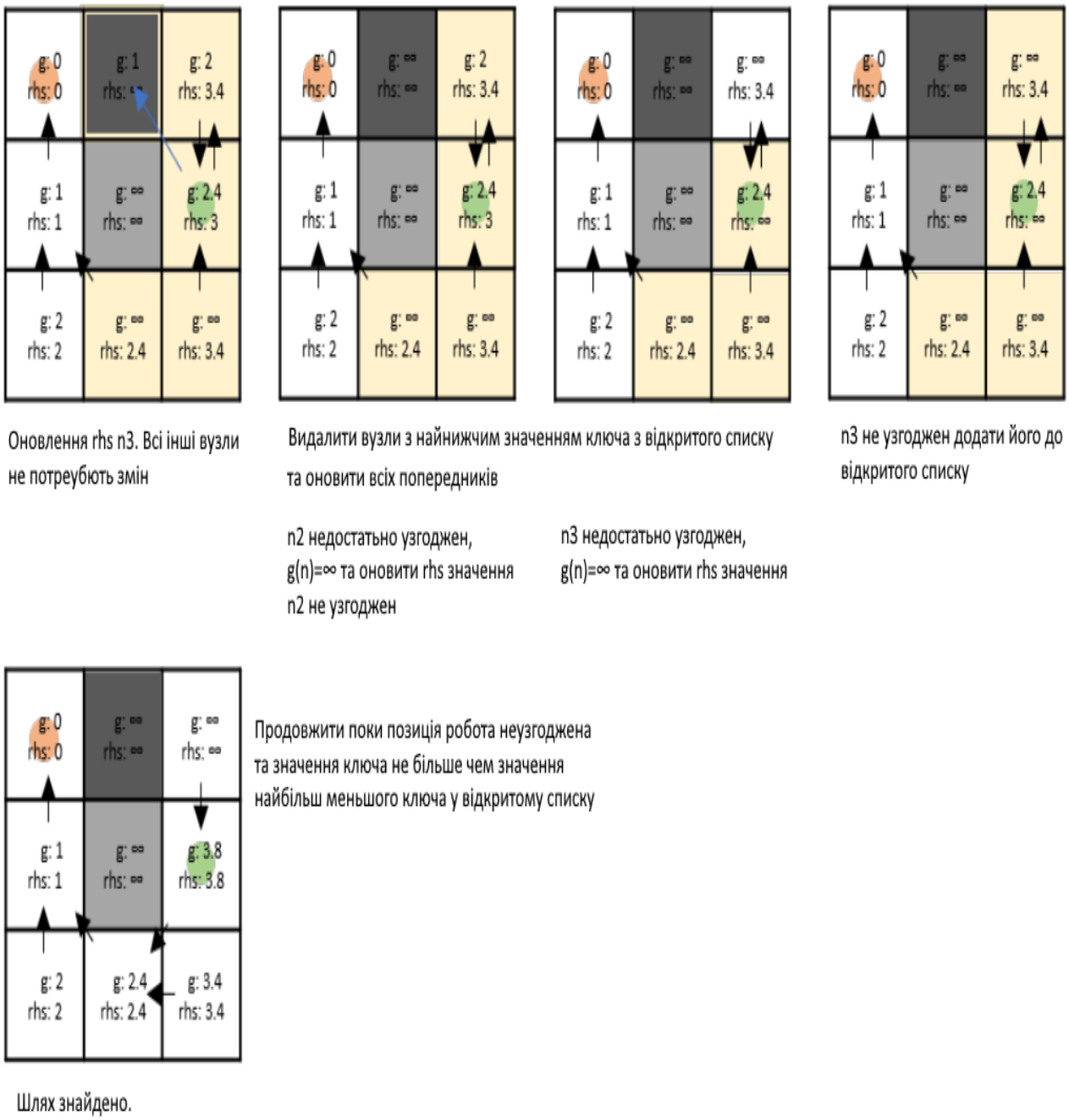


Рисунок 2.11 – Приклад роботи D* Lite

2.4.2 Переваги і недоліки

Як перевагу алгоритму можна виділити адаптованість до умов динамічного оточення.

Як недолік алгоритму можна виділити, що, теоретично в умовах статичного оточення має значно гіршу продуктивність ніж алгоритм A*.

2.5 Алгоритм Field D*

2.5.1 Загальний опис алгоритму

Field D* – це алгоритм пошуку шляху під будь-яким кутом, який використовує лінійну інтерполяцію для обчислення більш точних оцінок вартості шляху. Таким чином, він здатний знаходити плавні шляхи через неоднорідні сітки витрат, не обмежуючи рух краями сітки.

Як випливає з назви, Field D* розширює можливості D* Lite. Оскільки основним нововведенням Field D* є новий метод обчислення шляхів, він не змінює базову структуру алгоритму. Більшість планувальників шляху, таких як D* lite, використовують дискретний набір переходів, обмежуючи курс робота на кут, кратний 45°. В результаті, ці шляхи можуть бути оптимальними в дискретних середовищах, але неоптимальними в неперервних, особливо коли ціль не знаходиться під кутом, кратним 45° по відношенню до агента. Взагалі кажучи, лінійна інтерполяція не є найкращою, але вона дає хороші наближення з хорошими характеристиками [26].

По відношенню до класичних методів планування представлення середовища змінюється переміщенням вузлів з центру вузла в лівий нижній кут. Причому розглядаються не тільки переміщення на 45°, але й проміжні. Оптимальні шляхи, найімовірніше, перетинатимуть один з восьми відрізків, визначених парою вершин (рисунок 2.12).

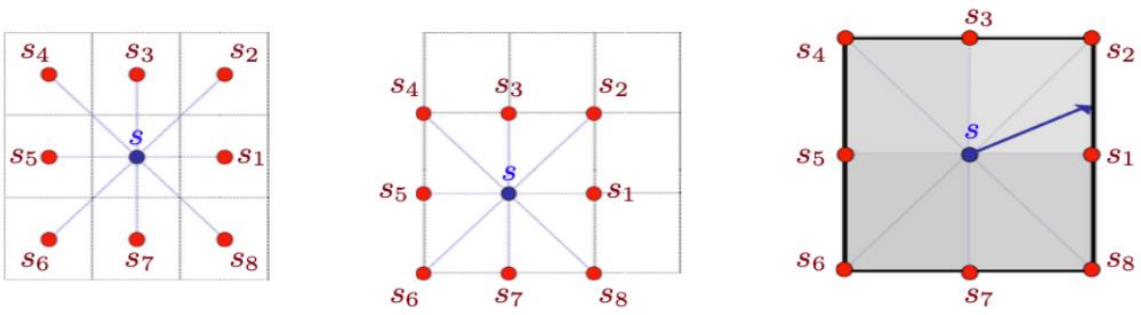


Рисунок 2.12 – Модифікації представлені в алгоритмі Field D*

Ключовим моментом цього алгоритму є те, як обчислюється вартість шляху. Отже, мінімальна сума між вартістю переходу з вузла s до одного з його сусідів та вартістю переходу до цього сусіда. Цей розрахунок припускає, що єдиними можливими переходами з вузла s є прямолінійні траєкторії до одного з його сусідів. Однак, якщо послабити ці припущення, то з вузла s можливі траєкторії в будь-якому напрямку до його меж. Отже, вартість вузла s з урахуванням двох його сусідів s1 та s2 (рисунок 2.13) за формулою (2.5) дорівнює:

$$g(s) = min_{x,y}\left[bx + c\sqrt{(1-x)^2 + y^2} + g(s_2)y + g(s_1)(1-y)\right], \qquad (2.5)$$

де $g(s)$ – вартість вузла s;

b – вартість від s до s1;

c – вартість від s до s2;

x – відстань, пройдена вздовж нижнього краю;

y – відстань, пройдена на краю між s1 та s2.

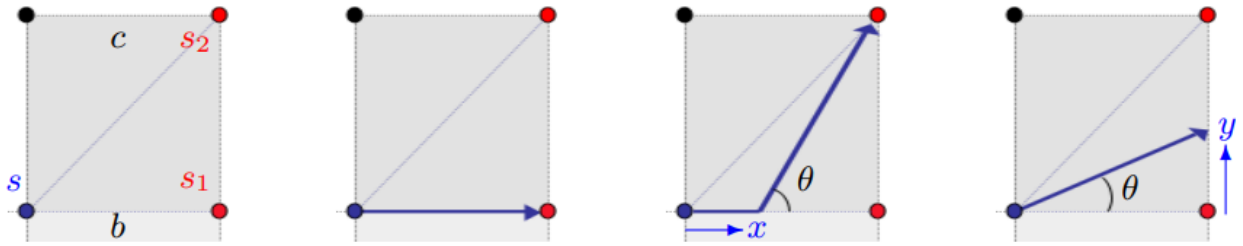


Рисунок 2.13 – Можливі обчислення вартості вузла s

У модифікованому підході вартість ребра, яке знаходиться на межі двох клітин сітки, визначається як мінімум вартості обходу кожної з двох клітин. Потім ми розглядаємо вузли нашого графа як вибіркові точки неперервного поля витрат. Оптимальний шлях з вершини s повинен проходити через ребро, що з'єднує двох послідовних сусідів s, наприклад s1s2. Таким чином, вартість шляху

з s дорівнює мінімальній вартості шляху через будь-яке з цих ребер, які розглядаються по черзі. Для обчислення вартості шляху s по ребру s1s2 використовуються вартості шляхів у вершинах s1 та s2, а також вартості обходу центральної та нижньої клітинок. Інтуїтивно зрозуміло, що якщо частковий обхід центральної клітини завжди дешевший, ніж обхід навколо границі, то найдешевше буде повністю обходити клітину. Таким чином, шлях або пройде вздовж усього нижнього краю до s1, або пройде відстань x вздовж нижнього краю, а потім по прямій безпосередньо до s2 , або пройде по прямій від s до деякої точки sy на правому краю [27].

Який з цих шляхів є найдешевшим, залежить від відносних розмірів c, b та різниці f у вартості шляху між s1 та s2: f = g(s1) - g(s2). Зокрема, якщо f < 0, то оптимальний шлях з s йде прямо до s1 і матиме вартість шляху min(c, b) + g(s1). Якщо f = b, то вартість шляху з використанням частини нижнього ребра буде еквівалентна вартості шляху без використання нижнього ребра.

2.5.2 Приклад використання алгоритму

Opportunity здійснив криволінійний 15,8-метровий маневр під час своєї 1160-ї марсіанської доби (29 квітня 2007 року) рисунок 2.14.

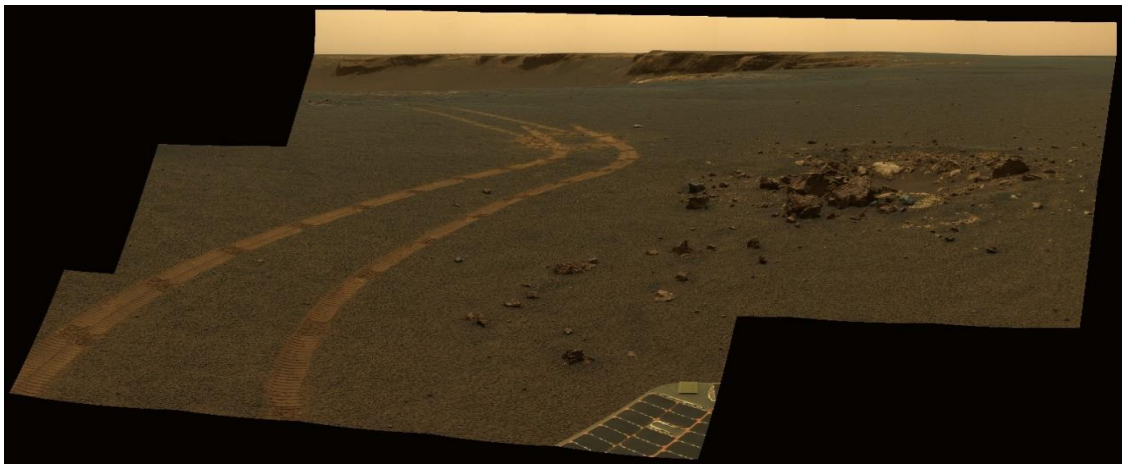


Рисунок 2.14 – Маневр здійснений марсоходом Opportunity за допомогою алгоритму Field D* [28]

Він випробовував навігаційну систему під назвою Field D*, яка дає змогу марсоходу планувати оптимальні дальні об'їзди будь-яких перешкод для того, щоб рухатись найкоротшим безпечним маршрутом до визначеного місця призначення.

Opportunity і Spirit не мали такої можливості до третього року після посадки на Марс у січні 2004 року. Раніше вони могли розпізнавати небезпеку, коли наближалися до неї, потім поверталися назад і пробували інший кут, але не завжди могли знайти безпечний маршрут подалі від небезпеки. Field D* був частиною нового бортового програмного забезпечення, завантаженого з Землі в 2006 році. Поїздка Opportunity Sol 1160 була марсіанським польовим випробуванням Field D* і також візуальну одометрію для відстеження фактичного положення марсохода після кожного сегменту поїздки, уникнення визначених заборонених зон і об'єднання інформації з двох наборів стереозображень для розгляду широкої смуги місцевості при аналізі маршруту.

2.5.3 Переваги і недоліки

Як перевагу алгоритму можна виділити, що завдяки використанню інтерполяції теоретично дозволяє отримувати більш оптимальний шлях.

Як недолік алгоритму можна виділити, що через більшу кількість обчислень та більшу кількість досліджених вузлів вимагає більше пам’яті та теоретично працює повільніше ніж D* Lite.

2.5 Висновки до другого розділу

У розділі досліджено алгоритм A* та його модифікації такі як Theta*, D* Lite, Field D* для умов статичного і динамічного оточення, визначено їх переваги та недоліки, також досліджені евристичні функції, наведено приклад використання Field D* у якості системи навігації марсоходу.

3 ПРОВЕДЕННЯ ІМІТАЦІЙНОГО МОДЕЛЮВАННЯ

3.1 Створення імітаційної моделі

Для створення СППР необхідно визначити такі критерії алгоритмів системи керування мобільним роботом як:

– оптимальність отриманого шляху;

– споживання пам’яті;

– швидкість роботи.

Для визначеннях критеріїв в рамках цієї роботи буде використовуватися імітаційне моделювання. Під імітацією розуміють відтворення за допомогою комп’ютерної програми процесу функціонування складної системи в часі. У результаті багатократних прогонів імітаційної моделі дослідник отримує інформацію про властивості реальної системи. Класифікація імітаційних моделей схематично показана на рисунку 3.1.

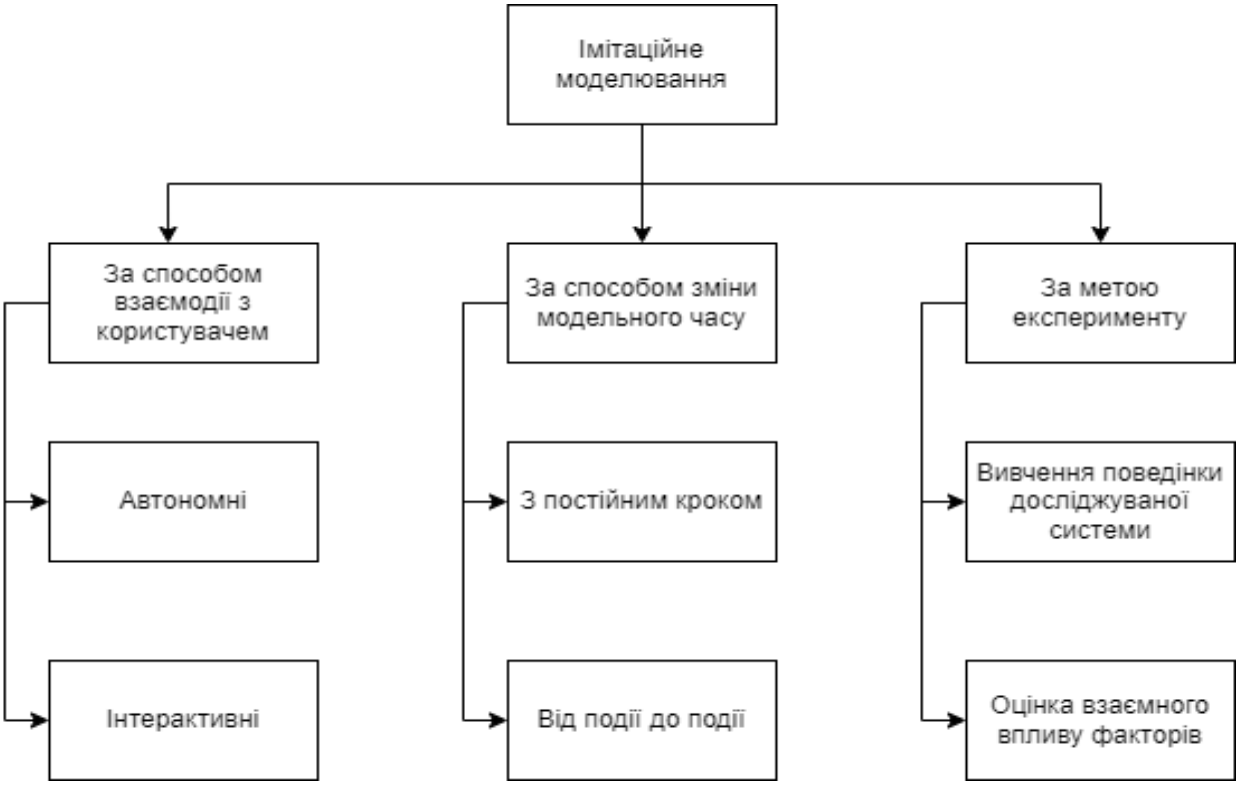


Рисунок 3.1 – Класифікація імітаційних моделей [29]

Виходячи с наведеної класифікації можна охарактеризувати створену імітаційну модель як автономну, зі зміною часу від події до події, метою експерименту є як вивчення поведінки досліджуваної системи так і оцінка взаємного впливу факторів.

Алгоритм імітаційного моделювання системи керування мобільним роботом зображено на рисунку 3.2.

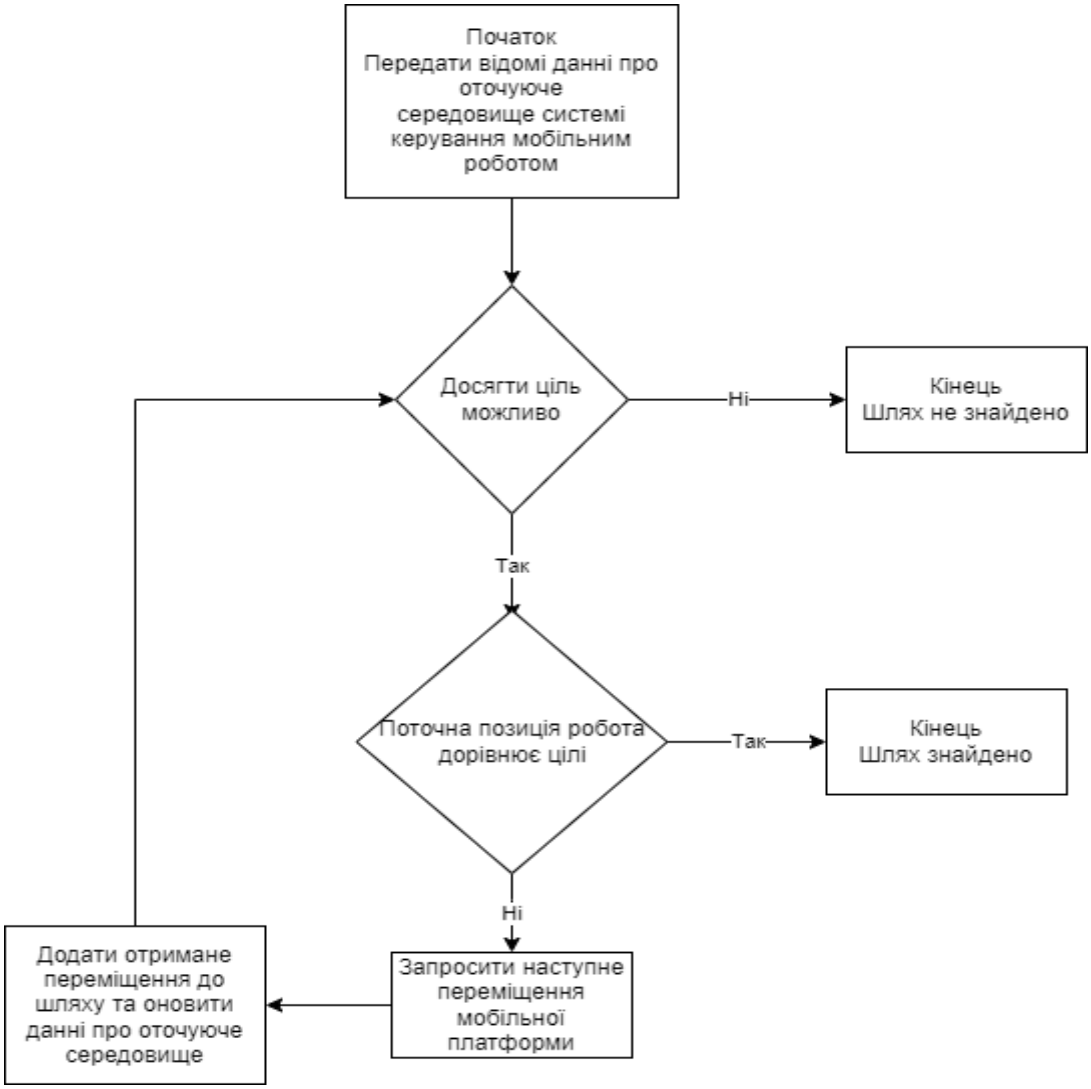


Рисунок 3.2 – Алгоритм імітаційного моделювання системи керування мобільним роботом

Під час аналізу алгоритмів було встановлено, що деякі з цих алгоритмів не адаптовані до умов динамічного оточення, тому, було прийняте рішення

тестувати їх окремо у статичному оточенні. Для проведення експерименту буде використовуватися 4 мапи розмірами 1000 × 1000, 2000 × 2000 та 5000 × 5000 клітин заповнених на 35 % відомими перешкодами, на одній карті для кожного випадку буде проведено по 10 прогонів та до підсумкової таблиці буде внесено найкраще значення цих результатів. Для оцінки алгоритмів, які призначені для роботи у динамічному оточенні додатково будуть проведені експерименти при відомості мапи на 100 %, 75 %, 50 %, 25 %, 0 %. Для всіх алгоритмів у якості базової евристичної функції буде використана евклідова відстань з різним множником.

