Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет ННЦЗФН
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Використання еволюційного навчання глибокої нейронної мережі в завданні обходу перешкод
(тема)

Виконав:
студент 2 курсу, групи СШІзм-20-1
Скворцов Д.О.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

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

Освітня програма Системи штучного інтелекту
(повна назва спеціалізації)

Керівник к.т.н., доц. Дейнеко А.О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ В.О. Філатов
(підпис) (прізвище, ініціали)

2021 р.

Харківський національний університет радіоелектроніки

Факультет ННЦЗФН
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

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

Освітня програма Системи штучного інтелекту (СШІ)
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 20 __ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Скворцову Денису Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Використання еволюційного навчання глибокої нейронної мережі в завданні обходу перешкод

затверджена наказом університету від 25 жовтня 2021 р. № 163Стз

2. Термін подання студентом роботи до екзаменаційної комісії _____ _____ 20 __ р.

3. Вихідні дані до роботи Науково-технічна література та публікації, дані Інтернет-джерел та відомих наукових проектів щодо розробки нейронних мереж, дані методів обходу перешкод та пошуку маршруту, порівняльна характеристика методів обходу перешкод та пошуку маршруту

4. Перелік питань, що потрібно опрацювати в роботі

1) Огляд основних понять і галузі дослідження

2) Методи обходу перешкод і пошуку маршруту

3) Нейроеволюційний підхід у завданні обходу перешкод

4) Імітаційне моделювання та вирішення задачі обходу перешкод

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Рисунок 1.1 – Нелінійна модель нейрона, Рисунок 1.2 – Приклади активаційних функцій, Рисунок 1.3 – Одношарова мережа прямого поширення, Рисунок 1.4 – Повнозв'язна мережа прямого поширення з одним прихованим шаром, Рисунок 1.5 - Рекурентна мережа з прихованим шаром нейронів, Рисунок 1.6 – Блокова діаграма навчання з учителем, Рисунок 1.7 – Блокова діаграма навчання без вчителя, Рисунок 1.8 – Стан перенавчання, Рисунок 2.1 – Класифікація методів планування маршруту, Рисунок 2.2 – Методи на основі графів, Рисунок 2.3 – Зважений граф G, Рисунок 2.4 – Методи на основі клітинної декомпозиції, Рисунок 3.1 – Робота генетичного алгоритму, Рисунок 4.1 – Менеджер для управління ходом навчання, Рисунок 4.2 – Навчання НМ на першій ітерації, Рисунок 4.3 – Навчання НМ через декілька ітерацій, Рисунок 4.4 – Навчання НМ через 10-20 ітерацій, Рисунок 4.5 – Навчання НМ через 20-30 ітерацій

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

Найменування розділу	Консультант (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 дипломну роботу	01.09.2021	виконано
2	Теоретичні дослідження	19.09.2021 – 01.11.2021	виконано
3	Експериментальне моделювання та навчання	04.11.2021 – 23.11.2021	виконано
4	Оформлення пояснювальної записки	25.11.2021 – 30.11.2021	виконано
5	Оформлення графічних матеріалів	01.12.2021	виконано
6	Попередній захист		
7	Захист перед ЕК		

Дата видачі завдання 01 вересня 2021 р.

Студент _____
(підпис)

Керівник роботи _____ _____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 78 с., 1 табл., 18 рис., 2 дод., 39 джерел.

ГЕНЕТИЧНИЙ АЛГОРИТМ, ГЛИБОКІ НЕЙРОННІ МЕРЕЖІ, НЕЙРОННІ МЕРЕЖІ, ОБХІД ПЕРЕШКОД, ПОШУК МАРШРУТУ, ФУНКЦІЯ АКТИВАЦІЇ, A*, C#, D*, UNITY, VFF, VFH

Метою роботи є створення багатошарової нейронної мережі з використанням генетичного (еволюційного) алгоритму навчання.

Об’єкт дослідження – процес створення маршруту з обходженням перешкод в умовах статичного навколишнього середовища.

Предмет дослідження – методи вирішення задачі обходу перешкод, в особливості за допомогою багатошарової нейронної мережі на основі генетичного алгоритму.

Методи дослідження – теорія штучних нейронний мереж, теоретичні аспекти підходів до вирішення задачі пошуку маршруту, імітаційне моделювання.

РЕФЕРАТ

Пояснительная записка: 78 с., 1 табл., 18 рис., 2 прил., 39 источников.

ГЕНЕТИЧЕСКИЙ АЛГОРИТМ, ГЛУБОКИЕ НЕЙРОННЫЕ СЕТИ, НЕЙРОННЫЕ СЕТИ, ОБХОД ПРЕПЯТСТВИЙ, ПОИСК МАРШРУТА, ФУНКЦИЯ АКТИВАЦИИ, A*, C#, D*, UNITY, VFF, VFH

Целью работы является создание многослойной нейронной сети с использованием генетического (эволюционного) алгоритма обучения.

Объект исследования – процесс создания маршрута с обходом препятствий в условиях статической окружающей среды.

Предмет исследования – методы решения задачи обхода препятствий, в особенности с помощью многослойной нейронной сети на основе генетического алгоритма.

Методы исследования – теория искусственных нейронный сетей, теоретические аспекты подходов к решению задачи поиска маршрута, имитационное моделирование.

ABSTRACT

Explanatory note: 78 p., 18 fig., 1 tabl., 2 ann., 39 sources.

A*, ACTIVATION FUNCTION, C #, D*, DEEP NEURAL NETWORKS, GENETIC ALGORITHM, NEURAL NETWORKS, OBSTACLE CIRCUMVENTION, ROUTE SEARCH, UNITY, VFF, VFH

The purpose of the qualifying master's work is to create a multilayer neural network using a genetic (evolutionary) learning algorithm.

The object of the research is the process of creating a route with obstacles in a static environment.

The subject of the research are methods of solving the problem of obstacle circumvention, in particular using a multilayer neural network based on a genetic algorithm.

Teaching methods – analysis of artificial neural networks, theoretical aspects of approaches to solving the problem of route search, simulation.

ЗМІСТ

ВСТУП

Штучний інтелект, нейронні мережі, машинне навчання та еволюційні алгоритми користуються великою популярністю останнім часом. Відповідна кількість статистичних даних пошукових запитів, що надаються сервісами Google, підтверджують зростання інтересу до цих напрямків.

Штучна нейронна мережа (ШНМ), як один з основних прийомів комп'ютерного алгоритмічного моделювання інтелекту, дозволяє вирішувати широкий спектр завдань, таких як розпізнавання образів, обробка сигналів, обхід перешкод тощо.

Модель ШНМ концептуально намагається повторити структуру мозку, який використовує спосіб обробки інформації, що принципово відрізняється від методів, що використовуються звичайними комп'ютерами. Мозок являє собою складний, нелінійний, паралельний комп'ютер, який управляє своїми структурними компонентами, нейронами.

При цьому нейронні мережі не програмуються у звичному сенсі слова, а навчаються. Однією з основних переваг використання ШНМ є можливість отримання оптимальних результатів на основі даних, що не зустрічаються в процесі навчання. Сам процес навчання НМ з математичної позиції є багатовимірним завданням нелінійної оптимізації.

Генетичний алгоритм – це тип еволюційного розрахунку, який використовує природні еволюційні методи, такі як спадкування, мутація, відбір і перетин для вирішення завдань оптимізації.

У даній роботі буде реалізована багатошарова нейронна мережа з генетичним алгоритмом навчання, призначена для рішення завдання обходу перешкод і пошуку маршруту. У рамках середовища для імітаційного моделювання обрано Unity для кращої візуалізації процесу навчання.

1 ОГЛЯД ОСНОВНИХ ПОНЯТЬ І ГАЛУЗІ ДОСЛІДЖЕННЯ

Нейронні мережі – це один з напрямків в розробці систем штучного інтелекту. Їх назва походить від принципів роботи математичної моделі, яка може нагадувати функціонування нервової системи біологічних істот. Утворюють нервову систему нейрони. Їх головне завдання – поширювати інформацію по всьому тілу, використовуючи електро- та хімічні сигнали. Вони черпають її з навколишнього середовища або організму, аналізують її, розраховують, як відреагувати, а також запам'ятовують.

Принцип роботи нейронних мереж близький до людської нервової системи, точніше він намагається імітувати біологічну нервову систему. Кожен нейрон тут – це елемент, у якого є безліч вхідних отворів для отримання інформації і одне вихідне. Спосіб, яким численні вхідні сигнали формуються в вихідний, визначається певним алгоритмом обчислення.

Сама нейронна мережа являє собою безліч таких нейронів (процесорів) об'єднаних в єдину систему. Поодинці ці процесори досить прості, але будучи з'єднаними у велику систему, такі процесори здатні виконувати дуже складні завдання зі збору інформації, її аналізу і створення нової.

Головною перевагою нейронних мереж і їх відмінністю від інших математичних моделей є здатність до навчання.

Серед основних областей застосування нейронних мереж – прогнозування, прийняття рішень, розпізнавання образів, оптимізація, аналіз даних. Нейронні мережі лежать в основі більшості сучасних систем розпізнавання і синтезу мови, знаходження шляху, розпізнавання і обробки зображень [1].

1.1 Основні поняття роботи нейронних мереж

Штучний нейрон – це одиниця обробки інформації в нейронних мережах. На рисунку 1.2 уявлена модель нейрона, з яких складається нейронна мережа. Розподіляється вона на три основні елементи:

– набір синапсів (зв'язків), кожен з яких характеризується своєю силою або вагою. Синаптична вага штучного нейрона може приймати як негативні значення, так й позитивні;

– суматор, що є ядром нейрона. Суть суматора у складанні сигналів, що входять зважених щодо відповідних синапсів нейрона. На виході суматор визначає результат лінійної комбінації входів нейрона, коефіцієнтами якого є ваги зв'язків;

– функція активації (ФА), або функція стиснення, що використовується для обмеження вихідного сигналу нейрона. Зазвичай в результаті нормалізації значення виходу нейрона лежить в інтервалі [0;1] або [-1; 1].

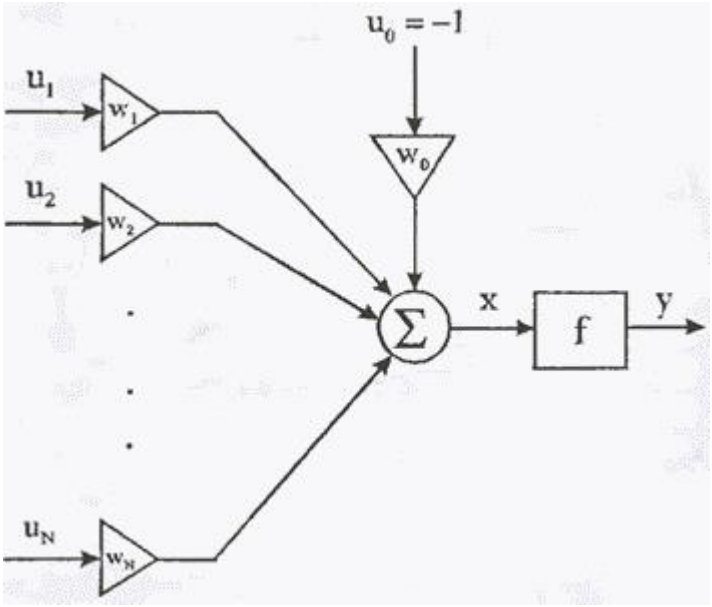

Рисунок 1.1 – Нелінійна модель нейрона

Модель нейрона, зображеного на рисунку 1.2, має вхідний сигнал фіксованої величини, що дорівнює одиниці, а також синаптична вага, використовувана в якості порогового елемента. В даному випадку значення ваги w_{k0} дозволяє зрушувати вихідне значення нейрона щодо початку координат, тобто фактично забезпечує ефект афінного перетворення.