3.2 Проведення експерименту з алгоритмами в статичному середовище та порівняння результатів

3.2.1 Алгоритм A*

Показники алгоритму при пошуку маршруту на мапі розміром 1000 × 1000 наведені в таблиці 3.1.

Таблиця 3.1 – Показники алгоритму при пошуку маршруту на мапі розміром 1000 × 1000

Множник евристичної функції	Довжина шляху (клітин)	Споживана пам’ять (МБ)	Швидкість роботи (мс)
1	2	3	4
1	1166	19,61	334

Продовження таблиці 3.1

2	1255	4,30	30
3	1259	4,30	29
4	1265	4,30	29
5	1262	4,30	29

Показники алгоритму при пошуку маршруту на мапі розміром 2000 × 2000 наведені в таблиці 3.2.

Таблиця 3.2 – Показники алгоритму при пошуку маршруту на мапі розміром 2000×2000

Множник евристичної функції	Довжина шляху (клітин)	Споживана пам’ять (МБ)	Швидкість роботи (мс)
1	2340	78,07	1651
2	2512	16,59	119
3	2503	16,58	117
4	2507	16,58	121
5	2508	16,58	117

Показники алгоритму при пошуку маршруту на мапі розміром 5000 × 5000 наведені в таблиці 3.3.

Таблиця 3.3 – Показники алгоритму при пошуку маршруту на мапі розміром 5000 × 5000

Множник евристичної функції	Довжина шляху (клітин)	Споживана пам’ять (МБ)	Швидкість роботи (мс)
1	5789	502,20	8939
2	6194	101,65	630
3	6221	101,64	679
4	6252	101,64	652
5	6258	101,64	685

На рисунку 3.3 відображена залежність споживаної пам'яті від площі мапи. Синім кольором позначено значення для евристики з коефіцієнтом 1, помаранчевим - значення для евристики з коефіцієнтом 2.

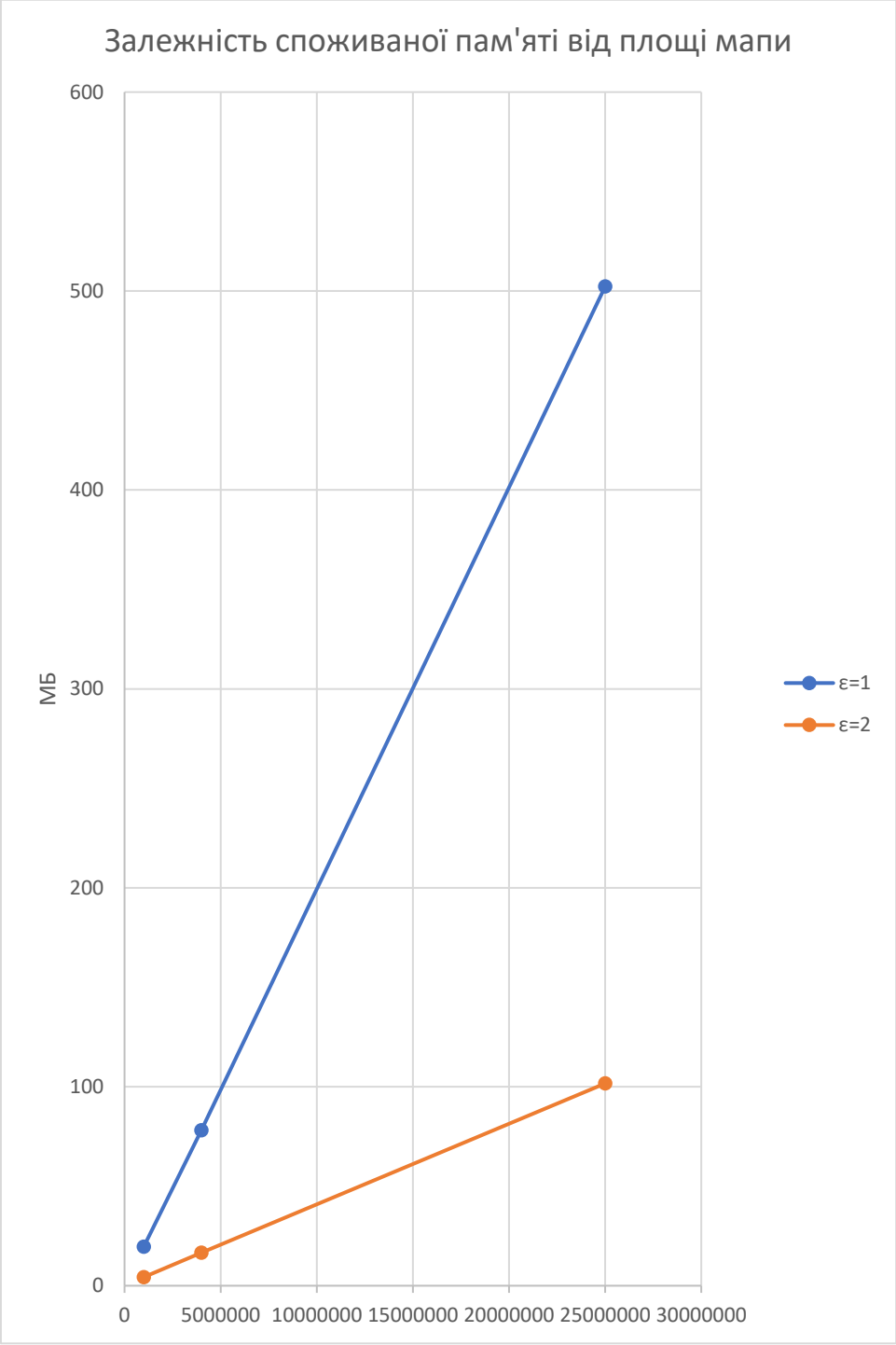


Рисунок 3.3 – Залежність споживаної пам'яті від площі мапи

На рисунку 3.4 відображена залежність швидкості роботи від площі мапи. Синім кольором позначено значення для евристики з коефіцієнтом 1, помаранчевим - значення для евристики з коефіцієнтом 2.

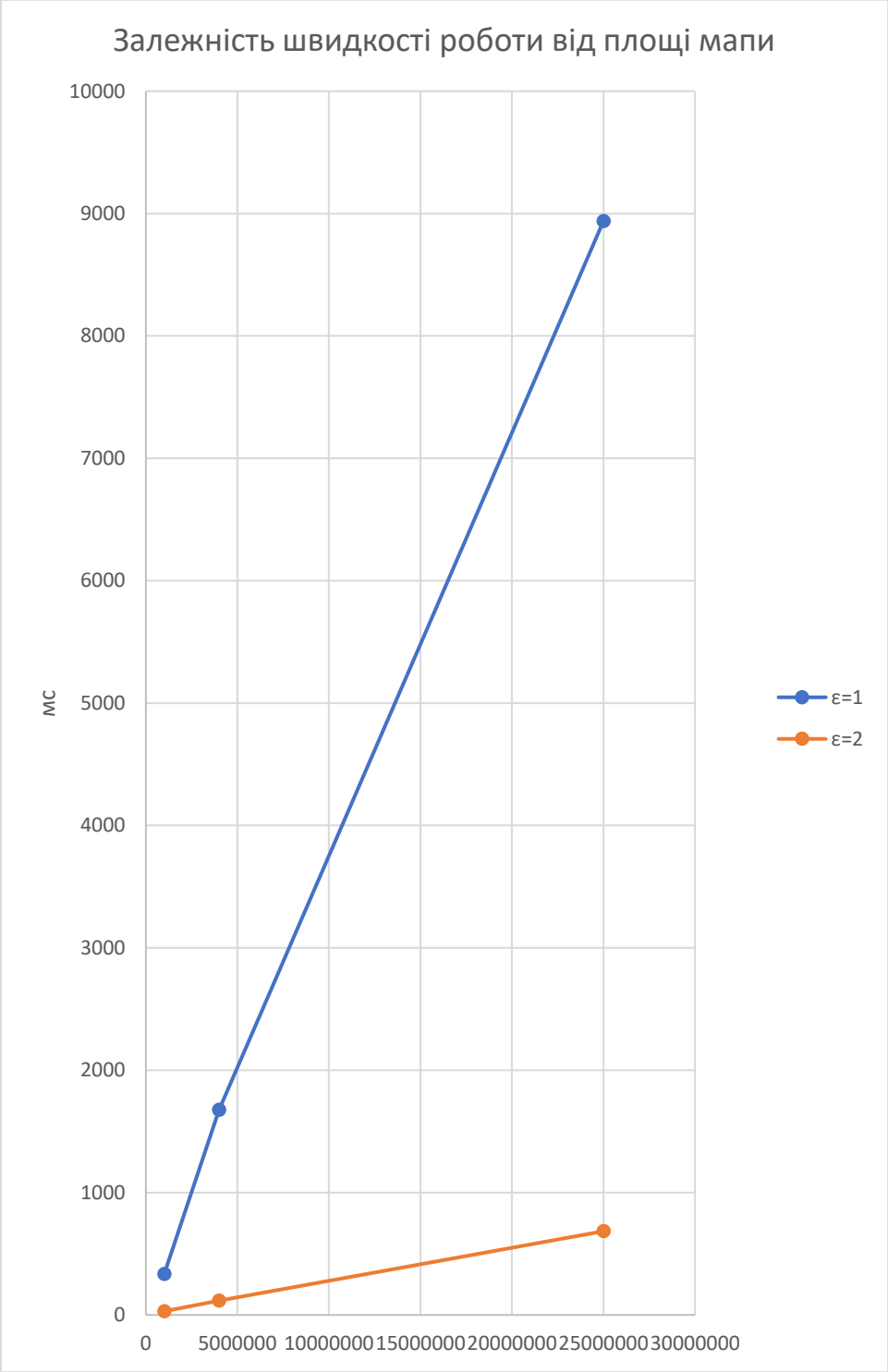


Рисунок 3.4 – Залежність швидкості роботи від площі мапи

За результатами проведених досліджень була сформована таблиця 3.4 погіршень і поліпшень показників при використанні множника евристики 2 у порівнянні з множником евристики 1. Порівнювались тільки множники 2 та 1, тому що, за результатами експериментів було з’ясовано що подальше збільшення евристики майже не впливає на показники.

Таблиця 3.4 – Зміна показників при використанні множника евристики 2 у порівняння з множником евристики 1

Площа мапи	Довжина шляху	Споживана пам’ять	Швидкість роботи
1000000	7.09 %	- 356.20 %	1013.33 %
4000000	6.85 %	- 370.57 %	1287.39 %
25000000	6.54 %	- 394.03 %	1318.89 %

Базуючись на отриманих даних можна зробити наступні висновки: зміна множника евристики з 1 до 2 збільшую довжину знайденого шляху у середньому на 6,83 %, зменшує максимальну кількість споживаної пам’яті у середньому на 373,60 % та прискорює швидкість роботи алгоритму у середньому на 1206,54 %.

3.2.2 Алгоритм Theta*

Показники алгоритму при пошуку маршруту на мапі розміром 1000 × 1000 наведені в таблиці 3.5.

Таблиця 3.5 – Показники алгоритму при пошуку маршруту на мапі розміром 1000 × 1000

Множник евристичної функції	Довжина шляху (клітин)	Споживана пам’ять (МБ)	Швидкість роботи (мс)
1	1117	14,42	965
2	1177	3,58	226
3	1179	3,58	225
4	1180	3,58	226
5	1185	3,58	226

Показники алгоритму при пошуку маршруту на мапі розміром 2000 × 2000 наведені в таблиці 3.6.

Таблиця 3.6 – Показники алгоритму при пошуку маршруту на мапі розміром 2000 × 2000

Множник евристичної функції	Довжина шляху (клітин)	Споживана пам’ять (МБ)	Швидкість роботи (мс)
1	2268	58,91	4053
2	2392	13,80	985
3	2399	13,80	982
4	2405	13,80	980
5	2412	13,79	981

Показники алгоритму при пошуку маршруту на мапі розміром 5000 × 5000 наведені в таблиці 3.7.

Таблиця 3.7 – Показники алгоритму при пошуку маршруту на мапі розміром 5000 × 5000

Множник евристичної функції	Довжина шляху (клітин)	Споживана пам’ять (МБ)	Швидкість роботи (мс)
1	5586	375,31	21775
2	5902	85,18	5406
3	5958	85,09	5405
4	5989	85,08	5402
5	5605	85,07	5403

На рисунку 3.5 відображена залежність споживаної пам'яті від площі мапи. Синім кольором позначено значення для евристики з коефіцієнтом 1, помаранчевим - значення для евристики з коефіцієнтом 2.

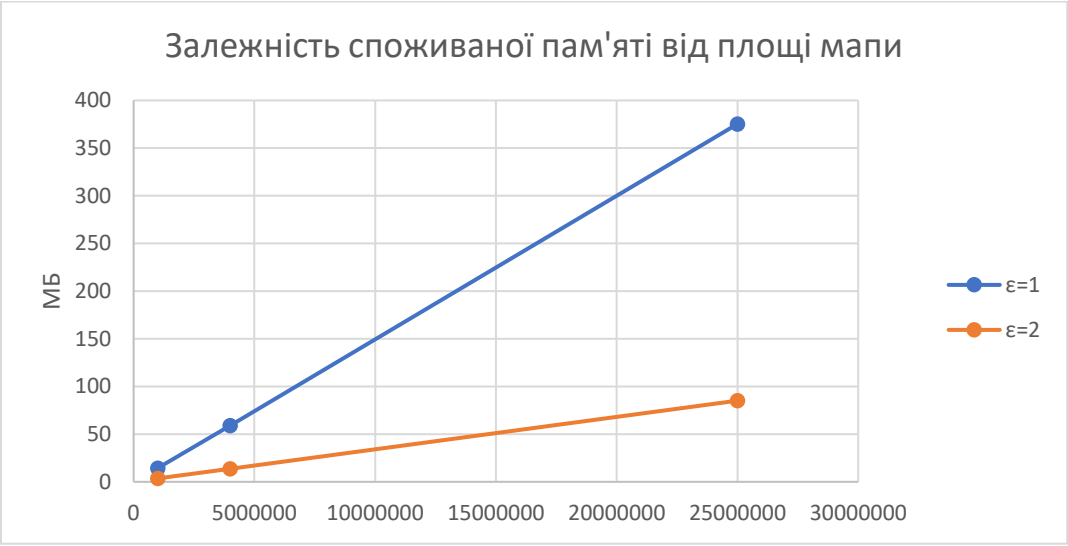


Рисунок 3.5 – Залежність споживаної пам'яті від площі мапи

На рисунку 3.6 відображена залежність швидкості роботи від площі мапи. Синім кольором позначено значення для евристики з коефіцієнтом 1, помаранчевим - значення для евристики з коефіцієнтом 2.

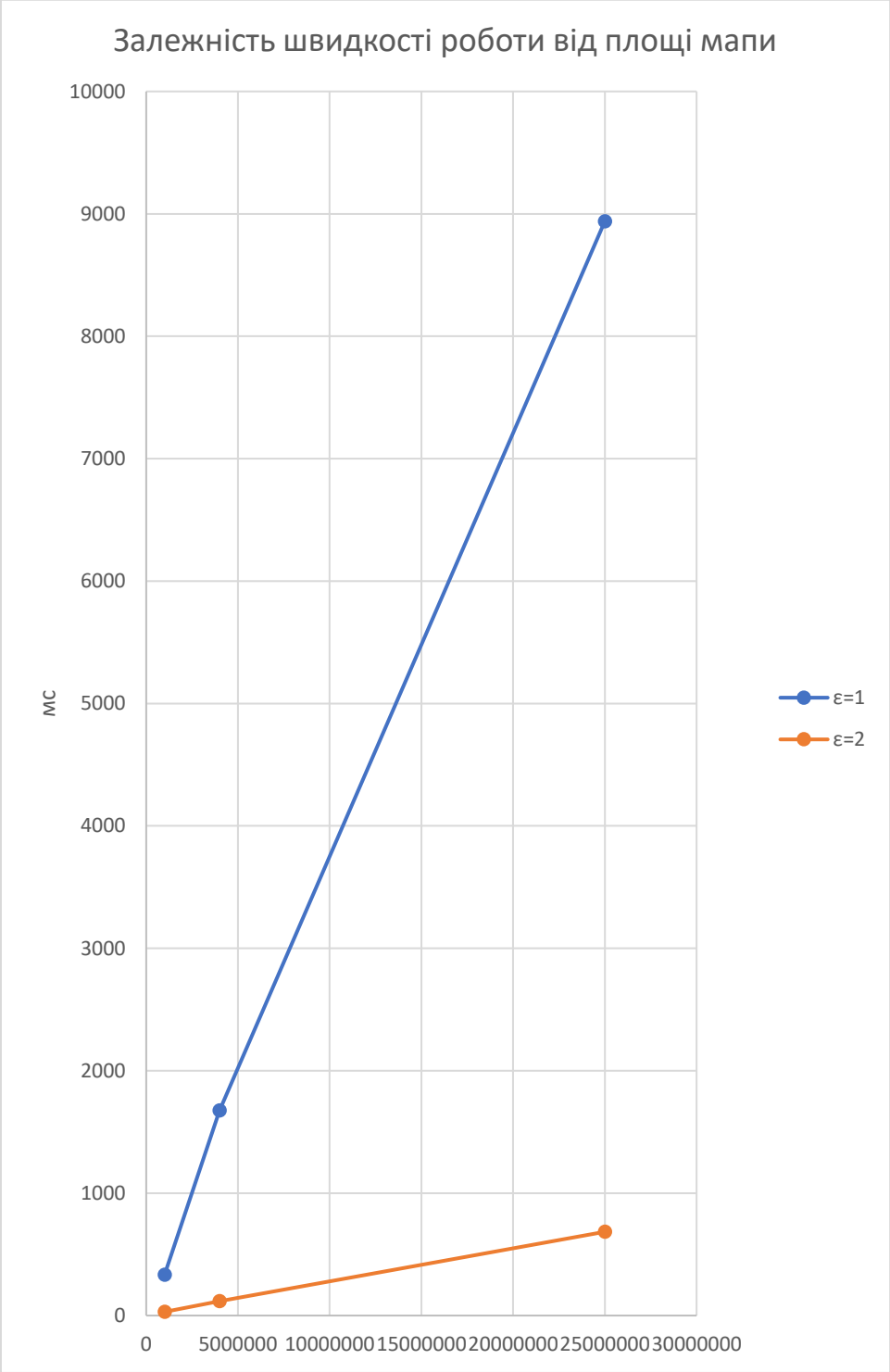


Рисунок 3.6 – Залежність споживаної пам'яті від площі мапи

За результатами проведених досліджень була сформована таблиця 3.8 погіршень і поліпшень показників при використанні множника евристики 2 у порівнянні з множником евристики 1. Порівнювались тільки множники 2 та 1, тому що, за результатами експериментів було з’ясовано що подальше збільшення евристики майже не впливає на показники.

Таблиця 3.8 – Зміна показників при використанні множника евристики 2 у порівняння з множником евристики 1

Площа мапи	Довжина шляху	Споживана пам’ять	Швидкість роботи
1000000	5,09 %	- 302,39 %	326,99 %
4000000	5,18 %	- 326,99 %	311,47 %
25000000	5,35 %	- 340,92 %	302,79 %

Базуючись на отриманих даних можна зробити наступні висновки: показники при множник евристики більше 2 майже не відрізняються від показників при множнику евристики 2, зміна множника евристики з 1 до 2 збільшую довжину знайденого шляху у середньому на 5,21 %, зменшує максимальну кількість споживаної пам’яті у середньому на 323,43 % та прискорює швидкість роботи алгоритму у середньому на 313,75 %.