Математичне уявлення функціонування нейрона k представлено у формулах:

$$v_k = \sum_{j=0} w_{kj} x_j, \tag{1.1}$$

$$y_k = \varphi(v_k)\,. \tag{1.2}$$

де $x_0, x_1, \ldots, x_m$ – вхідні сигнали;

$w_{k0}, w_{k1}, \ldots, w_{km}$ – синаптичні ваги нейрона k;

v_k – потенціал активації;

φ – функція активації;

y_k – вихідний сигнал нейрона.

Функція активації, що позначається φ(v), в залежності від потенціалу активації або індукованого локального поля визначає вихідний сигнал нейрона. Функції активації можна віднести до трьох основних типів:

– функція одиничного стрибка, або порогова функція. В цьому випадку нейрон буде приймати на виході нульове значення, якщо індуковане локальне поле цього нейрона не є позитивним числом і 1 в зворотному випадку;

– кусочно-лінійна функція. Таку функцію можна розглядати як апроксимацію нелінійного підсилювача;

– сигмоїдальна функція, графік якої нагадує латинську букву S (логістична функція). Цей тип активаційних функцій є найпоширенішим і

часто використовуваним, він підтримує баланс між лінійною і нелінійною поведінкою.

Область значень функцій активації, в більшості випадків, обмежена відрізком [0;1], але можливі ситуації, коли вихідне значення нейрона знаходиться в діапазоні [-1; 1]. У подібного роду випадках функція активації буде симетрична щодо точки початку координат [2].

На рисунку 1.2 наведено приклади активаційних функцій у графічному вигляді.

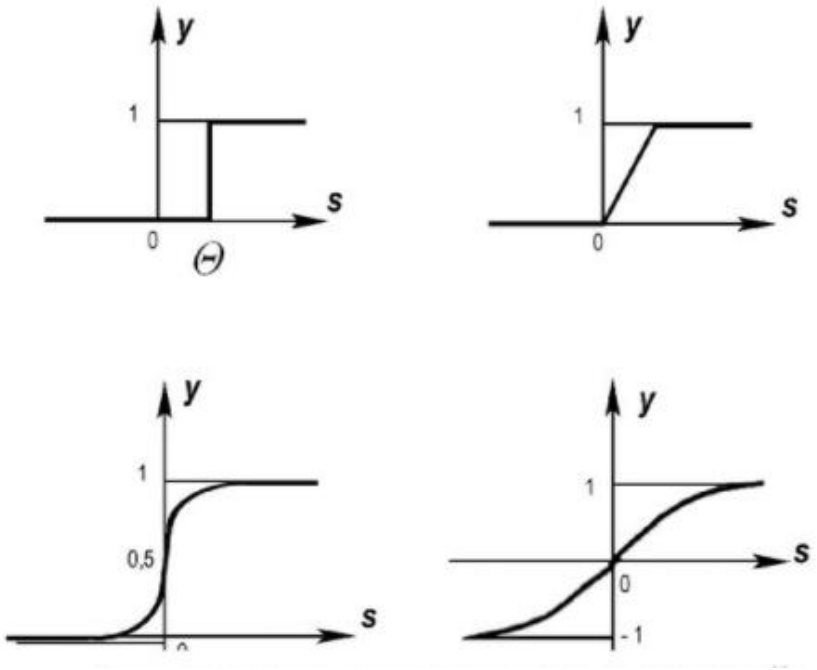

Рисунок 1.2 – Приклади активаційних функцій

1.2 Архітектура нейронних мереж

Серед штучних нейронних мереж глобально можна виділити три фундаментальних класи на які вони поділяються: одношарові мережі прямого поширення, багатошарові мережі прямого поширення і рекурентні мережі. У багатошарової нейронної мережі нейрони розташовуються по шарах, як це і випливає з назви. Якщо вхідна інформація всередині мережі

в послідовному режимі передається від перших шарів до наступних, то таку нейронну мережу називають ациклічною або мережею прямого поширення.

У найпростішому випадку в ШНМ існує перший вхідний шар нейронів, які не піддаються ніяким обчисленням і змінам, і другий вихідний шар, інформація за яким передається починаючи з вузлів джерела. На рисунку 3 зображена структура подібного типу мережі. Представлена ШНМ та інші мережі з аналогічною архітектурою називаються одношаровими персептронами.

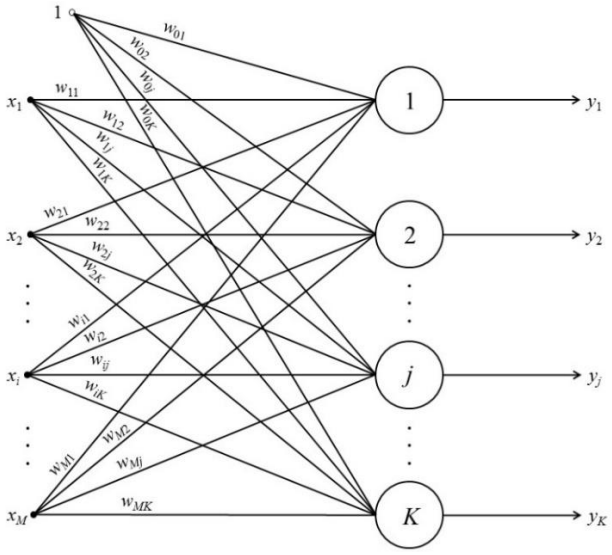

Рисунок 1.3 – Одношарова мережа прямого поширення

Одношаровий персептрон використовується в разі необхідності класифікації лінійно-розділюваних сигналів і також називається персептроном Розенблатта. Якщо ШНМ даного типу містить в собі тільки один нейрон, то така мережа здатна вирішувати завдання з поділом лише двох класів. Якщо збільшувати розмірності обчислювального шару НМ персептрона, включаючи в нього більшу кількість нейронів, то НМ зможе класифікувати більше двох класів.

Найчастіше, в якості активаційної функції для нейронних мереж типу одношарового персептрона, вибирають порогову функцію, що, фактично,

визначає лінійну поведінку нейрона. Однак вірним є твердження, що стійке прийняття рішення одношаровим персептроном не залежить від виду нелінійності нейрону. У зв'язку з цим, можна стверджувати, що даний вид ШНМ здатний виконувати класифікацію образів для лінійно-розділюваних класів.

Наступний клас нейронних мереж прямого поширення крім вхідного і вихідного шарів володіє прихованими шарами, одним або декількома. Вузли таких шарів називаються прихованими нейронами. Ускладнення архітектури штучних нейронних мереж прихованими шарами дозволяє виділити статистики високого порядку. Такі нейронні мережі називаються багатошаровими персептронами [3].

На рисунку 1.4 представлена структура багатошарової мережі прямого поширення з одним прихованим шаром. У вхідному шарі зображеної ШНМ розташовано три нейрона, в єдиному прихованому шарі – також три нейрона, а у вихідному – один. Така ШНМ називається мережею 3-3-1, а в приватному випадку мережу прямого поширення називають мережею m-h1-h2-q, де m – кількість входів, h1 – кількість нейронів першого рівня, h2 – кількість нейронів другого рівня і q – кількість виходів.

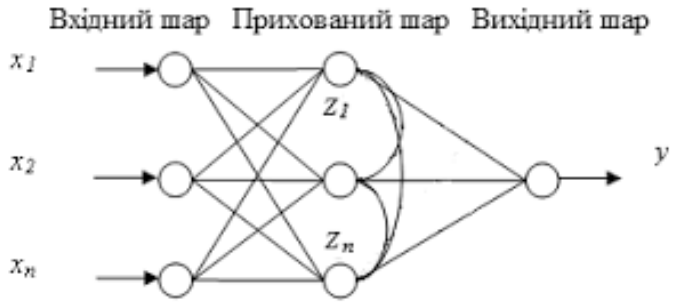

Рисунок 1.4 – Повнозв'язна мережа прямого поширення з одним прихованим шаром

Нейронна мережа, зображена на рисунку 1.4, є повнозв'язною, так як всі вузли конкретного шару пов'язані синапсами з усіма вузлами сусідніх шарів. Мережі, для яких дане правило не виконується, називаються неповнозв'язними.

Багатошарові персептрони мають кілька відмінних ознак:

– всі нейрони використовують нелінійну функцію активації. Необхідно, щоб ця функція була всюди диференційованої, тобто гладкою, на відміну від порогової функції, використовуваної в персептроні Розенблатта. У зв'язку з цими умовами, найчастіше в якості функції активації для багатошарового персептрона є сигмоїдальна. Наявність нелінійності грає дуже важливу роль, так як інакше таку ІНС можна буде привести до звичайного одношарового персептрону. Крім цього, використання логістичної функції обумовлено біологічними факторами, так як в ній враховується відновна фаза реального нейрона;

– така мережа завжди має, як мінімум, один прихований шар нейронів. На відміну від простих нейронних мереж, приховані нейрони дозволяють багатошаровим мережам вчитися вирішувати більше складні завдання, шляхом послідовного вилучення найбільш важливих ознак їх вхідного вектору;

– мережа має високий ступінь зв'язності, який гарантується великою кількістю синаптичних з'єднань. При зміні рівня зв'язності необхідно змінювати безлічі синапсів або їх вагових коефіцієнтів.

Комбінація всіх перерахованих вище якостей зі здатністю до самонавчання визначає обчислювальну потужність багатошарового персептрона. Однак, це також призводить до неповноти сучасних знань про точну природу поведінок таких мереж. Висока зв'язність і нелінійність істотно ускладнюють теоретичний аналіз багатошарового персептрона. А наявність в нейронних мережах прихованих шарів робить процес навчання складніше при необхідності візуалізацій.

У разі рекурентних нейронних мереж властиво наявність хоча б одного зворотного зв'язку. В цьому і полягає їх відмінність від мереж прямого поширення. На рисунку 1.5 зображена ШНМ з прихованим шаром, в якому один з нейронів направляє свій вихідний сигнал на перший вхід нейронної мережі.