3.2.3 Порівняння показників A* та Theta*

У ході проведення експериментів було з’ясовано, що множник евристичної функції більше ніж 2 майже не пливає на результат, відповідно будуть розглянуті та порівнянні показники алгоритмів A* та Theta* з множниками 1 та

2. У таблиці 3.9 відображено співвідношення характеристик алгоритмів A* порівняно з Theta* для мапи розміром 1000 ×1000.

Таблиця 3.9 – Співвідношення характеристик алгоритмів A* порівняно з Theta* для мапи розміром 1000 × 1000

Алгоритм	Довжина шляху	Споживана пам’ять	Швидкість роботи
Theta* ε = 1	1,00	4,02	32,17
Theta* ε = 2	1,05	1,00	7,53
A* ε = 1	1,04	5,47	11,13
A* ε = 2	1,12	1,20	1,00

У таблиці 3.10 відображено співвідношення характеристик алгоритмів A* порівняно з Theta* для мапи розміром 2000 × 2000.

Таблиця 3.10 – Співвідношення характеристик алгоритмів A* порівняно з Theta* для мапи розміром 2000 × 2000

Алгоритм	Довжина шляху	Споживана пам’ять	Швидкість роботи
Theta* ε = 1	1,00	4,27	34,06
Theta* ε = 2	1,05	1,00	8,28
A* ε = 1	1,03	5,66	13,87
A* ε = 2	1,11	1,20	1,00

У таблиці 3.11 відображено співвідношення характеристик алгоритмів A* порівняно з Theta* для мапи розміром 5000 × 5000.

Таблиця 3.11 – Співвідношення характеристик алгоритмів A* порівняно з Theta* для мапи розміром 5000 × 5000

Алгоритм	Довжина шляху	Споживана пам’ять	Швидкість роботи
Theta* ε = 1	1,00	4,41	34,56
Theta* ε = 2	1,06	1,00	8,58
A* ε = 1	1,04	5,90	14,19
A* ε = 2	1,11	1,19	1,00

У таблиці 3.12 відображено середнє співвідношення характеристик алгоритмів A* порівняно з Theta* для всіх розглянутих мап.

Таблиця 3.12 – Середнє співвідношення характеристик алгоритмів A* порівняно з Theta* для всіх розглянутих мап

Алгоритм	Довжина шляху	Споживана пам’ять	Швидкість роботи
Theta* ε = 1	1,00	4,23	33,60
Theta* ε = 2	1,05	1,00	8,13
A* ε = 1	1,04	5,68	13,07
A* ε = 2	1,11	1,20	1,00

На рисунку 3.7 відображено середнє співвідношення характеристик алгоритмів A* порівняно з Theta* для всіх розглянутих мап.

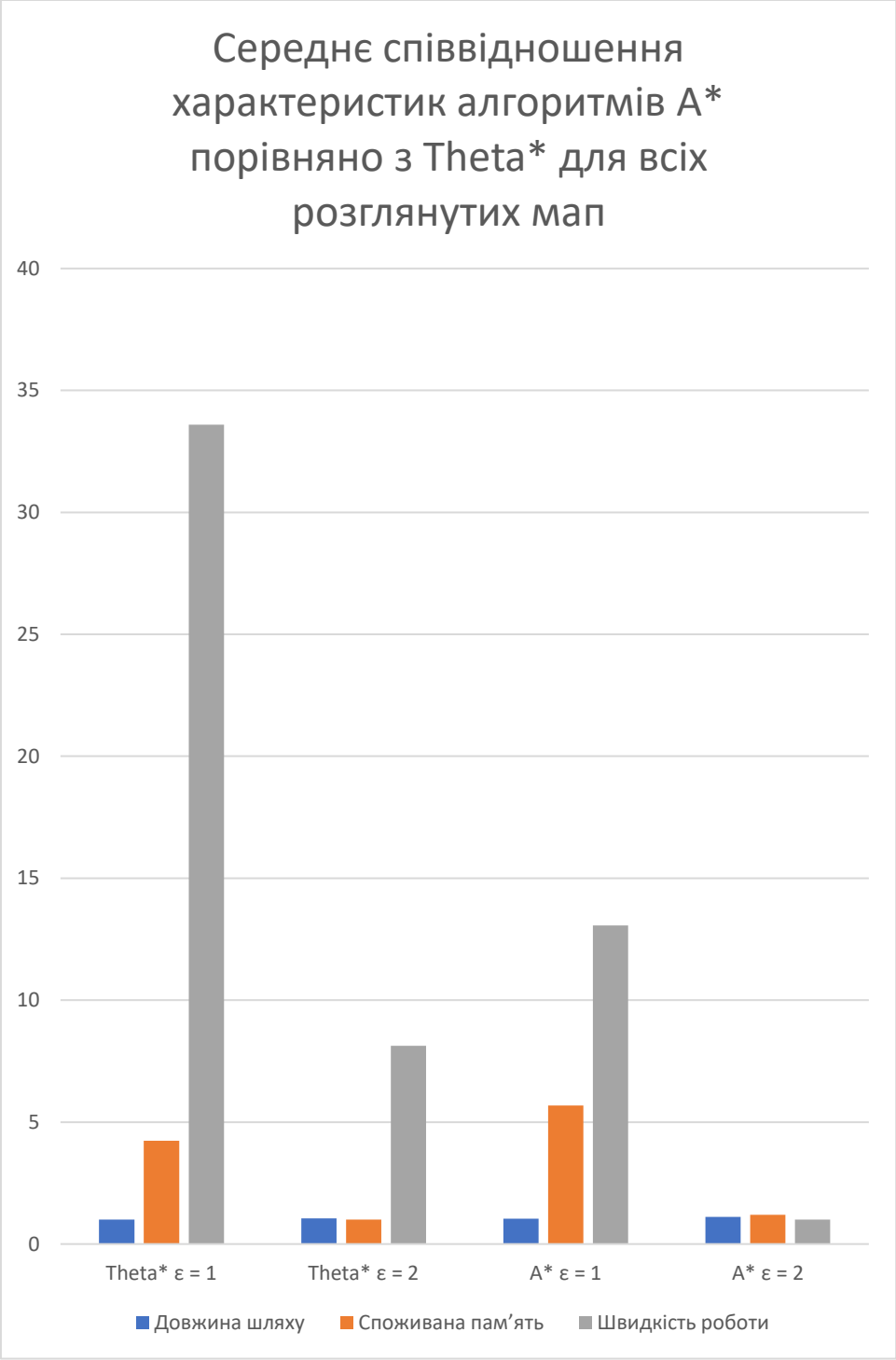


Рисунок 3.9 – Співвідношення характеристик алгоритмів A* порівняно з Theta* для всіх розглянутих мап

Базуючись на отриманих експериментальних даних можна зробити наступні висновки:

– алгоритм Theta* незалежно від розміру мапи та значення множника евристичної функції знаходить найбільш коротший шлях та споживає найменшу кількість пам’яті, але, при цьому, має найбільш довгий час виконання;

– для обох алгоритмів підвищення коефіцієнту евристики призводить до того, що довжина отриманого шляху збільшується, але, при цьому зменшується максимальне споживання пам’яті та скорочується час роботи алгоритму;

– показники алгоритмів A* та Theta* при різних значеннях множника евристичної функції дозволяє ОПР обрати відповідний алгоритм в залежності від того, які показники та на скільки йому більш важливі ніж інші.

На рисунку 3.10 відображено візуалізацію знайдених алгоритмами A* та Theta* з множниками евристики 1 та 2 шляхів для мапи розміром 1000 × 1000.

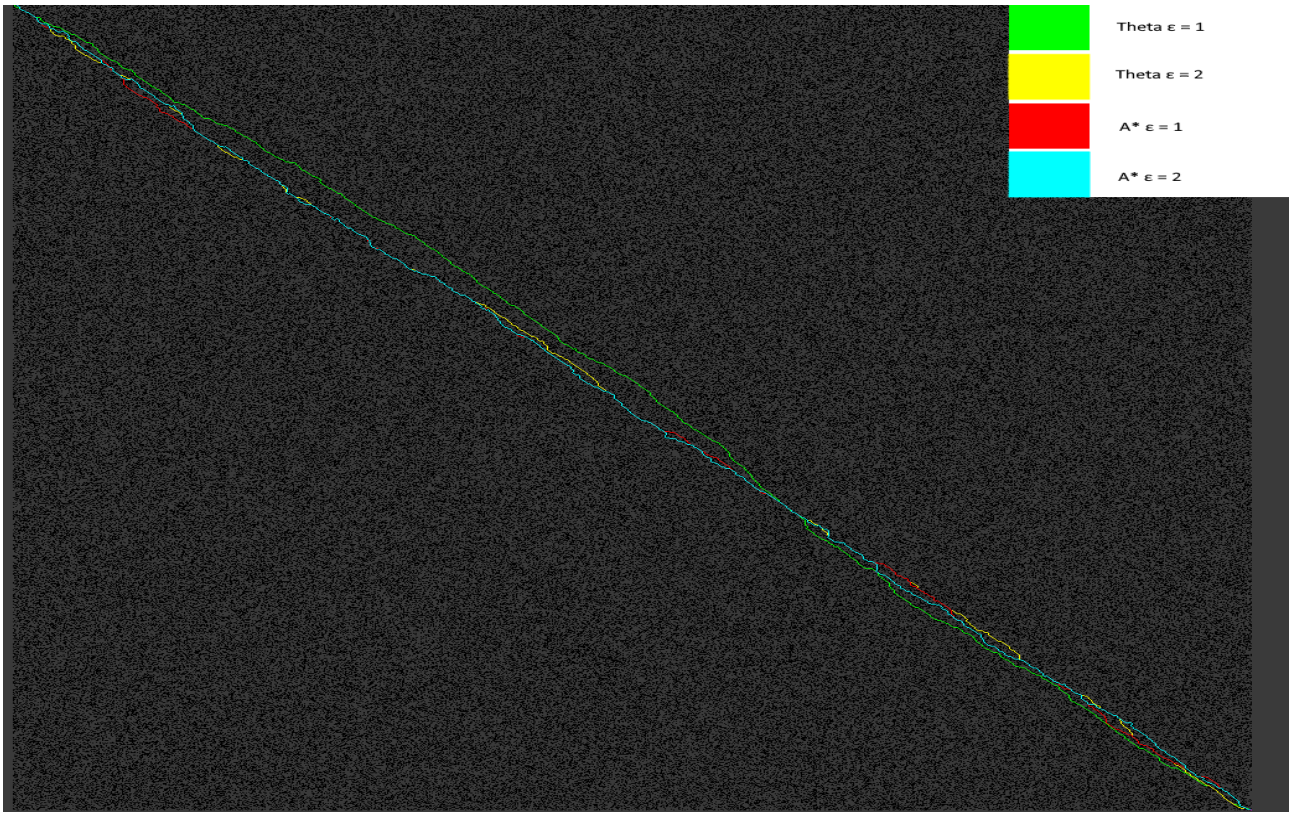


Рисунок 3.10 – Візуалізація знайдених алгоритмами A* та Theta* з множниками евристики 1 та 2 шляхів

При розгляді першого сектору мапи можна побачити як алгоритми по різному уникають перешкод це зображено на рисунку 3.11.

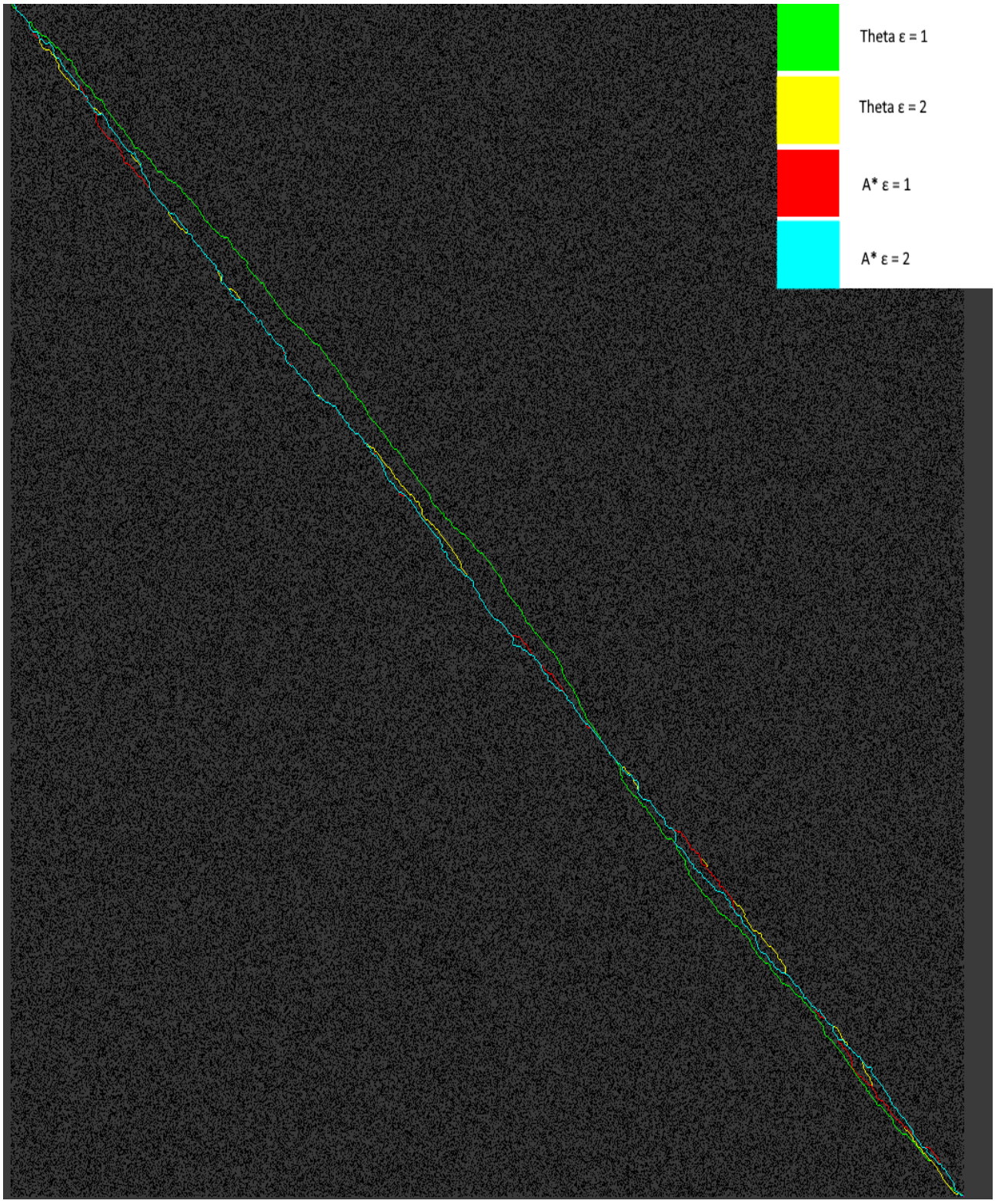


Рисунок 3.11 – Перший сектор мапи

При розгляді другого сектору мапи можна побачити як алгоритми по різному уникають перешкод це зображено на рисунку 3.12.

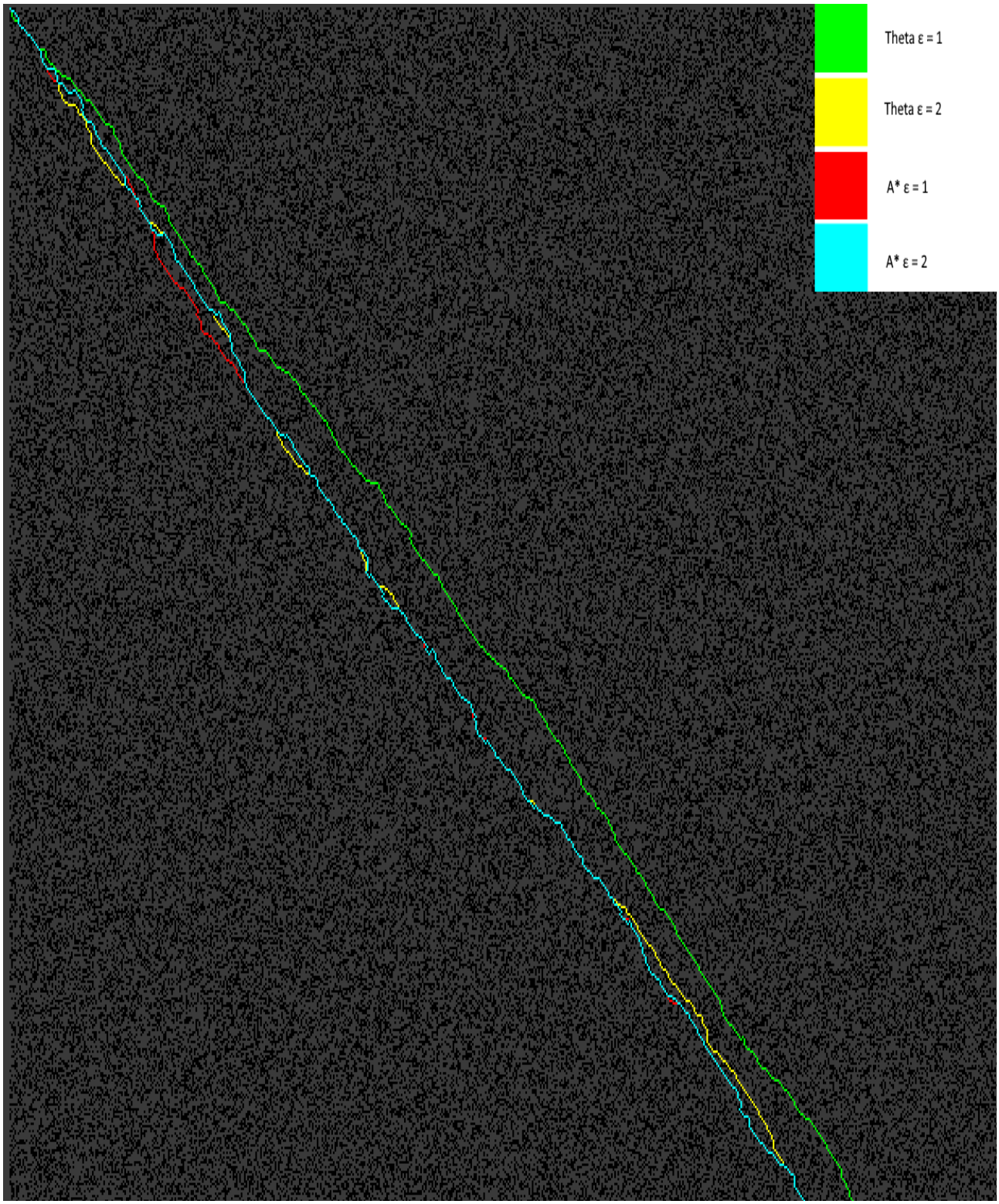


Рисунок 3.12 – Другий сектор мапи

При розгляді третього сектору мапи можна побачити як алгоритми по різному уникають перешкод це зображено на рисунку 3.13.

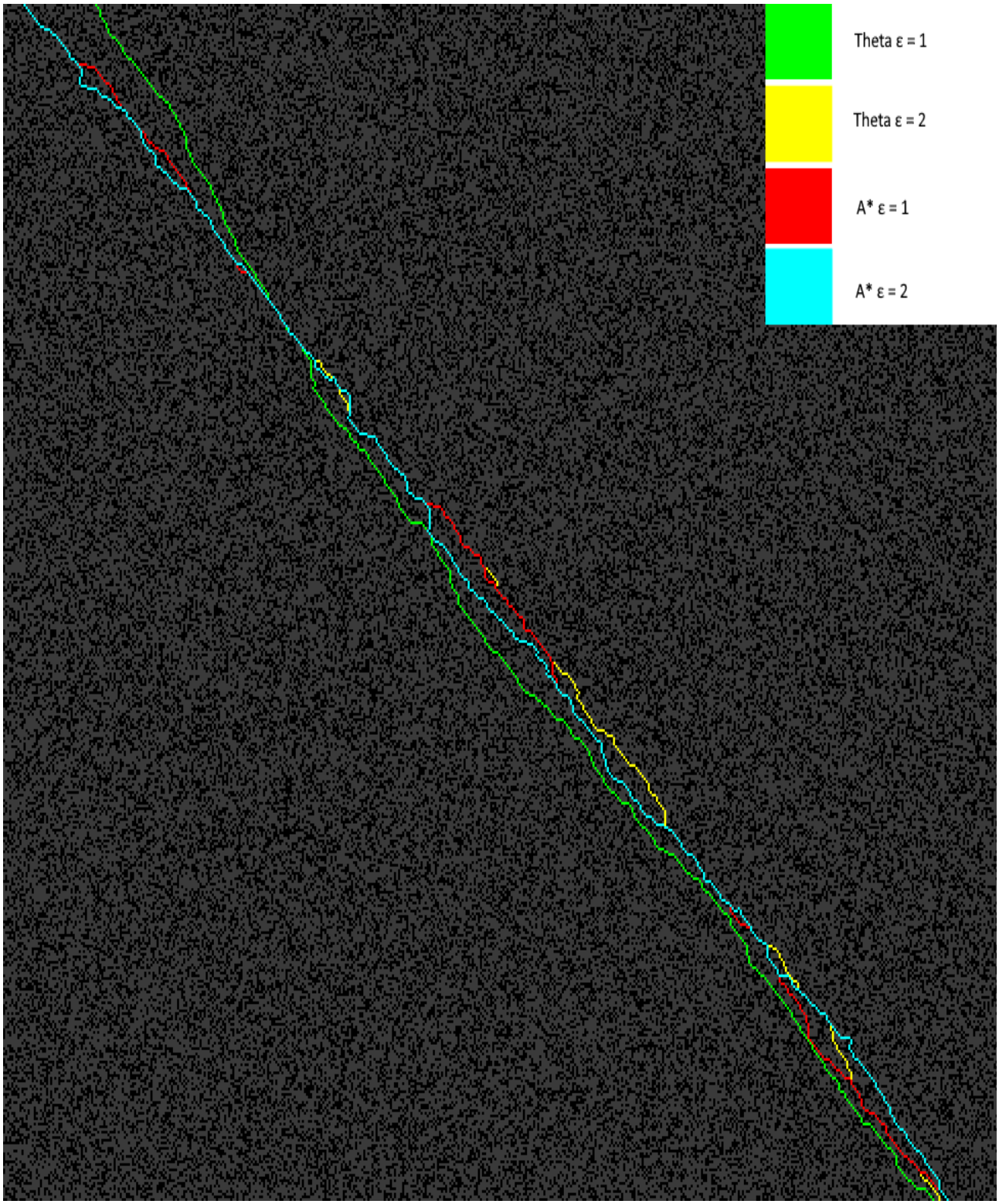


Рисунок 3.13 – Третій сектор мапи

При розгляді четвертого сектору мапи можна побачити як алгоритми по різному уникають перешкод це зображено на рисунку 3.14.