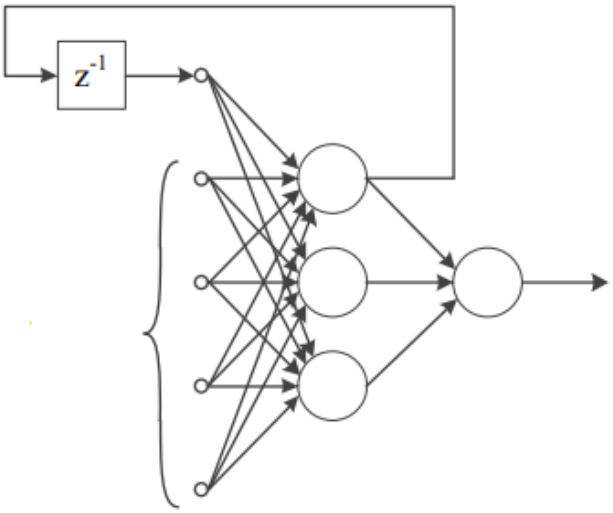

Рисунок 1.5 - Рекурентна мережа з прихованим шаром нейронів

Зворотні зв'язки істотно впливають на здатність мереж до навчання і на рівень їх продуктивності. За винятком цього, наявність зворотного зв'язку робить необхідним використання елементів одиничної затримки. Таке рішення призводить до нелінійної динамічної поведінки мережі, коли в мережі містяться нелінійні нейрони.

1.3 Принцип навчання штучних нейронних мереж.

Головна властивість штучних нейронних мереж – їх здатність навчатися на основі даних з навколишнього середовища і в результаті навчання підвищувати свою продуктивність. Процес навчання полягає в коригуванні синаптичними вагами нейронів.

Навчання – це такий процес, в якому вільні параметри штучної нейронної мережі налаштовуються в процесі моделювання навколишнього середовища, в яку вбудована ця нейронна мережа. Тип навчання визначається способом підстроювання цих параметрів.

Можна виділити три етапи процесу навчання нейронної мережі:

– в нейронну мережу з навколишнього середовища надходять стимули;

– в результаті надходження стимулів змінюються вільні параметри НМ;

– після зміни вільних параметрів зміниться внутрішня структура нейронної мережі і буде відповідати на збудження інакше.

Список зазначених вище процесів вирішує проблему навчання і називається алгоритмом навчання. Немає універсального алгоритму навчання для всіх існуючих архітектур нейронних мереж, а кожна одиниця з їх безлічі має власні переваги і недоліки.

Можна визначити п'ять основних видів навчання НМ:

– на основі корекції помилок;

– з використанням пам'яті;

– Хеббівське навчання;

– конкурентне навчання;

– метод Больцмана.

У першому випадку, навчання, засноване на корекції помилок, реалізує метод оптимальної фільтрації. Навчання на основі пам'яті передбачає явне використання навчальних даних. Метод Хебба і конкурентний підхід до навчання засновані на нейробіологічних принципах. В основу методу Больцмана покладені ідеї статичної механіки.

Виділяють дві парадигми навчання нейронних мереж: навчання з учителем і без вчителя.

Під навчанням з учителем можна розуміти існування знань про навколишнє середовище, які представлені у вигляді пар вхід-вихід. Однак,

саме середовище для нейронної мережі невідоме. На рисунку 1.6 показана блокова діаграма, що ілюструє цю форму навчання.

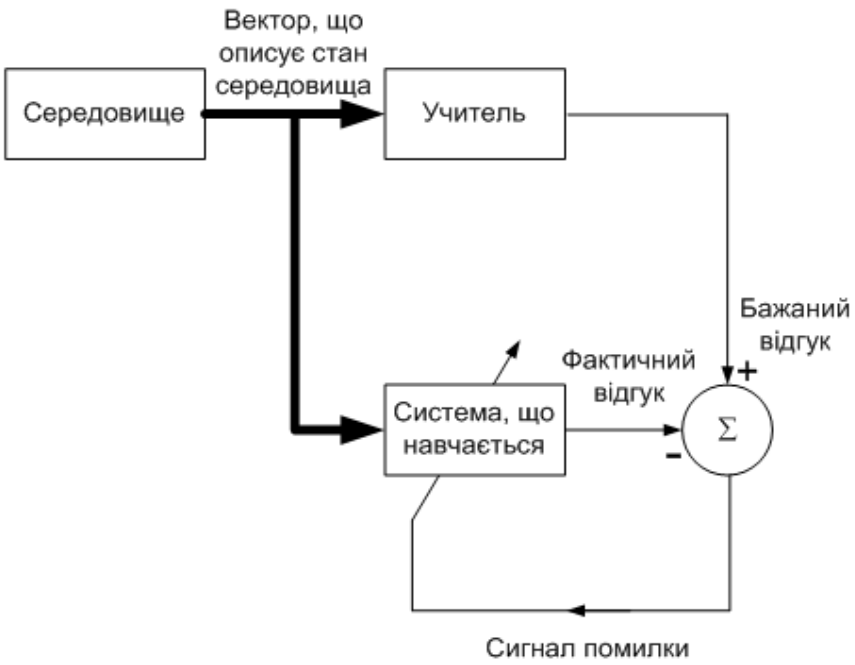

Рисунок 1.6 – Блокова діаграма навчання з учителем

Вчителю і нейронної мережі надходить навчальний вектор з навколишнього середовища. Учитель, ґрунтуючись на своїх знаннях, формує і передає ШНМ, що навчається бажаний відгук, який відповідає цьому вхідному вектору. Бажаний відгук являє собою оптимальні дії, які виконає нейронна мережа, що навчається. Параметри цієї мережі змінюються в залежності від сигналу з помилкою. Цей сигнал визначає різницю між бажаним результатом і фактичним відгуком ШНМ.

Навчання без вчителя не має на увазі присутність зовнішнього коректора, який керує процесом налаштування вагових коефіцієнтів нейронної мережі. При використанні цього підходу немає маркованих прикладів, на яких проводилося б навчання мережі. В цьому випадку є тільки незалежна від завдання міра якості уявлення, якої і повинна

навчитися штучна нейронна мережа, а вільні параметри цієї мережі будуть оптимізуватися по відношенню до міри якості [4].

На рисунку 1.7 зображена блокова діаграма навчання ШНМ без вчителя.

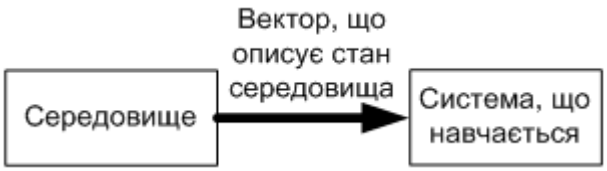

Рисунок 1.7 – Блокова діаграма навчання без вчителя

У даній роботі для навчання нейронної мережі буде використовуватися генетичний алгоритм, який відноситься до парадигми навчання з учителем.

1.4 Проблеми навчання нейронних мереж

Хоча основний алгоритм навчання багатошарових нейронних мереж відкритий вже давно і досить добре досліджений, навчання нейронних мереж на сьогоднішній день все ще більше нагадує «шаманство», ніж технологію.

Необхідно пробувати безліч різних архітектур нейронних мереж, проводити навчання кожної з них, а потім вибирати ту НМ, яка найкращим чином вирішує поставлене завдання.

Існують загальні рекомендації щодо вибору архітектури мережі для вирішення певного класу завдань. Також є формула для грубої оцінки кількості елементів у прихованому шарі за кількістю необхідного числа синаптичних ваг у багатошаровій мережі з сигмоїдальними передавальними функціями:

$$\frac{mN}{1+log_2 N} \leq L_w \leq m\left(\frac{N}{m}+1\right)(n+m+1)+m, \quad (1.3)$$

де n-розмірність вхідного сигналу;

m-розмірність вихідного сигналу;

N-число елементів навчальної вибірки [5].

$$L = \frac{L_w}{n+m}. \quad (1.4)$$

Незважаючи на це, проблема залишається, так як після вибору архітектури нейронної мережі її все одно необхідно буде навчати. За своєю суттю нейронна мережа є універсальним апроксиматором. У процесі налаштування мережа не займається обчисленням цільової функції, а лише підбирає внутрішній набір функцій. При додаванні цього набору функцій формується інша функція, яка на виході утворює ряд значень, що нагадує вихідний ряд, пред'явлений їй в процесі навчання (апроксимаційний поліном). Звідси можна зробити висновок про те, що вихідні дані нейронної мережі будуть містити помилку, величина якої заздалегідь не буде відомою. Відомо тільки те, що в процесі навчання дана помилка може бути зменшена до деякого оптимального рівня. Робоча точка ШНМ під час навчання ковзає по поверхні помилок прагнучі до глобального мінімуму цільової функції. В силу нерівності поверхні помилок, НМ має шанс застрягти в межах локального мінімуму далеко від очікуваного глобального. Також, якщо схил локального мінімуму досить крутий і крок навчання надто малий, щоб робоча точка зрушилася на його край, настає стан паралічу мережі. У цьому стані НМ на навчальній вибірці дає неточні результати, але навчання все одно зупиняється.

Також можливий стан, при якому помилка навчання безладно коливається (осцилює) і мережа переходить в стан перенавчання, як показано на рисунку 1.8. Це відповідає надмірно точної апроксимації даних, на яких навчається НМ [6].

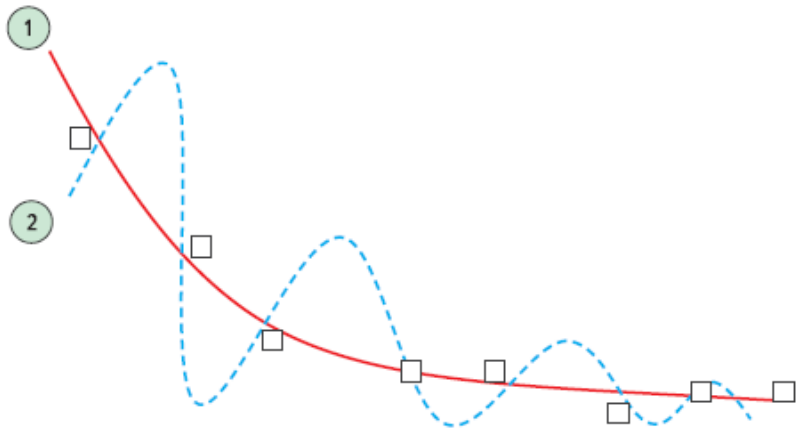

Рисунок 1.8 – Стан перенавчання: 1 – справжня залежність; 2 – занадто точна апроксимація поліномом

Існують труднощі, пов'язані з внутрішньою організацією вхідних даних. Це так зване "прокляття розмірності". Зазвичай дані на вході містять в собі шум і деяке число малозначущої інформації. Якщо спробувати змусити НМ розібрати дані по кластерах без попередньої підготовки вхідних даних, є ризик отримати неправильні Узагальнення. Однак, в разі, коли внутрішня структура даних спочатку невідома необхідно збільшувати число вхідних елементів мережі. Достатня кількість вхідних нейронів стрімко зростає зі збільшенням розмірності простору, яке кластеризується (приблизно як 2n). Слідом за підвищенням кількості нейронів на вході, також змінюється число нейронів прихованого шару. У зворотному випадку процес кластеризації може втрачати корисну інформацію. Це ускладнює процес навчання нейронної мережі і робить його менш передбачуваним.