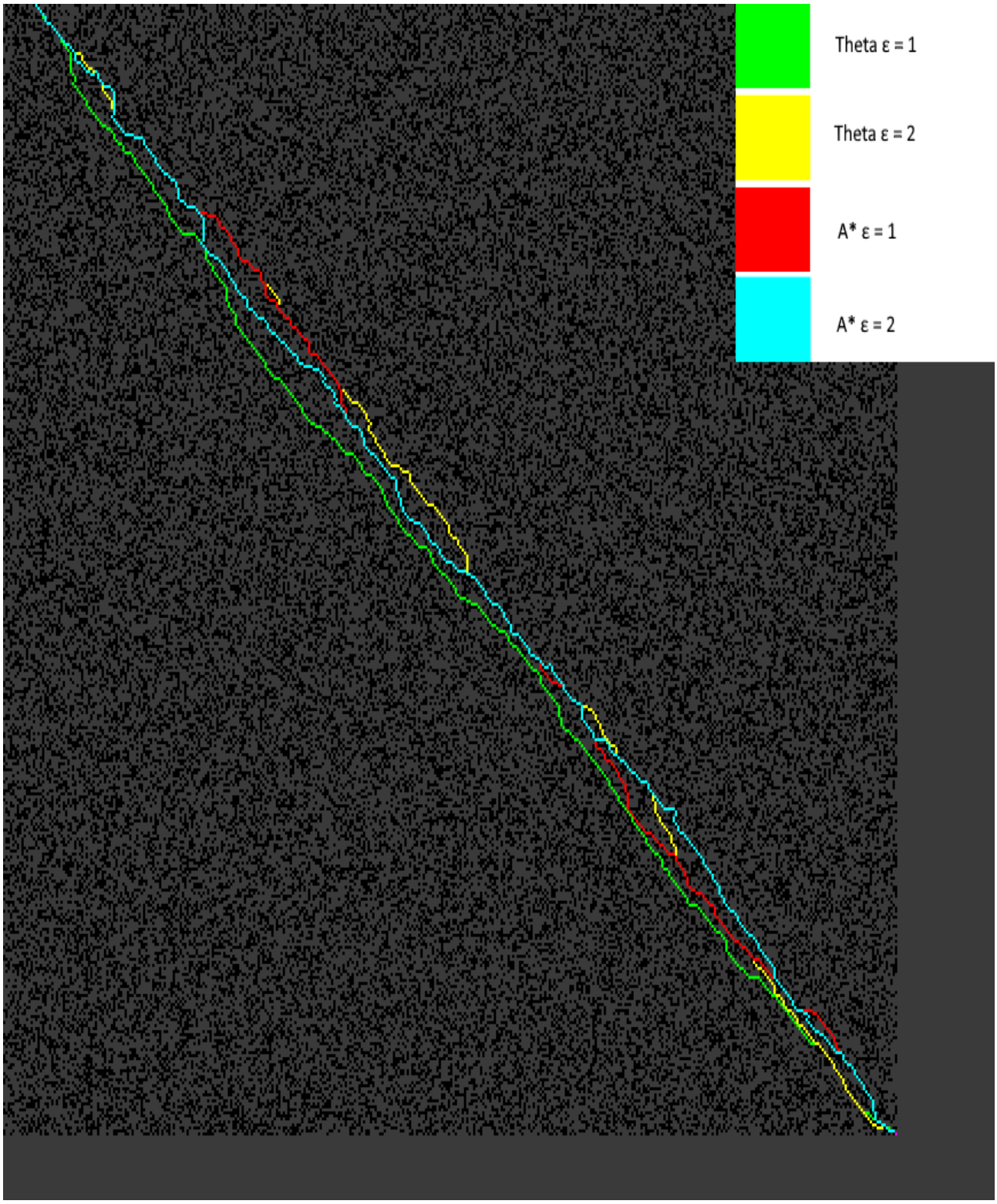


Рисунок 3.14 – Четвертий сектор мапи

При розгляді п’ятого сектору мапи можна побачити як алгоритми по різному уникають перешкод це зображено на рисунку 3.15.

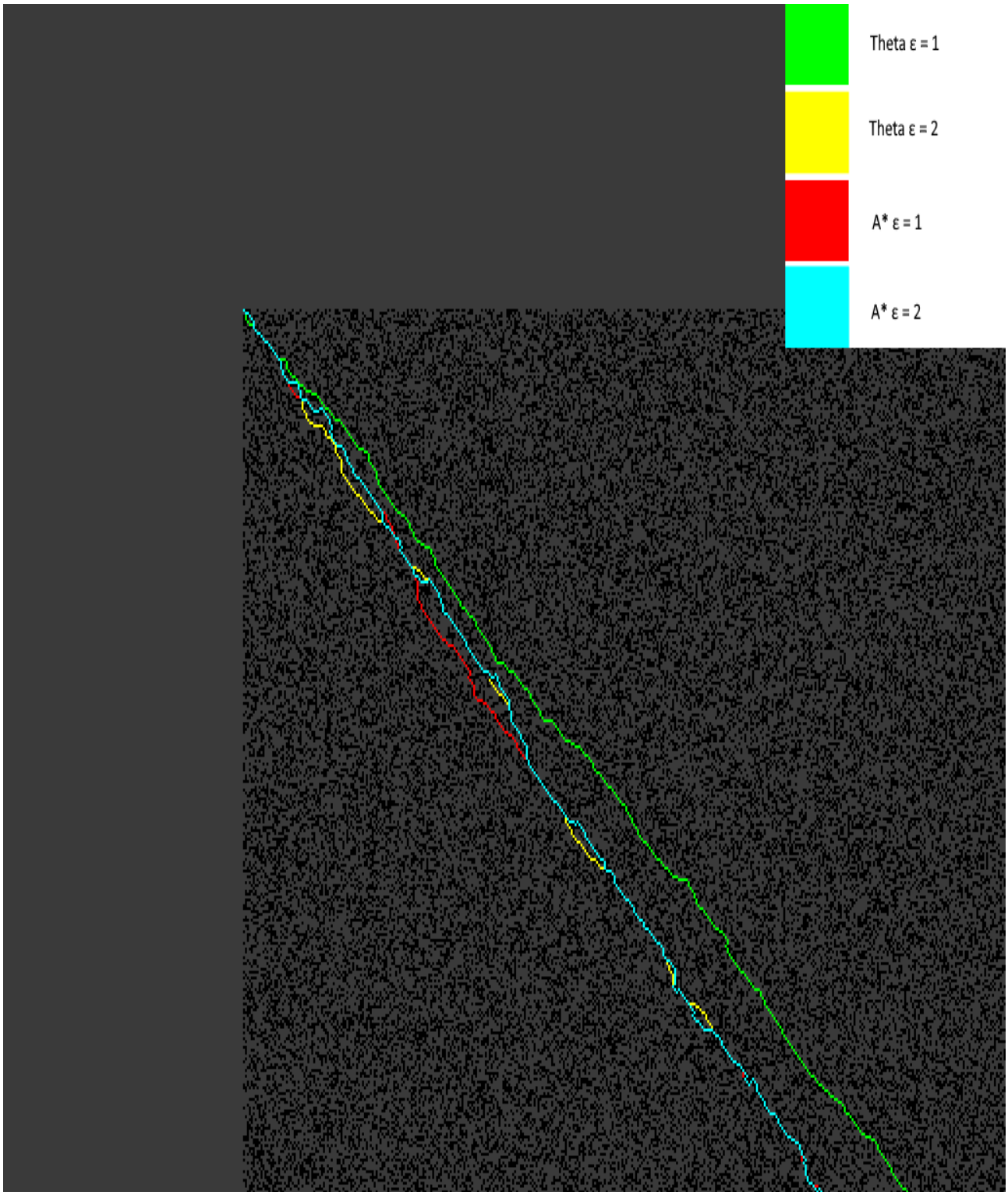


Рисунок 3.15 – П’ятий сектор мапи

3.3 Проведення експерименту з алгоритмами в динамічному середовище та порівняння результатів

3.3.1 Алгоритм D* Lite

Базуючись на експериментальних даних алгоритмів A* та Theta* з алгоритмом D* Lite було проведено попередні дослідження, у ході яких з’ясувалося, що множник коефіцієнта евристики більше 2 також майже не впливає на показники алгоритму, тому, ці данні, не будуть включені до таблиць.

Показники алгоритму при пошуку маршруту на мапі розміром 1000 × 1000 наведені в таблиці 3.13.

Таблиця 3.13 – Показники алгоритму при пошуку маршруту на мапі розміром 1000 × 1000

Відсоток невідомих перешкод	Множник евристичної функції	Довжина шляху (клітин)	Споживана пам’ять (МБ)	Швидкість роботи (мс)
1	2	3	4	5
0	1	1166	20,83	1756
	2	1255	20,62	63
25	1	1209	20,49	1602
	2	1290	20,07	56

Продовження таблиці 3.13

50	1	1254	20,13	1480
	2	1339	19,53	50
75	1	1300	19,80	1356
	2	1385	19,01	44
100	1	1348	19,46	1332
	2	1421	18,60	43

Показники алгоритму при пошуку маршруту на мапі розміром 2000 × 2000 наведені в таблиці 3.14.

Таблиця 3.14 – Показники алгоритму при пошуку маршруту на мапі розміром 2000 × 2000

Відсоток невідомих перешкод	Множник евристичної функції	Довжина шляху (клітин)	Споживана пам’ять (МБ)	Швидкість роботи (мс)
0	1	2330	83,52	6446
	2	2512	82,68	214
25	1	2419	82,06	5913
	2	2598	80,30	196
50	1	2511	80,63	5471
	2	2681	78,62	180
75	1	2607	79,22	5008
	2	2767	77,28	162
100	1	2706	77,84	4895
	2	2855	75,96	159

Показники алгоритму при пошуку маршруту на мапі розміром 5000 × 5000 наведені в таблиці 3.15.

Таблиця 3.15 – Показники алгоритму при пошуку маршруту на мапі розміром 5000 × 5000

Відсоток невідомих перешкод	Множник евристичної функції	Довжина шляху (клітин)	Споживана пам’ять (МБ)	Швидкість роботи (мс)
0	1	5789	510,42	39948
	2	6194	505,22	1118
25	1	6021	499,93	36125
	2	6434	495,37	1036
50	1	6262	489,64	33062
	2	6680	482,65	946
75	1	6513	479,57	30262
	2	6935	475,08	863
100	1	6774	469,71	29787
	2	7200	467,64	859

Середнє співвідношення характеристик алгоритму наведені в таблиці 3.16.

Таблиця 3.16 – Середнє співвідношення характеристик алгоритму

Відсоток невідомих перешкод	Множник евристичної функції	Довжина шляху	Споживана пам’ять	Швидкість роботи
0	1	1,00	1,10	42,63
	2	1,07	1,09	1,37
25	1	1,04	1,08	38,83
	2	1,11	1,07	1,25
50	1	1,08	1,06	35,77
	2	1,15	1,04	1,13
75	1	1,12	1,04	32,75
	2	1,19	1,02	1,02
100	1	1,16	1,03	32,15
	2	1,23	1,00	1,00

На рисунку 3.16 відображено середнє співвідношення характеристик алгоритму.

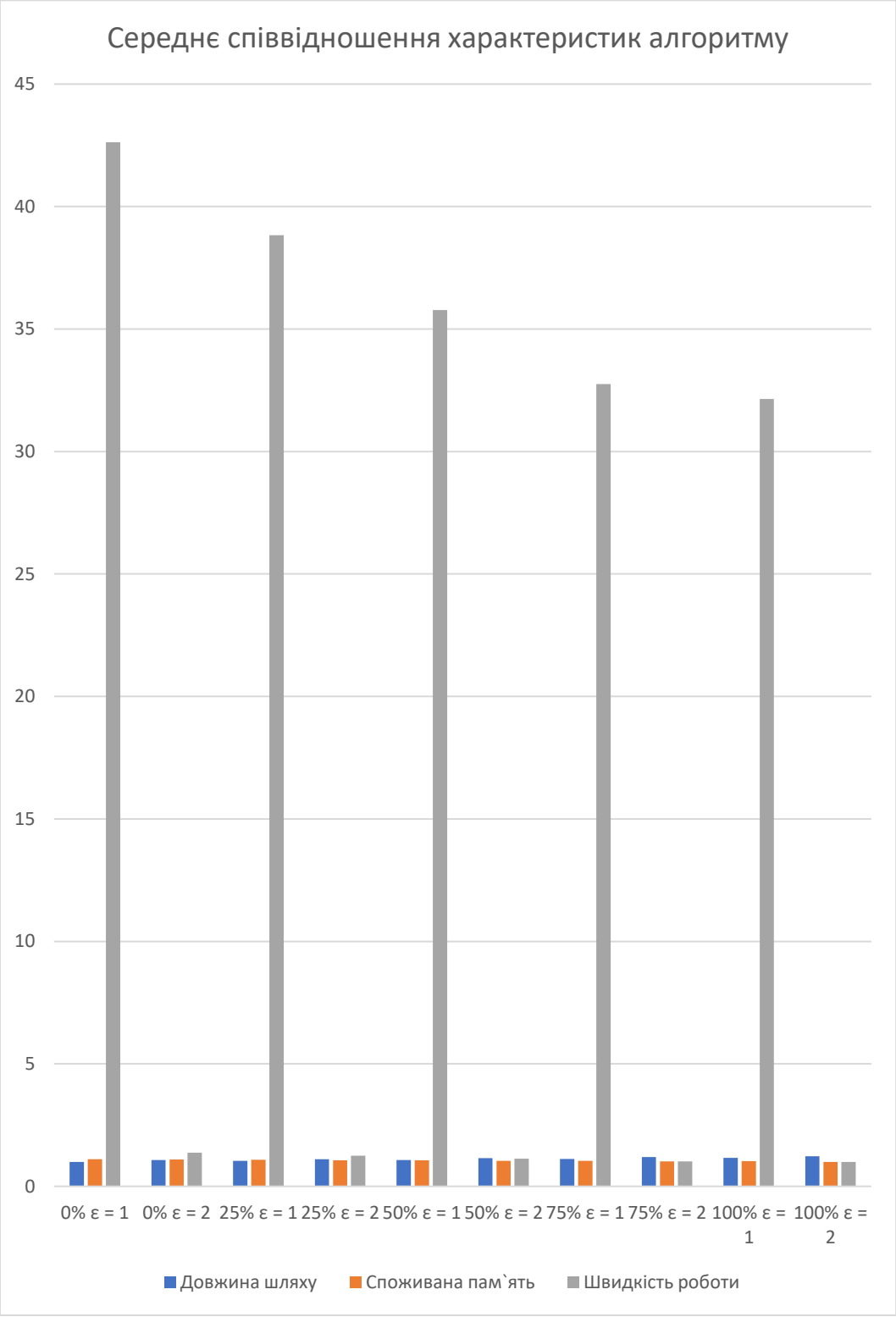


Рисунок 3.16 – Середнє співвідношення характеристик алгоритму

За результатами проведених досліджень була сформована таблиця 3.17 погіршень і поліпшень показників при використанні множника евристики 2 у порівнянні з множником евристики 1.

Таблиця 3.17 – Зміна показників при використанні множника евристики 2 у порівняння з множником евристики 1

Площа мапи	Довжина шляху	Споживана пам’ять	Швидкість роботи
1000000	5,02 %	- 2,96 %	2857,50 %
4000000	5,28%	- 2,15 %	2947,68 %
25000000	5,21 %	- 0,96 %	3405,86 %

Базуючись на отриманих даних можна зробити наступні висновки: зміна множника евристики з 1 до 2 збільшую довжину знайденого шляху у середньому на 5,17 %, зменшує максимальну кількість споживаної пам’яті у середньому на 2,02 % та прискорює швидкість роботи алгоритму у середньому на 3070,35 %.

3.3.2 Field D*

Базуючись на експериментальних даних алгоритмів A* та Theta* та D* Lite з алгоритмом Field D* було проведено попередні дослідження, у ході яких з’ясувалося, що множник коефіцієнта евристики більше 2 також майже не впливає на показники алгоритму, тому, ці данні, не будуть включені до таблиць.

Показники алгоритму при пошуку маршруту на мапі розміром 1000 × 1000 наведені в таблиці 3.18.

Таблиця 3.18 – Показники алгоритму при пошуку маршруту на мапі розміром 1000 × 1000

Відсоток невідомих перешкод	Множник евристичної функції	Довжина шляху (клітин)	Споживана пам’ять (МБ)	Швидкість роботи (мс)
0	1	1133	20,64	4269
	2	1222	20,48	82
25	1	1175	20,29	3949
	2	1268	19,93	73
50	1	1218	19,95	3645
	2	1316	19,40	68
75	1	1263	19,62	3296
	2	1361	18,88	60
100	1	1310	19,29	3238
	2	1396	18,37	58

Показники алгоритму при пошуку маршруту на мапі розміром 2000 × 2000 наведені в таблиці 3.19.

Таблиця 3.19 – Показники алгоритму при пошуку маршруту на мапі розміром 2000 × 2000

Відсоток невідомих перешкод	Множник евристичної функції	Довжина шляху (клітин)	Споживана пам’ять (МБ)	Швидкість роботи (мс)
0	1	2272	82,94	15671
	2	2450	80,91	280
25	1	2358	81,50	14579
	2	2536	79,52	256
50	1	2448	80,07	13379
	2	2617	78,17	236
75	1	2541	78,68	12342
	2	2701	76,84	212
100	1	2638	77,30	12069
	2	2787	75,53	208

Показники алгоритму при пошуку маршруту на мапі розміром 5000 × 5000 наведені в таблиці 3.20.

Таблиця 3.20 – Показники алгоритму при пошуку маршруту на мапі розміром 5000 × 5000

Відсоток невідомих перешкод	Множник евристичної функції	Довжина шляху (клітин)	Споживана пам’ять (МБ)	Швидкість роботи (мс)
0	1	5622	507,99	98497
	2	6088	499,41	1463
25	1	5838	497,54	89100
	2	6321	491,58	1355
50	1	6072	487,31	81519
	2	6562	478,96	1238
75	1	6315	477,29	74246
	2	6812	471,45	1130
100	1	6568	467,47	73435
	2	7072	464,06	1124

Середнє співвідношення характеристик алгоритму наведені в таблиці 3.21.

Таблиця 3.21 – Середнє співвідношення характеристик алгоритму.

Відсоток невідомих перешкод	Множник евристичної функції	Довжина шляху	Споживана пам’ять	Швидкість роботи
0	1	1,00	1,11	78,86
	2	1,08	1,09	1,35
25	1	1,04	1,09	72,48
	2	1,12	1,07	1,23
50	1	1,08	1,07	66,56
	2	1,16	1,04	1,14
75	1	1,12	1,05	60,74
	2	1,20	1,02	1,02
100	1	1,16	1,03	59,73
	2	1,24	1,00	1,00

На рисунку 3.17 відображено середнє співвідношення характеристик алгоритму.