Таким чином, для успішного вирішення завдання навчання нейронної мережі необхідно слідувати наступним пунктам:

– правильно визначити структуру НМ (кількість шарів і нейронів в кожному шарі). Обрана структура буде відповідати потребам розв'язуваної задачі;

– правильно визначити наступні параметри навчання НМ: норму навчання, кількість навчальних прикладів, сам алгоритм навчання і крок навчання;

– попередньо обробити дані НМ. А саме, визначити структуру вхідних даних, відфільтрувати шум, по можливості видалити малозначні складові вхідні дані. Іноді попередня обробка може вимагати лінійне перетворення, що виділяє із загальної кількості вхідних даних Набір найбільш цінних напрямків або векторів (метод головних компонент).

Навчання штучних нейронних мереж в даний час є складним завданням, тому глобальне застосування технологій, які засновані на нейронних мережах малоймовірно на сьогоднішній день.

1.5 Аналіз предметної області та постановка задачі дослідження

Незважаючи на складнощі коректного навчання нейронних мереж, їм знайшли застосування в таких сферах, як: прогнозування, прийняття рішень, розпізнавання образів, оптимізація, аналіз даних, обхід перешкод, пошук шляху.

У даній роботі буде вирішуватися задача обходу перешкод. На сьогоднішній день існує багато методів вирішення цього завдання, що відрізняються своєю структурою, використовуваними алгоритмами і багатьом іншим.

Об’єкт дослідження – процес створення маршруту з обходженням перешкод в умовах статичного навколишнього середовища.

Предмет дослідження – методи вирішення задачі обходу перешкод, в особливості за допомогою багатошарової нейронної мережі на основі генетичного алгоритму.

Метою роботи є створення багатошарової нейронної мережі з використанням генетичного (еволюційного) алгоритму навчання.

Методи дослідження – теорія штучних нейронний мереж, теоретичні аспекти підходів до вирішення задачі пошуку маршруту, імітаційне моделювання.

Генетичний алгоритм здатний навчити нейронну мережу коректно виконувати поставлене завдання. Алгоритму дається проста фітнес-функція, яка визначає, наскільки добре працює мережа. Недоліком алгоритму є те, що при роботі з великою мережею навчання може займати багато часу і бути повільніше алгоритму зворотного поширення. Однак, володіючи великим обсягом обчислювальної потужності, генетичний алгоритм здатний дати кращі результати, ніж зворотне поширення, оскільки вони майже завжди будуть в змозі досягти глобального мінімуму.

Ідея генетичного алгоритму заснована на теорії дарвінізму, за прикладом того, як відбувається біологічна еволюція. Однак, в роботі буде реалізовуватися спрощена модель, до якої застосовні ті ж еволюційні концепції. Суть алгоритму полягає в знаходженні правильної комбінації шляхом перебору і подальшого відбору.

Ділиться алгоритм на три етапи: схрещування, селекція (відбір) і формування нового покоління. Потім, якщо результат не задовільний кроки повторюються до тих пір, поки не відбудеться одна з наступних умов:

– результат буде задовільним;

– кількість поколінь (циклів) досягне заздалегідь визначеного максимуму;

– час на мутацію буде вичерпано.

Проведені в даній магістерській роботі дослідження показують, що еволюційний підхід є досить ефективним в рамках завдання обходження перешкод і показує хороші результати. Однак, алгоритм вимагає великий обсяг обчислювальних ресурсів і може витрачати багато часу при роботі з великою нейронною мережею.

2 МЕТОДИ ОБХОДУ ПЕРЕШКОД І ПОШУКУ МАРШРУТУ.

Планування маршруту є пріоритетним завданням в області навігації. Включає в себе таке завдання три основні аспекти:

– маршрут повинен пролягати від початкової точки до кінцевої;

– маршрут повинен планувати рух з обходом перешкод;

– маршрут повинен бути оптимальним.

Методи планування маршруту зазвичай класифікують за різними ознаками. У сфері використання інтелектуальних технологій можна виділити два основних типи методів планування маршруту: традиційні методи і евристичні. За характеристиками навколишнього середовища ці методи можна розділити на методи планування в статичному навколишньому середовищі і в динамічному. Однак, статичне середовище нечасто може зустрітися на практиці. Також методи піддаються класифікації за кількістю інформації про навколишнє середовище: методи з повною інформацією, при яких вирішується глобальне планування маршруту і методи з неповною інформацією, в разі аналізу перешкод поблизу від об'єкта навігації. Неповна інформація про навколишнє середовище також може бути в разі динамічно змінюваної обстановки.

В історії було запропоновано велику кількість методів планування маршруту і обходу перешкод, в яких застосовуються різні евристичні прийоми, що визначаються змістовним змістом розв'язуваної задачі. У даній роботі розглядаються основні підходи до вирішення завдання обходу перешкод.

На рисунку 2.1 зображена класифікація зазначених підходів.

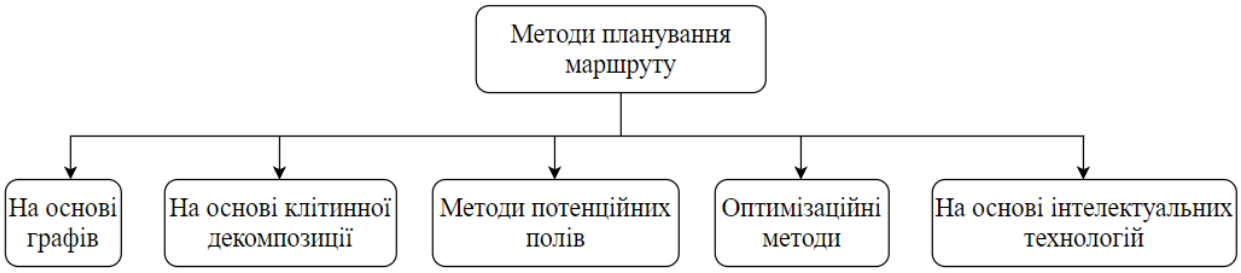

Рисунок 2.1 – Класифікація методів планування маршруту

2.1 Методи на основі графів

Граф (або дерево) відображає всі стани, в яких може перебувати об'єкт: кожен вузол графа являє собою конкретний стан об'єкта. Станом об'єкта може бути положення в просторі, кут напрямку, швидкість або прискорення об'єкта. Функція витрат характеризує переходи між цими станами. Це дозволяє виділити шлях, який має мінімальну загальну вартість досягнення цільового стану. Основні методи цього класу показані на рисунку 2.2.