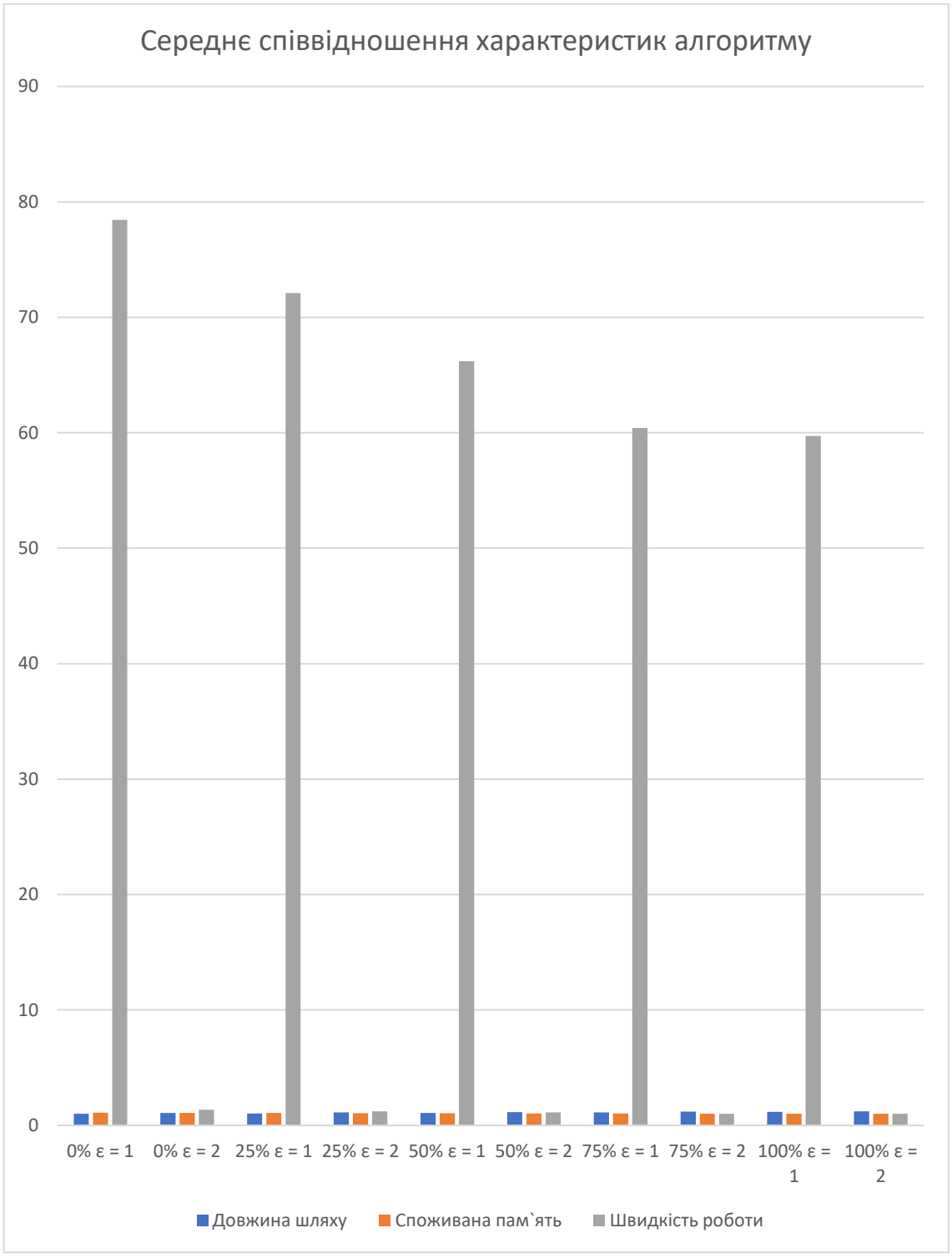


Рисунок 3.17 – Середнє співвідношення характеристик алгоритму

За результатами проведених досліджень була сформована таблиця 3.22 погіршень і поліпшень показників при використанні множника евристики 2 у порівнянні з множником евристики 1.

Таблиця 3.22 – Зміна показників при використанні множника евристики 2 у порівняння з множником евристики 1

Площа мапи	Довжина шляху	Споживана пам’ять	Швидкість роботи
1000000	5,90 %	- 2,88 %	5310,41 %
4000000	5,34%	- 2,43 %	5616,98 %
25000000	6,20%	- 1,33 %	6499,34 %

Базуючись на отриманих даних можна зробити наступні висновки: зміна множника евристики з 1 до 2 збільшую довжину знайденого шляху у середньому на 5,81 %, зменшує максимальну кількість споживаної пам’яті у середньому на 2,22 % та прискорює швидкість роботи алгоритму у середньому на 5808,91 %.

3.3.3 Порівняння показників D* Lite та Field D*

У таблиці 3.23 відображено середнє співвідношення характеристик алгоритмів D* Lite порівняно з Field D* мапи 1000 × 1000 та для всіх розглянутих відсотків невідомих перешкод.

Таблиця 3.23 – Середнє співвідношення характеристик алгоритмів D* Lite порівняно з Field D* мапи 1000 × 1000 та для всіх розглянутих відсотків невідомих перешкод

Алгоритм	Довжина шляху	Споживана пам’ять	Швидкість роботи
D* Lite ε = 1	1,01	1,04	29,40
D* Lite ε = 2	1,08	1,01	1,00
Field D* ε = 1	1,00	1,03	71,86
Field D* ε = 2	1,06	1,00	1,33

У таблиці 3.24 відображено середнє співвідношення характеристик алгоритмів D* Lite порівняно з Field D* мапи 2000 × 2000 та для всіх розглянутих відсотків невідомих перешкод.

Таблиця 3.24 – Середнє співвідношення характеристик алгоритмів D* Lite порівняно з Field D* мапи 2000 × 2000 та для всіх розглянутих відсотків невідомих перешкод

Алгоритм	Довжина шляху	Споживана пам’ять	Швидкість роботи
D* Lite ε = 1	1,03	1,03	30,44
D* Lite ε = 2	1,09	1,01	1,00
Field D* ε = 1	1,00	1,02	74,69
Field D* ε = 2	1,07	1,00	1,31

У таблиці 3.25 відображено середнє співвідношення характеристик алгоритмів D* Lite порівняно з Field D* мапи 5000 × 5000 та для всіх розглянутих відсотків невідомих перешкод.

Таблиця 3.25 – Середнє співвідношення характеристик алгоритмів D* Lite порівняно з Field D* мапи 5000 × 5000 та для всіх розглянутих відсотків невідомих перешкод

Алгоритм	Довжина шляху	Споживана пам’ять	Швидкість роботи
D* Lite ε = 1	1,03	1,02	35,09
D* Lite ε = 2	1,10	1,01	1,00
Field D* ε = 1	1,00	1,01	86,44
Field D* ε = 2	1,08	1,00	1,31

У таблиці 3.26 відображено середнє співвідношення характеристик алгоритмів D* Lite порівняно з Field D* для всіх розглянутих мап та для всіх розглянутих відсотків невідомих перешкод.

Таблиця 3.26 – Середнє співвідношення характеристик алгоритмів D* Lite порівняно з Field D* для всіх розглянутих мап та для всіх розглянутих відсотків невідомих перешкод

Алгоритм	Довжина шляху	Споживана пам’ять	Швидкість роботи
D* Lite ε = 1	1,02	1,03	31,64
D* Lite ε = 2	1,09	1,01	1,00
Field D* ε = 1	1,00	1,02	77,66
Field D* ε = 2	1,07	1,00	1,32

На рисунку 3.18 відображено середнє співвідношення характеристик алгоритмів D* Lite порівняно з Field D* для всіх розглянутих мап та для всіх розглянутих відсотків невідомих перешкод.

Рисунок 3.18 – Середнє співвідношення характеристик алгоритму

Базуючись на отриманих експериментальних даних можна зробити наступні висновки:

– алгоритм Field D* незалежно від розміру мапи та значення множника евристичної функції знаходить найбільш коротший шлях та споживає найменшу кількість пам’яті, але, при цьому, має найбільш довгий час виконання;

– для обох алгоритмів підвищення коефіцієнту евристики призводить до того, що довжина отриманого шляху збільшується, але, при цьому зменшується максимальне споживання пам’яті та скорочується час роботи алгоритму;

– показники алгоритмів D* Lite та Field D* при різних значеннях множника евристичної функції дозволяє ОПР обрати відповідний алгоритм в залежності від того, які показники та на скільки йому більш важливі ніж інші;

– при збільшені кількості невідомих перешкод обидва алгоритмам прискорюють швидкість роботи, знижують споживання пам’яті, але знаходять більш довгий шлях.

На рисунку 3.19 було візуалізовано знайдені шляхи алгоритмами D* Lite та Field D* з множниками евристики 1 та 2 для мапи розміром 1000 × 1000.

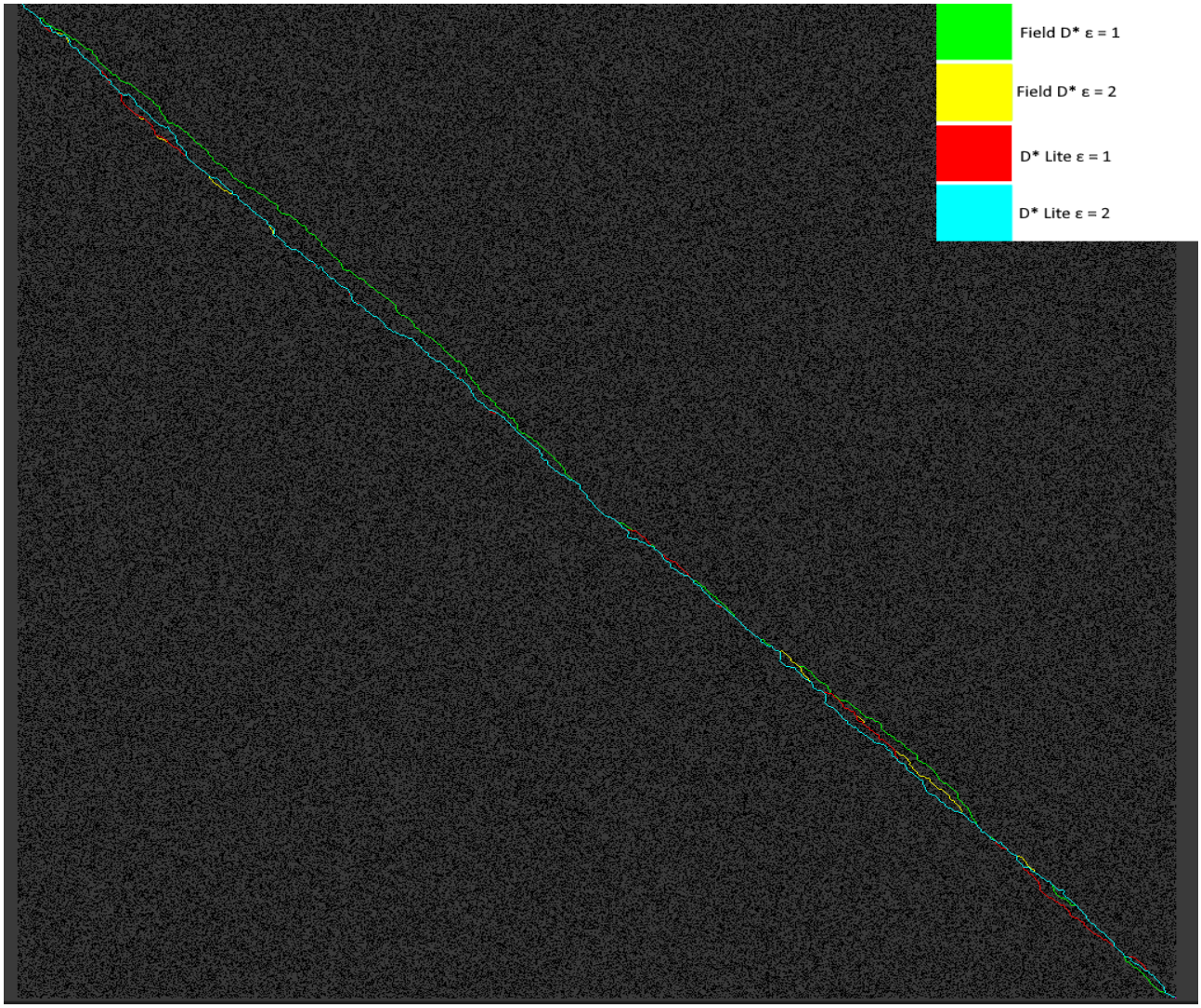


Рисунок 3.19 – Візуалізація знайдених шляхів алгоритмами D* Lite та Field D* з множниками евристики 1 та 2

При розгляді першого сектору мапи можна побачити як алгоритми по різному уникають перешкод це зображено на рисунку 3.20.

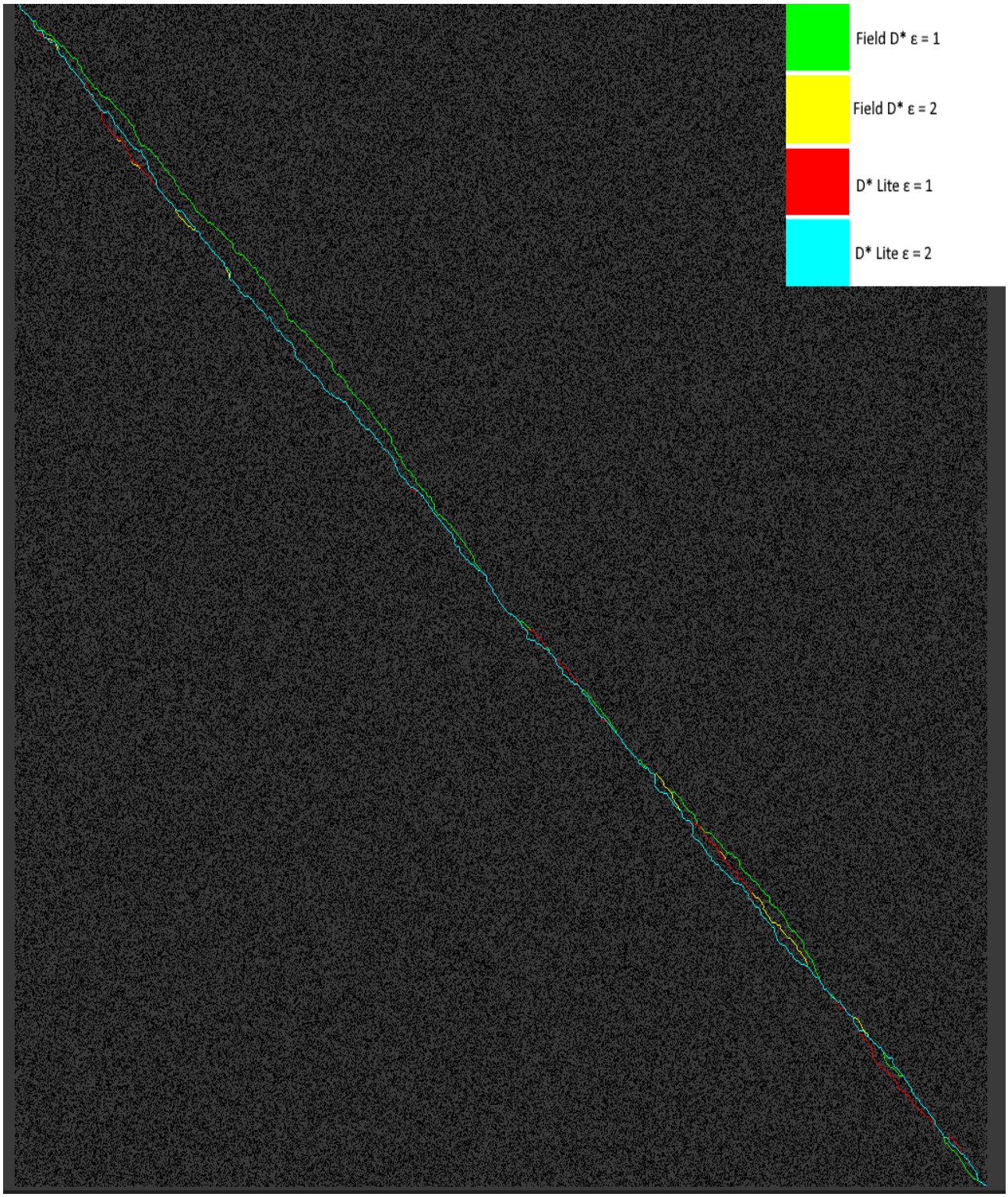


Рисунок 3.20 – Наближення першого сектора мапи

При розгляді другого сектору мапи можна побачити як алгоритми по різному уникають перешкод це зображено на рисунку 3.21.

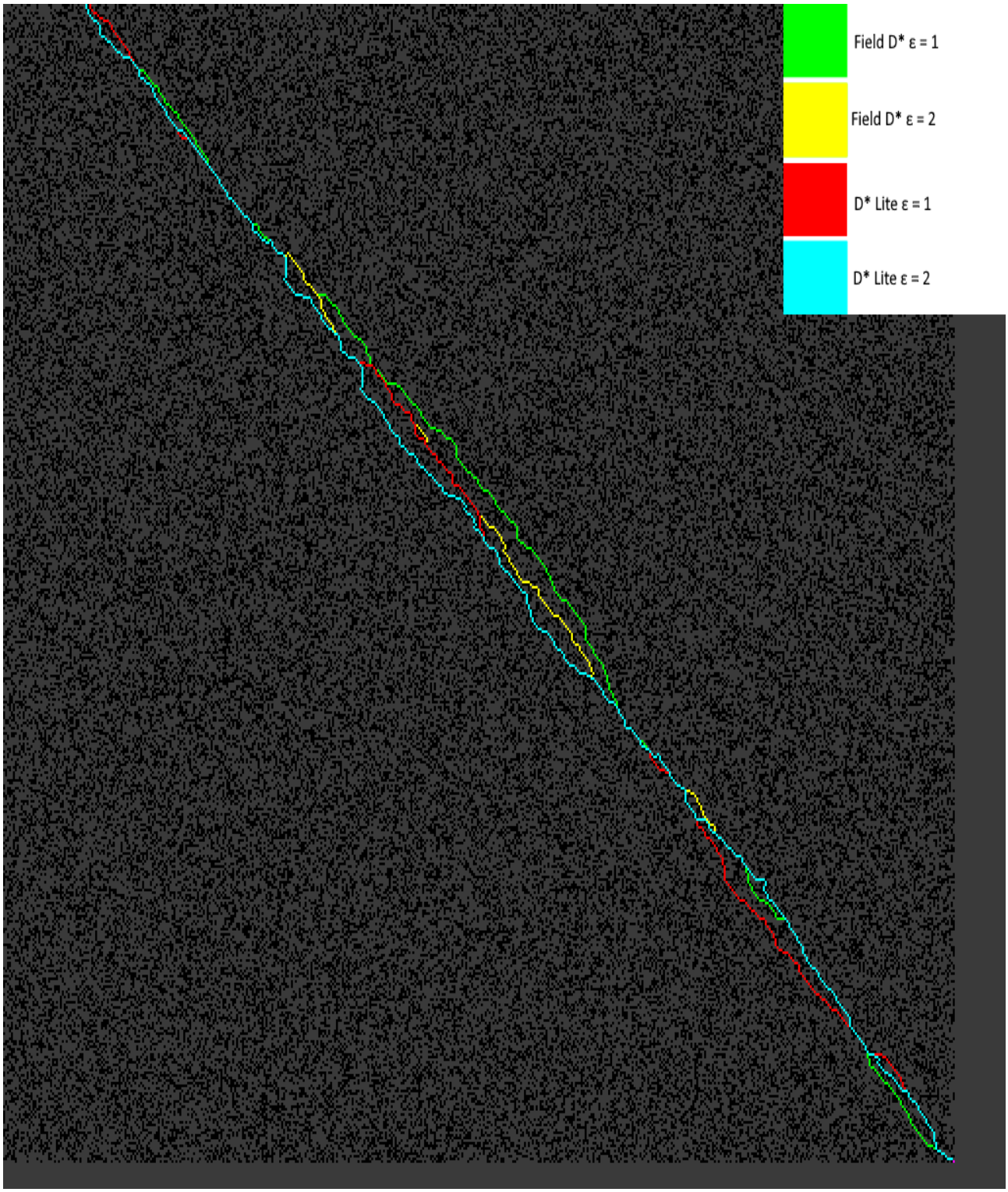


Рисунок 3.21 – Наближення другого сектора мапи

При розгляді третього сектору мапи можна побачити як алгоритми по різному уникають перешкод це зображено на рисунку 3.22.

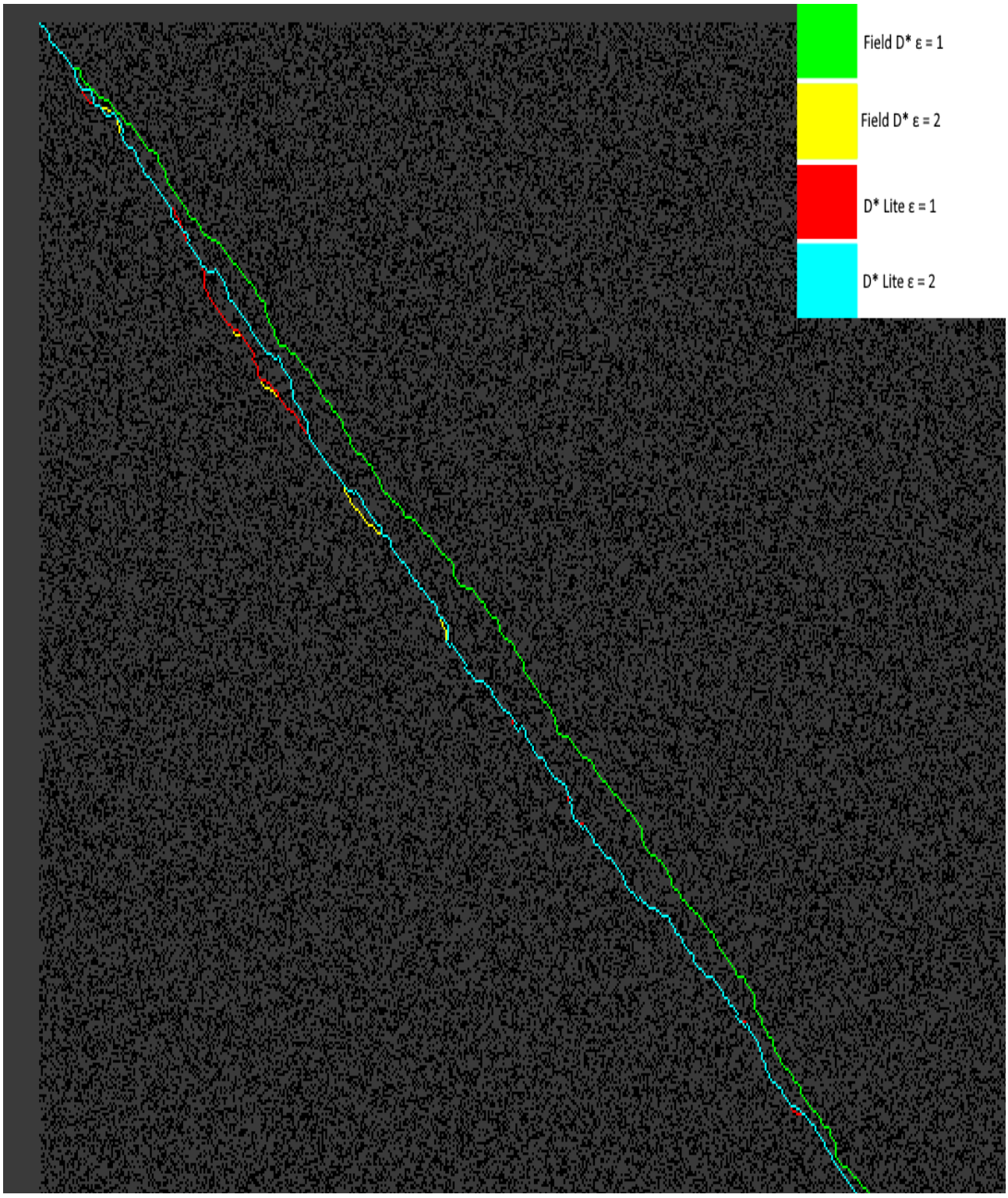


Рисунок 3.22 – Наближення третього сектора мапи

При розгляді четвертого сектору мапи можна побачити як алгоритми по різному уникають перешкод це зображено на рисунку 3.23.

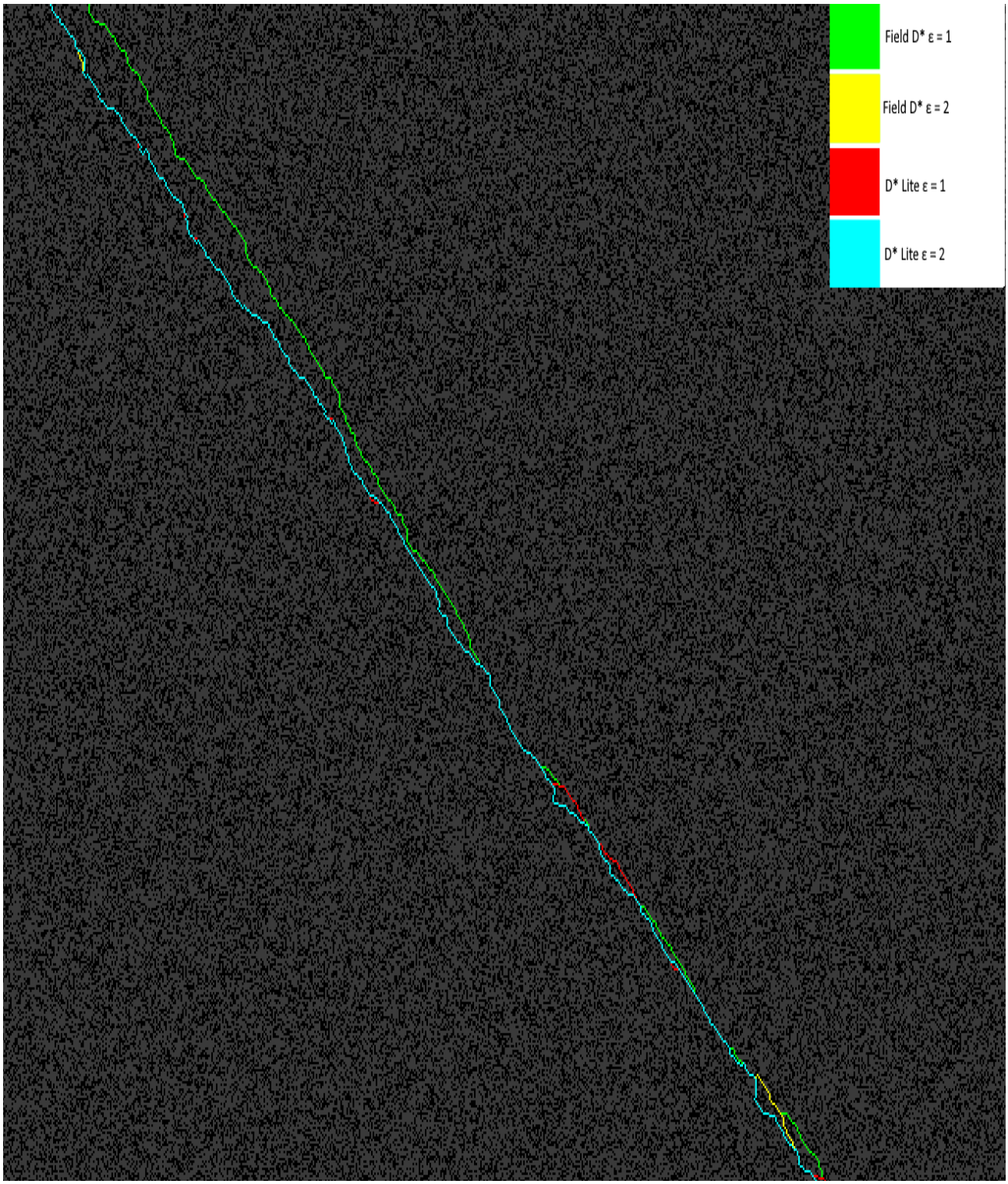


Рисунок 3.23 – Наближення четвертого сектора мапи

При розгляді п’ятого сектору мапи можна побачити як алгоритми по різному уникають перешкод це зображено на рисунку 3.24.

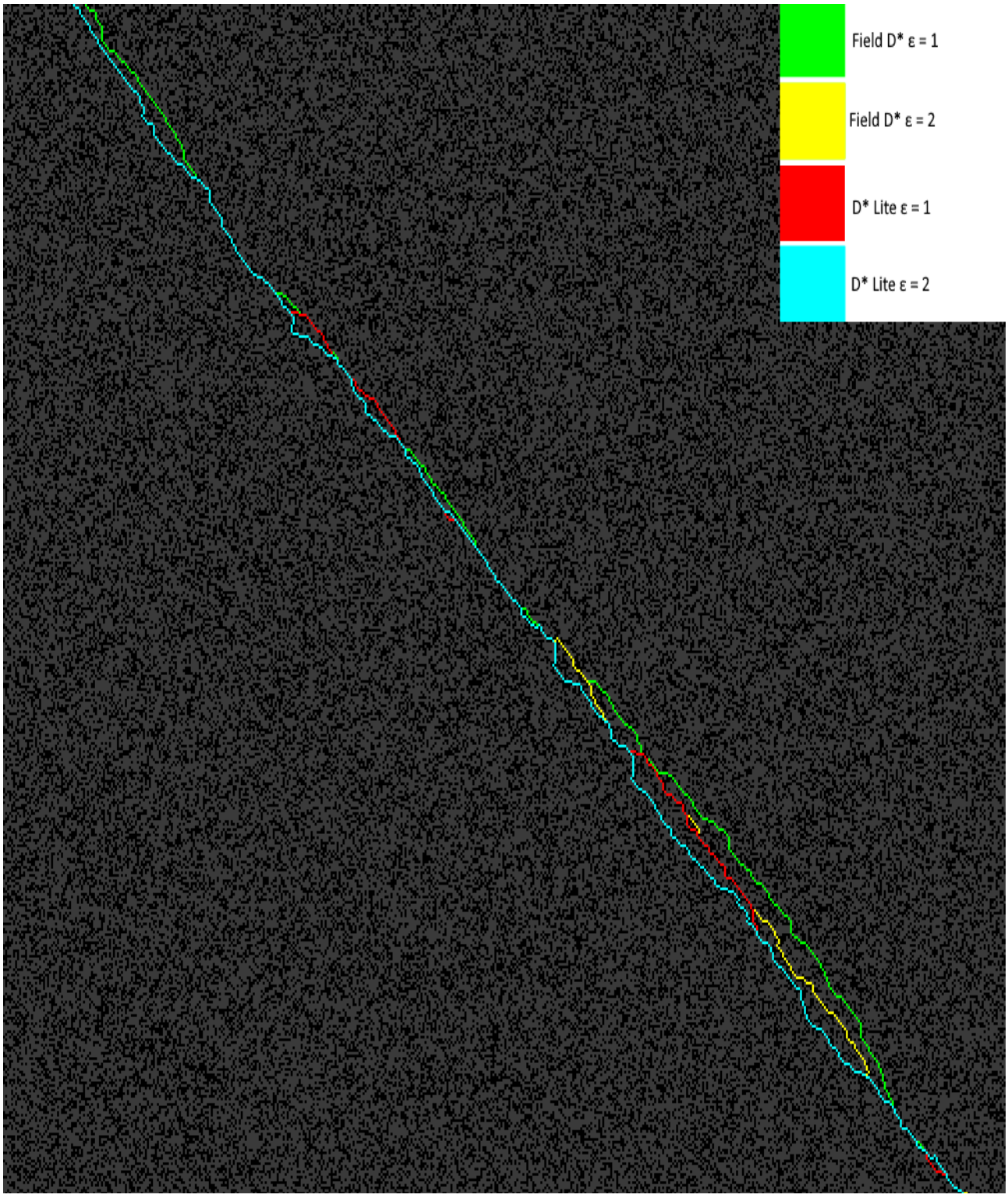


Рисунок 3.24 – Наближення п’ятого сектора мапи

3.4 Висновки до третього розділу

У розділі було проведено імітаційне моделювання алгоритмів для визначення їй характеристик, які необхідні для створення СППР, алгоритми було поділено на алгоритми роботи в динамічному та статичному середовищах, було з’ясовано вплив множників евристичної функції на показники алгоритмів, для кожного середовища зроблена порівняння алгоритмів та їхніх характеристик, зроблені відповідні висновки.

Всі розглянуті алгоритми пропонують покращення одних характеристик за рахунку погіршення інших, це дає змогу створити СППР яка здатна порекомендувати алгоритм в залежності від потреб користувача. Було також з’ясовано, що алгоритми які використовують зони видимості чи лілейну інтерполяцію знаходять більш коротший шлях та споживають менше пам’яті чим алгоритми які не використовують цього, але в той же час алгоритми яке це використовують потребують значно більше часу на обчислення.

4 СТВОРЕННЯ СППР

4.1 Вибір засобів реалізації

У якості мови реалізації СППР було обрано мову C#. C# - це об'єктно-орієнтована мова програмування від Microsoft, яка дозволяє розробникам створювати додатки, що працюють на платформі .NET. Мова C# походить з сімейства мов програмування C і має багато спільних характеристик з мовами C та C++, а також з мовами Java та JavaScript. C# забезпечує базову, читабельну мову для побудови логіки програми, приховуючи при цьому значну частину складності, що лежить в основі закладених в ній можливостей. C# вважається сильно типізованою мовою, що означає, що кожна змінна і константа має тип, так само як і вирази, які обчислюють значення. Тип описує структуру і поведінку даних. Це важливо при визначенні та роботі зі змінними, які можна розглядати як екземпляри типів [30].

При створенні додатків на C# розробники можуть використовувати оголошення типів для створення нових типів. Оголошення типу визначає ім'я та члени нового типу. Оголошення типів базуються на шести підкатегоріях, доступних для типів-значень та типів-посилань. До них відносяться типи-структури, типи-перерахування, типи-значення кортежів, типи-класи, типи-інтерфейси та типи-делегати. Мова C# призначена для роботи з платформою Microsoft .NET - програмною екосистемою для розробки, компіляції та виконання коду додатків.

Платформа включає в себе середовище виконання спільної мови (CLR) та набір бібліотек класів. CLR виконує код і надає послуги, які дозволяють і покращують розробку додатків і крос-платформних проектів. Вона також пропонує високорівневу підтримку таких мов програмування, як C#, F# та Visual Basic. Коли розробник створює додаток на C#, вихідний код компілюється в проміжну мову (IL), яка відповідає стандарту Common Language Infrastructure.

Код IL та інші ресурси програми зберігаються в збірці, яка завантажується в CLR під час запуску програми. CLR і платформа .NET також включають в себе функції, які допомагають спростити і поліпшити розробку додатків [31].

Також для реалізації СППР буде використовуватися Windows Forms. Windows Forms (WinForms) - це безкоштовна бібліотека графічних (GUI) класів з відкритим вихідним кодом, що входить до складу Microsoft .NET, .NET Framework або Mono Framework, забезпечуючи платформу для написання клієнтських додатків для настільних, портативних та планшетних комп'ютерів. Хоча вона розглядається як заміна більш ранньої та складної бібліотеки класів Microsoft Foundation Class Library, заснованої на C++, вона не пропонує порівнянної парадигми і діє лише як платформа для рівня користувальницького інтерфейсу в багаторівневому рішенні.

Додаток Windows Forms - це додаток, керований подіями, що підтримується платформою Microsoft .NET Framework. На відміну від пакетної програми, вона витрачає більшу частину свого часу на очікування дій користувача, наприклад, заповнення текстового поля або натискання кнопки. Код програми може бути написаний на мові програмування .NET, такій як C# або Visual Basic. Windows Forms надає доступ до власних елементів управління користувальницького інтерфейсу Windows, обертаючи існуючий Windows API в керований код. За допомогою Windows Forms .NET Framework забезпечує більш повну абстракцію над Win32 API, ніж Visual Basic або MFC.

Windows Forms схожа на бібліотеку Microsoft Foundation Class (MFC) при розробці клієнтських додатків. Вона надає обгортку, що складається з набору класів C++ для розробки Windows-додатків. Однак, вона не надає фреймворк додатків за замовчуванням, як MFC. Кожен елемент управління в додатку Windows Forms є конкретним екземпляром класу.

Усі візуальні елементи в бібліотеці класів Windows Forms походять від класу Control. Це забезпечує мінімальну функціональність елемента користувальницького інтерфейсу, таку як розташування, розмір, колір, шрифт, текст, а також загальні події, такі як клацання мишею та перетягування.

4.2 Алгоритм роботи СППР

Як було зазначено у розділі 1 – у якості метода прийняття рішень в умовах векторної (багатокритеріальної) оптимізації було обрано метод згортання критеріїв. Данні, які необхідні для використання цього методу були отримані у ході імітаційного моделювання в розділі 3. Алгоритм роботи СППР зображено на рисунку 4.1.

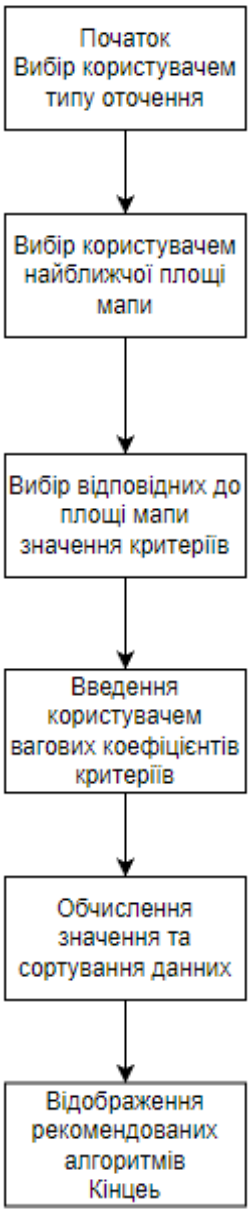


Рисунок 4.1 – Алгоритм роботи СППР

4.3 Тестування створеної СППР

Зовнішній вигляд створеної СППР відображено на рисунку 4.2.

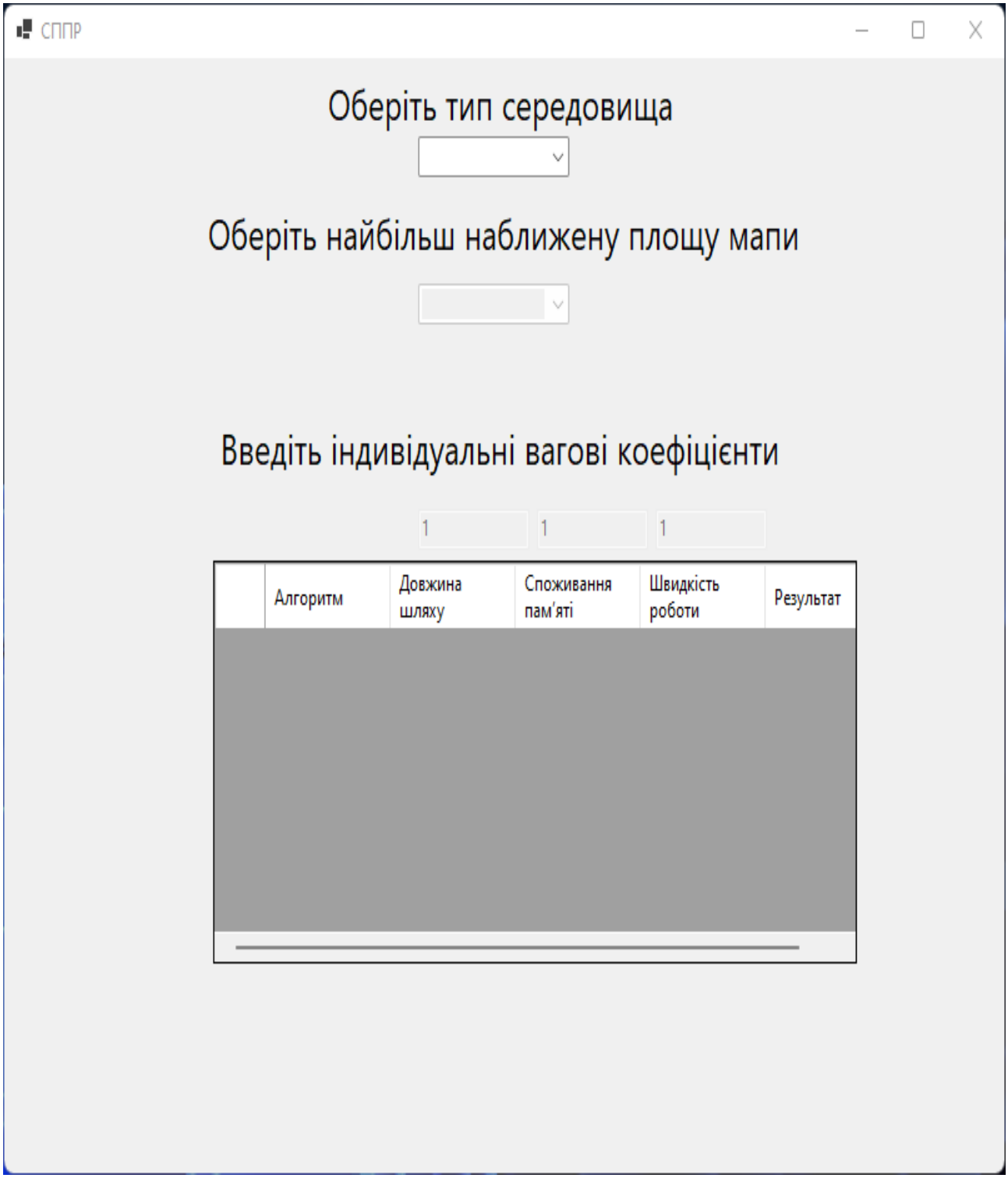


Рисунок 4.2 – Зовнішній вигляд створеної СППР

На рисунку 4.3 можна побачити рекомендовані СППР алгоритми для обраного типу середовища та найбільш наближеної площі мапи, для введених даних СППР рекомендує використовувати алгоритм A* з коефіцієнтом евристики 2.

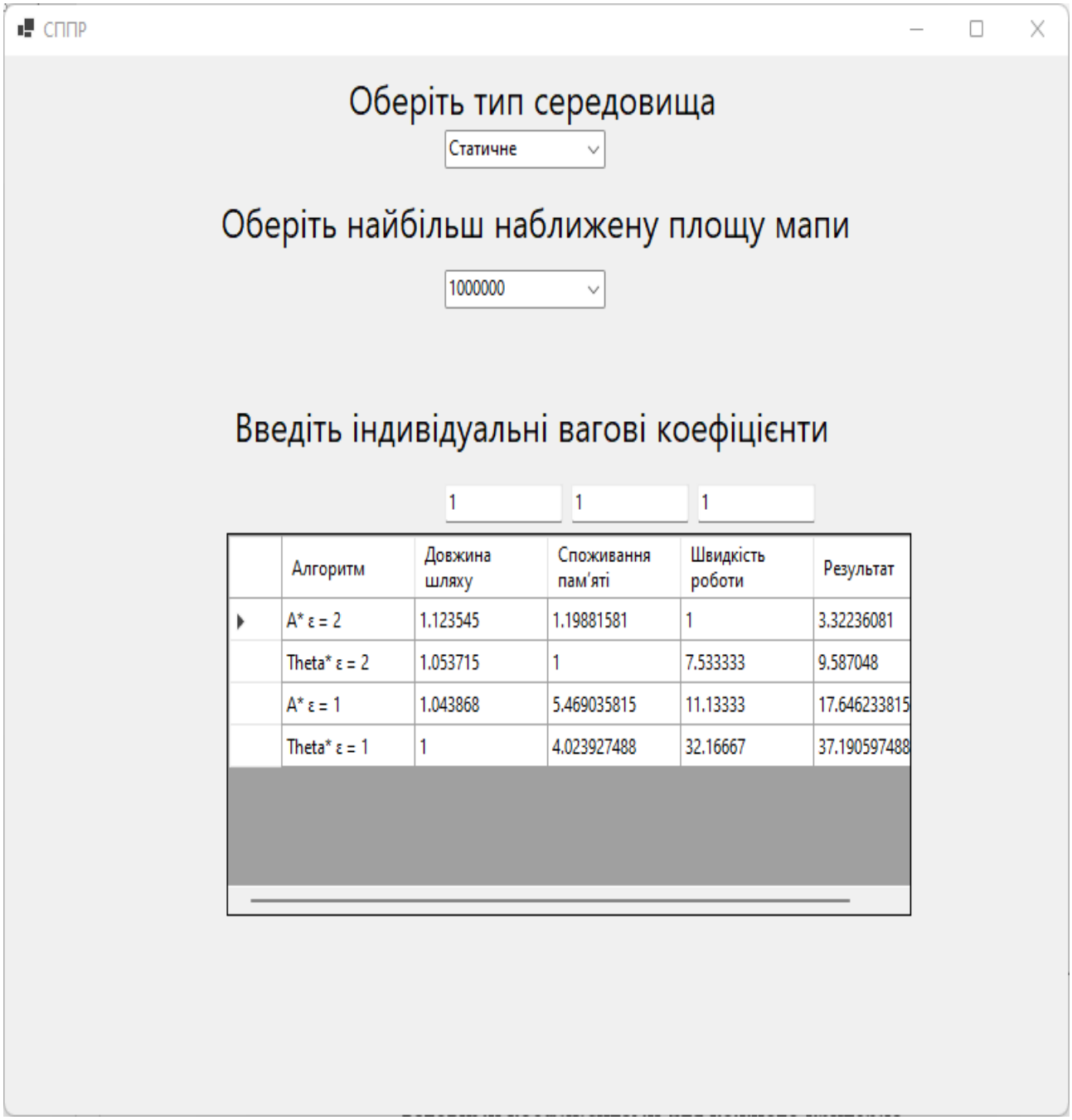
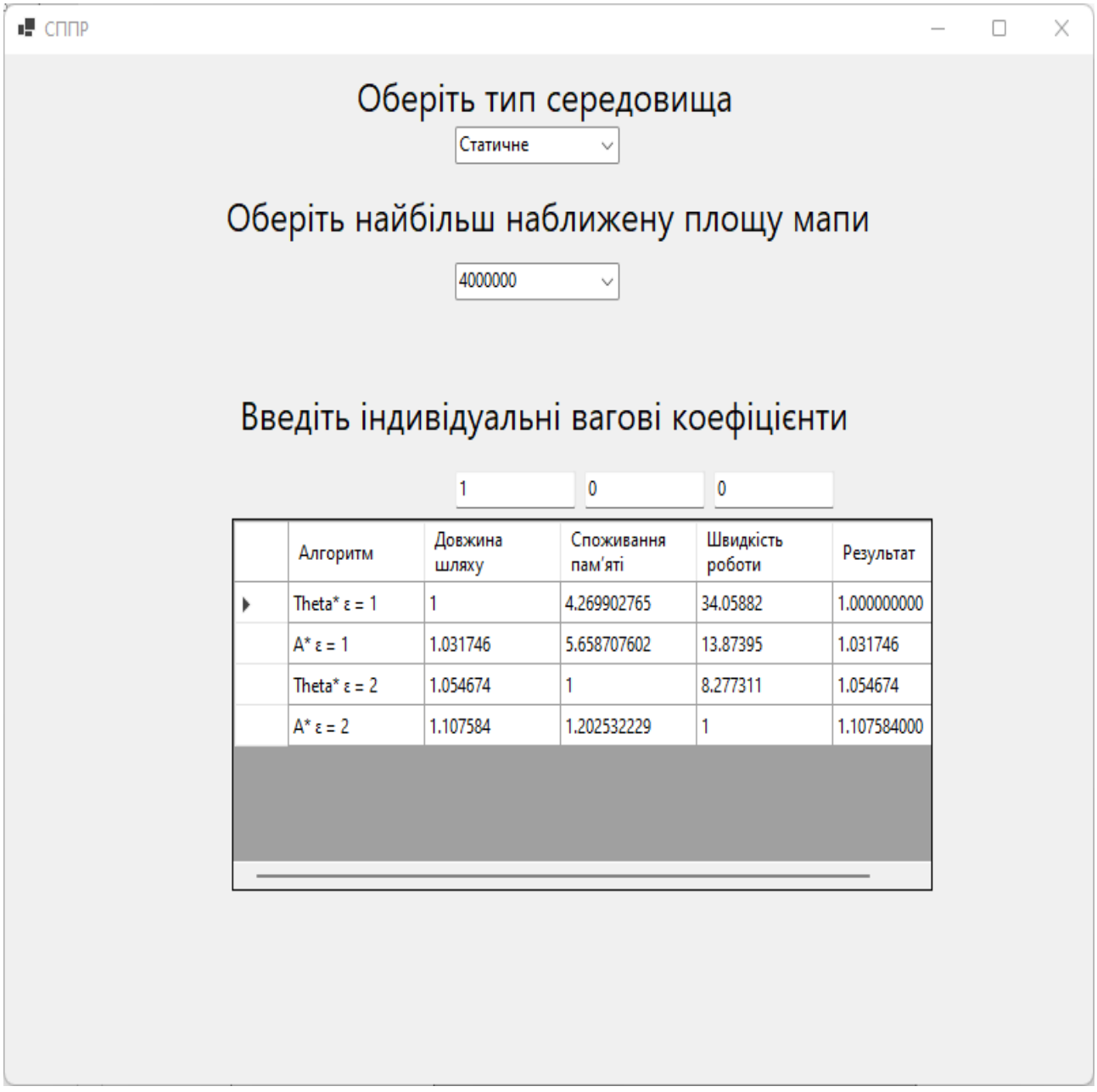


Рисунок 4.3 – Результати роботи СППР для статичного середовища при найбільш наближеної площі мапи 1000000 та з різними індивідуальними ваговими коефіцієнтами для кожного критеріїв

На рисунку 4.4 можна побачити рекомендовані СППР алгоритми для обраного типу середовища та найбільш наближеної площі мапи, для введених даних СППР рекомендує використовувати алгоритм Theta* з коефіцієнтом евристики 1, далі йде алгоритм A* з коефіцієнтом евристики 1, ОПР може обрати бажаний алгоритм використовуючи свої знання, або додаткові емпіричні данні.



	Алгоритм	Довжина шляху	Споживання пам'яті	Швидкість роботи	Результат
▶	Theta* ε = 1	1	4.269902765	34.05882	1.000000000
	A* ε = 1	1.031746	5.658707602	13.87395	1.031746
	Theta* ε = 2	1.054674	1	8.277311	1.054674
	A* ε = 2	1.107584	1.202532229	1	1.107584000

Рисунок 4.4 – Результати роботи СППР для статичного середовища при найбільш наближеної площі мапи 4000000 та з різними індивідуальними ваговими коефіцієнтами для кожного критерію

На рисунку 4.5 можна побачити рекомендовані СППР алгоритми для обраного типу середовища та найбільш наближеної площі мапи, для введених даних СППР рекомендує використовувати алгоритм Theta* з коефіцієнтом евристики 2, далі йде алгоритм A* з коефіцієнтом евристики 2, ОПР може обрати бажаний алгоритм використовуючи свої знання, або додаткові емпіричні данні.

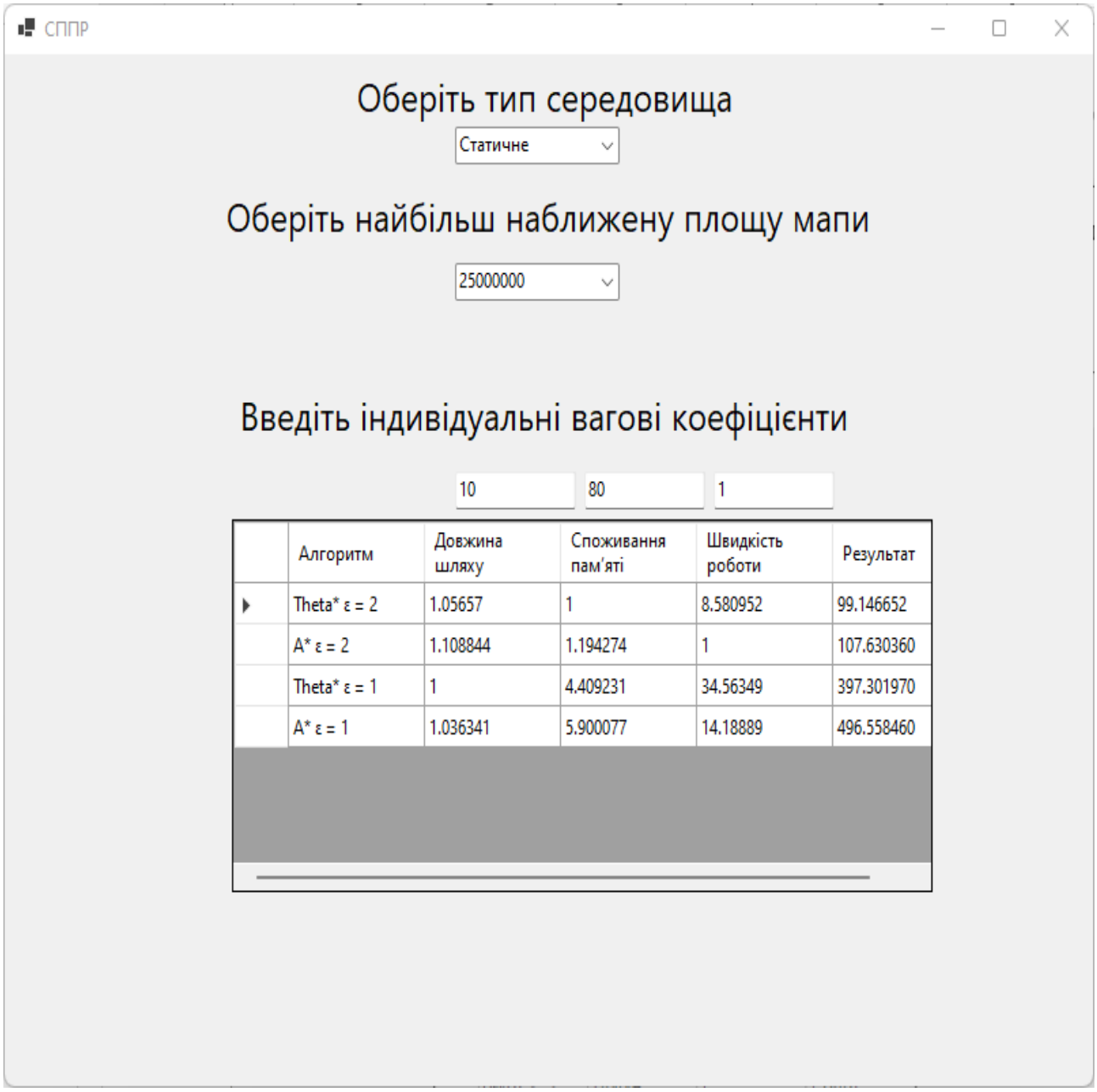
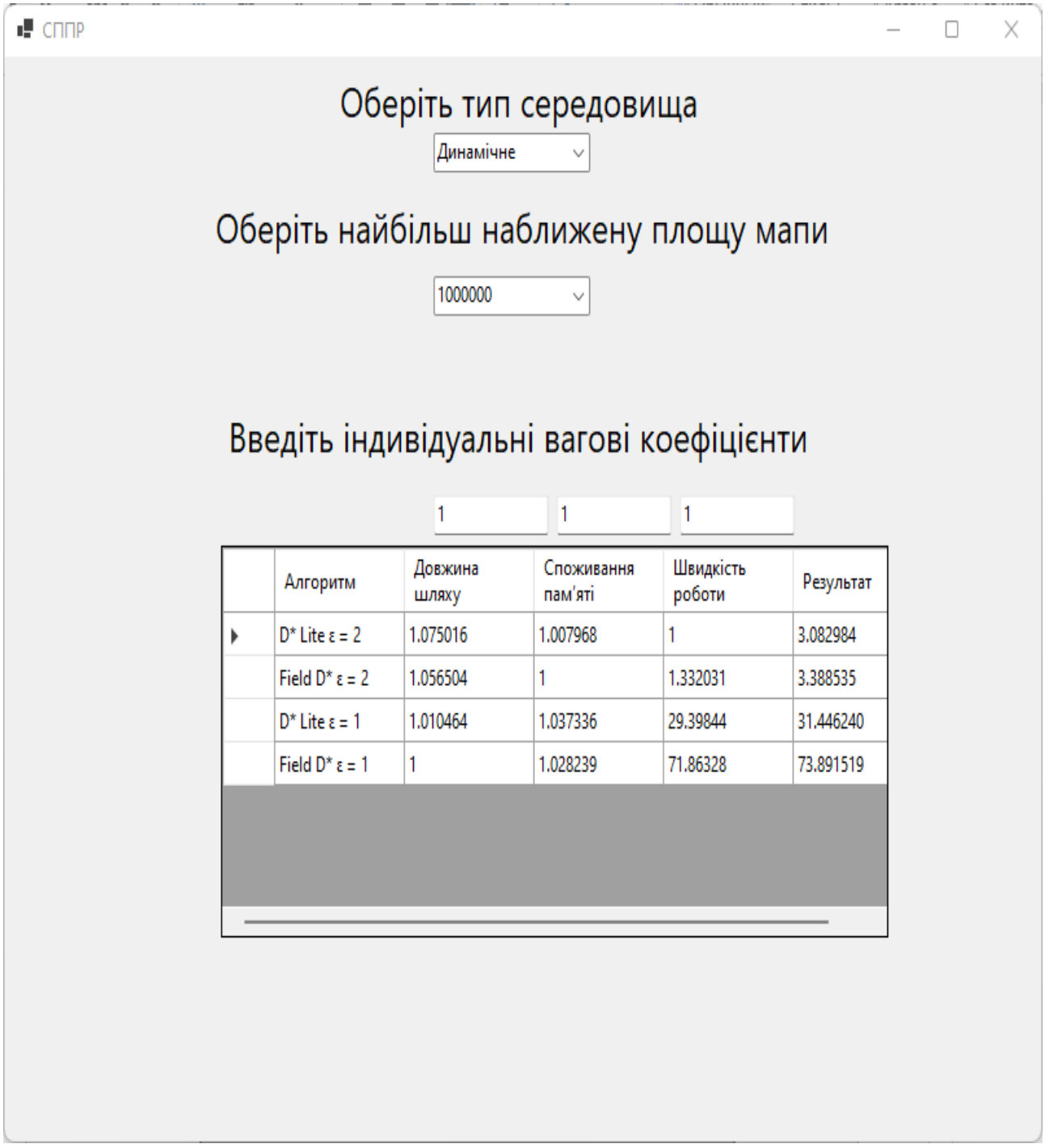


Рисунок 4.5 – Результати роботи СППР для статичного середовища при найбільш наближеної площі мапи 25000000 та з різними індивідуальними ваговими коефіцієнтами для кожного критерію

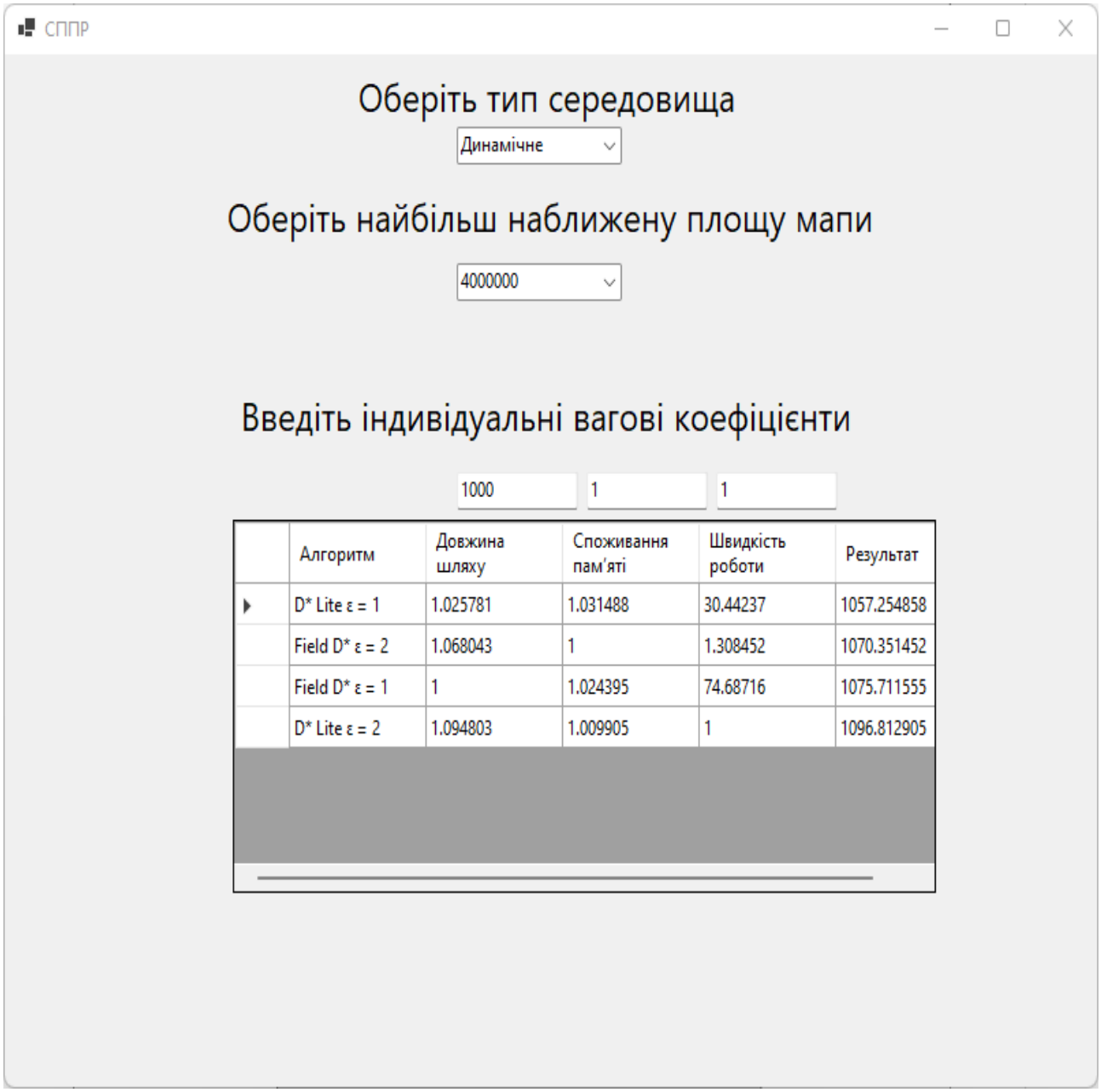
На рисунку 4.6 можна побачити рекомендовані СППР алгоритми для обраного типу середовища та найбільш наближеної площі мапи, для введених даних СППР рекомендує використовувати алгоритм D* Lite з коефіцієнтом евристики 2, далі йде алгоритм Field D* з коефіцієнтом евристики 2.



	Алгоритм	Довжина шляху	Споживання пам'яті	Швидкість роботи	Результат
▶	D* Lite ε = 2	1.075016	1.007968	1	3.082984
	Field D* ε = 2	1.056504	1	1.332031	3.388535
	D* Lite ε = 1	1.010464	1.037336	29.39844	31.446240
	Field D* ε = 1	1	1.028239	71.86328	73.891519

Рисунок 4.6 – Результати роботи СППР для динамічного середовища при найбільш наближеної площі мапи 1000000 та з різними індивідуальними ваговими коефіцієнтами для кожного критерію

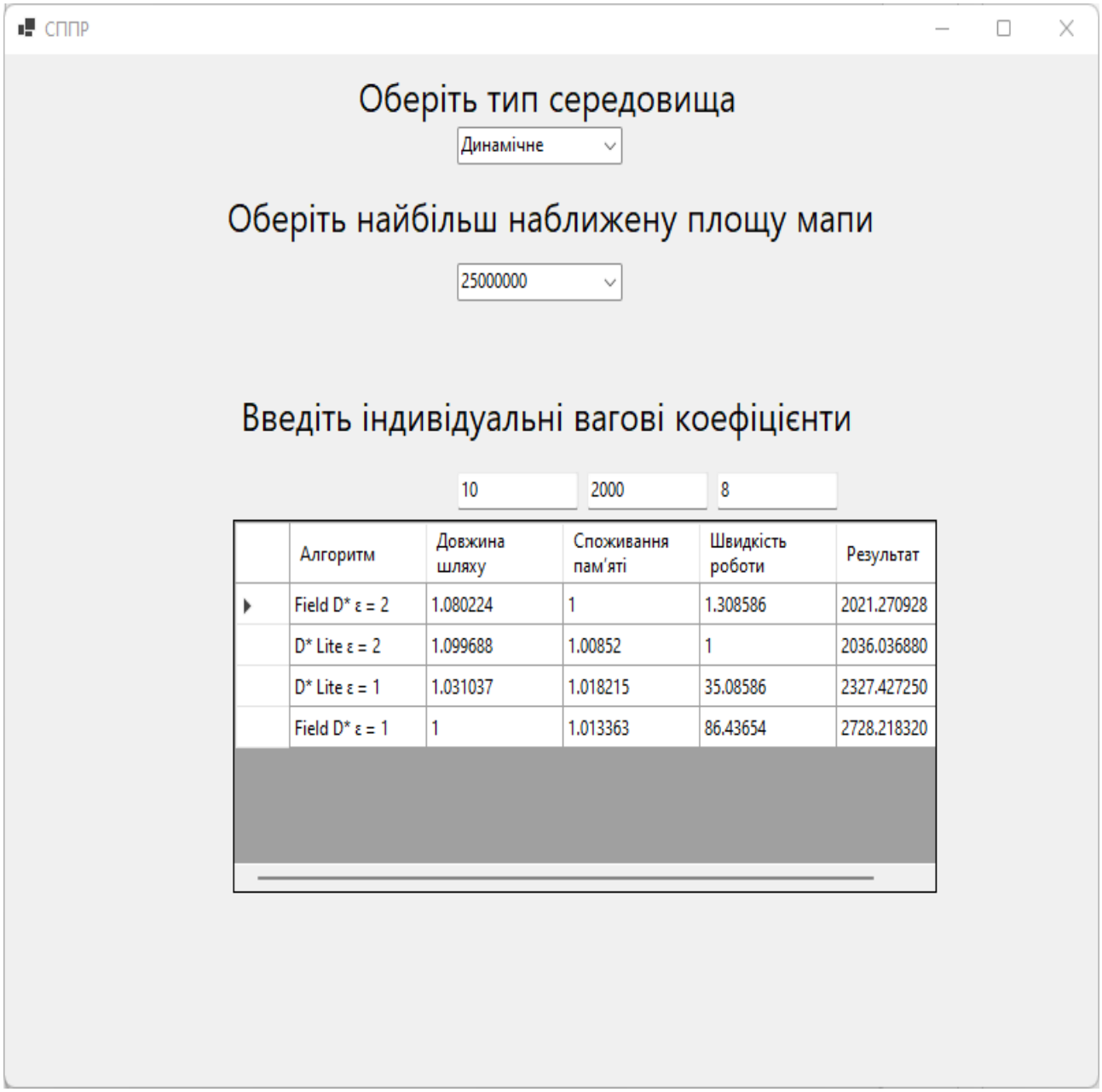
На рисунку 4.7 можна побачити рекомендовані СППР алгоритми для обраного типу середовища та найбільш наближеної площі мапи, для введених даних СППР рекомендує використовувати алгоритм D* Lite з коефіцієнтом евристики 1, далі йде алгоритм Field D* з коефіцієнтом евристики 2, ОПР може обрати бажаний алгоритм використовуючи свої знання, або додаткові емпіричні данні.



	Алгоритм	Довжина шляху	Споживання пам'яті	Швидкість роботи	Результат
▶	D* Lite ε = 1	1.025781	1.031488	30.44237	1057.254858
	Field D* ε = 2	1.068043	1	1.308452	1070.351452
	Field D* ε = 1	1	1.024395	74.68716	1075.711555
	D* Lite ε = 2	1.094803	1.009905	1	1096.812905

Рисунок 4.7 – Результати роботи СППР для динамічного середовища при найбільш наближеної площі мапи 4000000 та з різними індивідуальними ваговими коефіцієнтами для кожного критерію

На рисунку 4.8 можна побачити рекомендовані СППР алгоритми для обраного типу середовища та найбільш наближеної площі мапи, для введених даних СППР рекомендує використовувати алгоритм Field D* з коефіцієнтом евристики 2, далі йде алгоритм D* Lite з коефіцієнтом евристики 2, ОПР може обрати бажаний алгоритм використовуючи свої знання, або додаткові емпіричні данні.



	Алгоритм	Довжина шляху	Споживання пам'яті	Швидкість роботи	Результат
▶	Field D* ε = 2	1.080224	1	1.308586	2021.270928
	D* Lite ε = 2	1.099688	1.00852	1	2036.036880
	D* Lite ε = 1	1.031037	1.018215	35.08586	2327.427250
	Field D* ε = 1	1	1.013363	86.43654	2728.218320

Рисунок 4.8 – Результати роботи СППР для динамічного середовища при найбільш наближеної площі мапи 25000000 та з різними індивідуальними ваговими коефіцієнтами для кожного критерію

4.4 Висновки до четвертого розділу

Було обрано засоби створення СППР, розроблено алгоритм функціонування, та реалізовано СППР у вигляді програмного додатку. Проведено тестування роботи СППР з різними площами карт, типів оточуючого середовища та з різними індивідуальними ваговими коефіцієнтами для кожного критерію.

5 ОХОРОНА ПРАЦІ

Розробка програмного забезпечення проводиться в спеціальному приміщенні. Воно має параметри: п’ять робочих місць; один ПК (с рідкокристалічними монітором). Електрична мережа приміщення має такі характеристики: трифазна чотирипровідна мережа напругою 380/220 В змінного струму, частота 50 Гц, з глухозаземленою нейтраллю. Функціональна схема одного робочого місця представлена на рисунку 4.1.

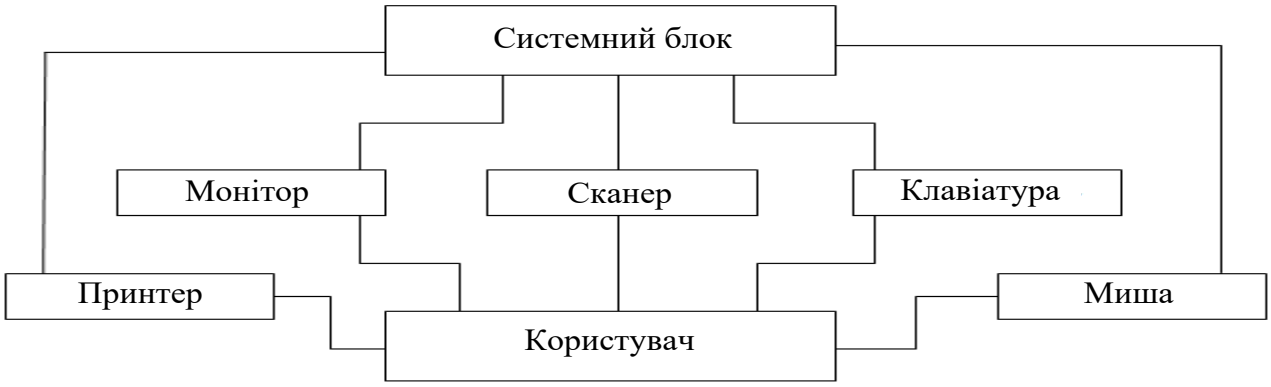


Рисунок 5.1 – Функціональна схема робочого місця

Приміщення відділу для розробки програмного забезпечення відноситься до приміщень без підвищеної небезпеки поразки людей електричним струмом згідно з НПАОП 40.1–1.21–98, так як немає умов створюють підвищену або особливу небезпеку. У приміщенні знаходиться електрощит, на якому встановлено пристрій струмового захисту. Всі розетки мають застережливий напис «220». З робітниками проводяться інструктажі з охорони праці відповідно до НПАОП 0.00-4.12-05.

Згідно з вимогами НПАОП 0.00–4.12–05 необхідно проводити вступний, первинний на робочому місці, повторний, а при необхідності – позаплановий і цільовий інструктажі [32].

Схема занулення представлена на рисунку 5.2, параметри цієї схеми відображені в таблиці 5.1.

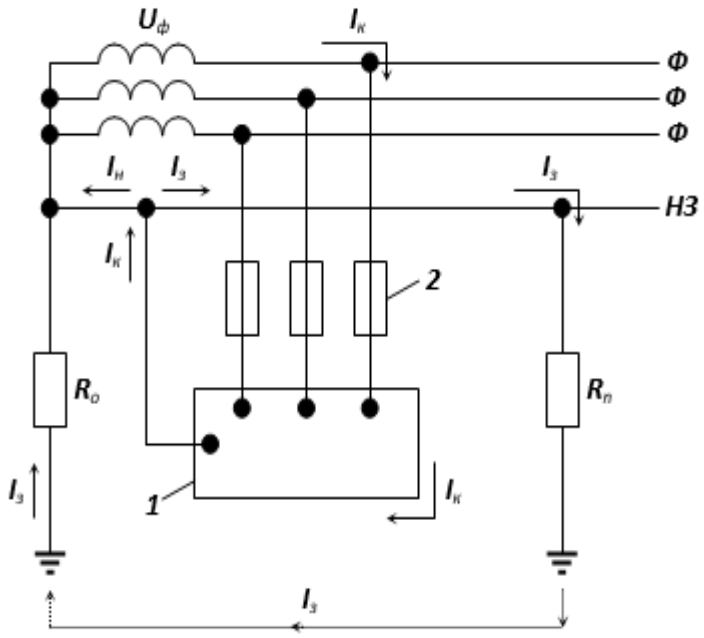


Рисунок 5.2 – Принципова схема занулення в трифазній мережі до 1000В

Таблиця 5.1 – Параметри схеми занулення в трифазній мережі до 1000 В

1	корпус електроустановки;
2	апарати захисту від струмів короткого замикання (КЗ)
Ф	фазний провід;
НЗ	нульовий захисний провідник;
$R_о$	опір заземлення нейтрали обмотки джерела струму;
$R_п$	опір повторного заземлення нульового захисного провідника;
$I_к$	ток КЗ;
$I_н$	частина струму КЗ, що протікає через нульовий захисний провідник;
$I_з$	частина струму КЗ, що протікає через землю.

Робоче місце оператора ПК піддається впливу таких небезпечних і шкідливих виробничих факторів як недостатня освітленість робочої зони, підвищений потенціал електростатичного поля, електричний струм, психофізіологічні навантаження, перевантаження аналізаторів.

5.1 Висновки до п’ятого розділу

Проведено огляд робочого місця, визначенні основні параметри , наведено необхідні вимоги, визначенні можливі фактори небезпеки.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було проведено аналіз технічного завдання, а саме – розглянуті системи підтримки прийняття рішень, їх структура та методи розв'язання оптимізаційних задач, було розглянуто питання мобільних роботів, задач керування мобільними роботами, знайдено та проаналізовано існуючи підходи до автономної навігації мобільних роботів. Проаналізовано існуючи дослідження та виявлено, що по сумі критерієві саме пошук за першим найкращим збігом є найбільш оптимальних, у наведених дослідженнях він був представлений алгоритмом A*. Було прийнято рішення проаналізувати алгоритми пошуку за першим найкращим збігом, визначити їх параметри та на основі отриманих даних створити СППР для задач навігації мобільної платформи.

Другий розділ дипломної роботи було присвячено детальному розгляду та аналізу алгоритмів пошуку за першим найкращим збігом, розглянуто евристичні функції, алгоритм A*, Theta*, D* Lite, Field D*, розглянуто механізми їх роботи та проаналізовано переваги і недоліки кожного, наведено приклад використання алгоритму Field D* у якості алгоритмам пошуку шляху для марсохода Opportunity.

Третій розділ дипломної роботи було присвячено проведенню імітаційного моделювання для визначення показників алгоритмів та у подальшому – створенні СППР використовуючи ці показники для підтримки прийняття рішень.

Створено імітаційну модель, визначені показники алгоритмів для роботи в статичному середовище, а саме для алгоритмів A* та Theta*, проведено порівняння їх показників, визначено, що використання різних множників евристичної функції та різних алгоритмів дає можливість обрати найбільш оптимальний алгоритм у залежності від потреб ОПР, що дозволяє використовувати отриманні данні при створенні СППР. Також було визначено

показники алгоритмів для роботи в динамічному оточенні таких як D* Lite та Field D*, порівнянні їх показники, визначено, що використання різних множників евристичної функції та різних алгоритмів дає можливість обрати найбільш оптимальний алгоритм у залежності від потреб ОПР, що дозволяє використовувати отриманні данні при створенні СППР.

Для кожного порівняння було візуалізовано знайдені алгоритмам шляхи, що дозволяє додатково візуально побачити різницю між ними.

У четвертому розділі було створено СППР, для цього використовувалися результати отримані у третьому розділі та вибрані засоби реалізації створеного алгоритму роботи СППР, було проведено тестування СППР та визначено коректність функціонування.

Базуючись на виконаній роботі та отриманим даним, можна визначити, що якщо необхідно створення більш точної СППР, то для цього пропонується наступне: необхідно додати до СППР експериментальний модуль та модуль формування звітності, експериментальний модуль має бути з’єднаний з роботом з відповідним ПЗ для проведення експерименту, модуль формування звітності має обробляти данні експериментального модуля та вносити відповідні результати до бази знань СППР, але перед цим необхідно провести аналіз наскільки необхідне і рентабельне створення такої СППР. СППР створення в ході цієї роботи дає змогу отримати відносну точну рекомендацію щодо вибору алгоритму навігації мобільної платформи.

Наукова новизна цієї роботи полягає у тому, що було проаналізовано характеристик алгоритмів A*, Theta*, D* Lite та Field D* в залежності від параметрів евристичної функції, виконано їх порівняння в залежності від типу оточення, та створено СППР, яка дозволяє в загальному випадку обрати оптимальний алгоритм в залежності від бажаних критеріїв.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ДСТУ 3008: 2015. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлення. К.: ДП “УкрНДНЦ”. 2016. 30 с.

2. Методичні вказівки з підготовки та захисту кваліфікаційної роботи здобувачами другого (магістерського) рівня вищої освіти спеціальності 151 Автоматизація та комп’ютерно-інтегровані технології, освітньо-професійних програм: «Автоматизоване управління технологічними процесами»,«Комп’ютерно-інтегровані технологічні процеси і виробництва», «Комп’ютеризовані та робототехнічні системи» / Упоряд. І. Ш. Невлюдов, Р. В. Артюх, В. В. Безкоровайний, Н. П. Демська, В. В. Євсєєв, О. І. Филипенко, О. М. Цимбал. Харків: ХНУРЕ, 2021. 55 с.

3. Системи і методи підтримки прийняття рішень / П.І. Бідюк, О.Л. Тимощук, А.Є. Коваленко, Л.О. Коршевнюк. Київ: КПІ ім. Ігоря Сікорського, 2022. С. 20 – 23.

4. A review of mobile robots: Concepts, methods, theoretical framework, and applications / Francisco Valero, Carlos Llopis-Albert. International Journal of Advanced Robotic Systems, 2019. С. 4 – 6.

5. A Comprehensive Study of Mobile Robot: History, Developments, Applications, and Future Research Perspectives / Manuel Armada, Seong-Ik Han. Applied Sciences, 2022. С. 35 – 36.

6. Будова та система керування мобільної платформи на базі робота павука / Бєлов П.О., Мандрікін М.С., Катков О.В., Цимбал О.М. Харків: Topical issues of modern science, society and education. Proceedings of the 6th International scientific and practical conference, 2021. С. 343.

7. A Comprehensive Study of Mobile Robot: History, Developments, Applications, and Future Research Perspectives / Manuel Armada, Seong-Ik Han. Applied Sciences, 2022. С. 8 – 12.

8. Genetic Algorithm Based Approach for Autonomous Mobile Robot Path

Planning / Chaymaa Laminia , Said Benhlimab, Ali Elbekr. The First International Conference On Intelligent Computing in Data Sciences, 2018. С. 323 – 328.

9. Neural Networks for Mobile Robot Navigation: A Survey / An-Min Zou, Zeng-Guang Hou, Si-Yao Fu. The Chinese Academy of Sciences, 2006. С. 10 – 12.

10. Application of Fuzzy Logic in Mobile Robot Navigation / Tang Sai Hong, Danial Nakhaeinia, Babak Karasfi. Fuzzy Logic Controls, Concepts, Theories and Applications, 2016. С. 120 – 123.

11. A study on best first search / M.Sinthiya, Dr.M. Chidambaram. IRJET, 2016. С. 497 – 499.

12. A comprehensive study for robot navigation technique / Faiza Gul, Wan Rahiman, Syed Sahal Nazli Alhady. Cogent Engineering, 2019. С. 534 – 535.

13. Real-Time Map Manipulation for Mobile Robot Navigation / Carlos Favis Ezequiel – Tampa: University of South Florida, 2013. С. 18 – 22.

14. Research on Path-Planning Algorithm Integrating Optimization A-Star Algorithm / Liu, L.; Wang, B.; Xu, H,. Electronics, 2022. С. 780 – 782.

15. Fast Path Planning for Firefighting UAV Based on A-Star algorithm / Gao, J., Zheng, Y., Ni, K,. Journal of Physics: Conference Series, 2021. С. 45 – 49.

16. Application of A-Star Algorithm on Pathfinding Game / Candra, Ade; Andri Budiman, Mohammad; Pohan, Rahmat Irfan. Journal of Physics, 2021. С. 82 – 84.

17. Generalized best-first search strategies and the optimality of A* / Dechter, Rina; Judea Pearl. Journal of the ACM, 1985. С. 32 – 38.

18. Theta*: Any-Angle Path Planning on Grids / Kenny Daniel, Alex Nash, Sven Koenig. Journal of Artificial Intelligence Research, 2010. С. 534 – 538.

19. Theta* for Any-Angle Pathfinding / Sven Koenig, Alex Nash. Game AI Pro 360, 2014. С. 634 – 635.

20. Fast Replanning for Navigation in Unknown Terrain / Koenig, S.; Likhachev, M,. Proceedings of the International Conference on Robotics and Automation, 2004. С. 360 – 362.

21. Research and Optimization of D-Start Lite Algorithm in Track Planning / Kaili Xie; Jie Qiang. IEEE Acces, 2020. С. 267 – 268.

22. D* Lite Algorithm for Autonomous Mobile Robot / Velappa Ganapathy; Soh Chin Yun. International Journal of Applied Science and Technology, 2011. С. 58 – 59.

23. A Systematic Review and Analysis of Intelligence-Based Pathfinding Algorithms in the Field of Video Games / Lawande, S.R.; Jasmine, G.; Anbarasi, J.; Izhar, L.I., Appl. Sci., 2022. С. 167 – 169.

24. A Survey of Path Planning Algorithms for Mobile Robots / Karur, K.; Sharma, N.; Dharmatti, C.; Siegel, J. Vehicles, 2021. С. 448 – 450.

25. Graph-based Path Planning for Mobile Robots / Wooden, D.. Georgia Institute of Technology, 2006. С. 129 – 130.

26. Field D*: An Interpolation-Based Path Planner and Replanner / Dave Ferguson, Anthony Stentz. Robotics Research: Results of the 12th International Symposium, 2005. С. 239 – 241.

27. An analysis of path planning algorithms / Megan Clancy. University of Dayton, 2011. С. 18 – 19.

28. D-Star Panorama by Opportunity. [Електроний ресурс]. Режим доступу: https://www.nasa.gov/mission_pages/mer/images/sol1162B-20080102a.html (дата звернення 09.10.2022). Назва з екрана.

29. Simulation modeling-An effective method in doing business and management research / Tram T. B. Nguyen. Open university journal of science, 2021. С. 43 – 44.

30. A tour of the C# language. [Електроний ресурс]. Режим доступу: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/ (дата звернення 09.10.2022).

31. C# - Overview. [Електроний ресурс]. Режим доступу: https://www.tutorialspoint.com/csharp/csharp_overview.htm (дата звернення 09.10.2022).

32. Перелік робіт з підвищеною небезпекою (НПАОП 0.00-4.12-2005). [Електроний ресурс]. Режим доступу https://zakon.rada.gov.ua/laws/show/z0232-05#Text (дата звернення 09.10.2022).