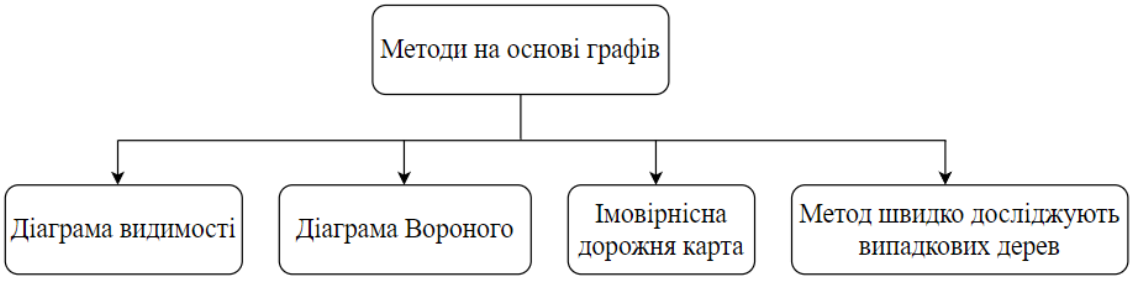

Рисунок 2.2 – Методи на основі графів

Пропонується набір методів, що відрізняються вибором вузлів відповідного графа. Такі методи часто використовуються в тих випадках, якщо дані про навколишнє середовище є незмінними (статичними) і заздалегідь відомі. Граф являє собою деяку непорожню множину V вершин.

Конкретні пари цих вершин з'єднані ребрами або дугами. На дугах задається один напрямок з двох можливих, граф з такими дугами називається орієнтованим графом (орграфом). На відміну від дуг, на ребрах напрямок не виділено і граф з ребрами називають неорієнтованим. Граф зображений на рисунку 2.3 і позначається, як *G = (V, E).*

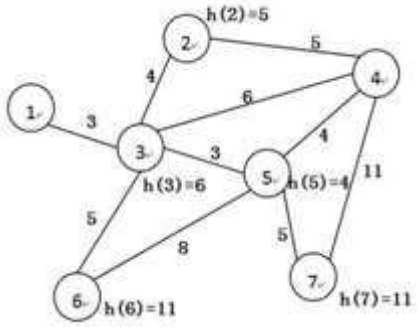

Рисунок 2.3 – Зважений граф G

Побудова графа має зв'язок з порядком вибору для нього опорних точок. Наведено в приклад метод на основі діаграми видимості. Визначимо діаграму у вигляді неорієнтованого графа $Vg(N, L)$, у якому $N = V \cup \{S_0, S_f\}$, що є непорожнім безліччю вузлів, яке включає в себе безліч вершин визначальних перешкоди, початкову точку S_0 і кінцеву точку S_f планованого маршруту, а L – множина ребер, які визначаються всіма парами точок в множені N. Ці точки об'єднуються відрізком прямої, яка не перетинається з перешкодами. Щоб використовувати діаграму видимості необхідно зрадити перешкодам форму багатокутника (в двовимірному випадку) або багатогранника (в тривимірному випадку). Так як маршрут будується по вершинах перешкод, а частина шляху збігається з краями безлічі перешкод, через що існує ймовірність зіткнення з цими перешкодами. Також, при збільшенні кількості перешкод буде здійснюватися швидке зростання складності графа. З чого можна зробити висновок, що найбільш пріоритетним завданням в цьому випадку буде

скорочення складності графа видимості. Можна виділити метод динамічної діаграми видимості і концепцію m-вектору [7]. Ці методи призначені для аналізу перешкод, щоб зрозуміти які перешкоди слід враховувати, а які ні. Однак, ці методи можуть привести до ситуації, в якій отриманий шлях не буде оптимальним, особливо в разі тривимірного простору.

Інший підхід розглядається в методі структурування вільного простору і методі на основі діаграми Вороного. Метод структурування вільного простору, який реалізований в плоскому випадку визначає вільні ланки-відрізки, які з'єднують вершини безлічі перешкод і не перетинаються з ними. Вільні ланки утворюють опуклі багатогранники, які описують всі області вільного простору. Такі дані будуть оформлятися у вигляді спеціального графа, який називається MAKLING, а спланований маршрут з обходом безлічі перешкод будується за допомогою алгоритмів пошуку шляхів в графі [9].

Використання методу на основі діаграм Вороного також, як і метод структурування вільного простору, призводить до утворення допустимого шляху, який максимально віддалений від перешкод і забезпечує додаткові можливості з боку позиціонування об'єкта. Обидва ці методи забезпечують найбільш безпечний шлях з боку можливих зіткнень з безліччю перешкод, проте загальна довжина маршруту збільшується [10].

Опорні (шляхові) точки можуть бути обрані абсолютно випадковим чином у вільній частині простору. Подібний спосіб призвів до утворення нового класу методів – імовірнісних. Методу полягає у випадковому виборі точок, які розподілені з якоюсь ймовірністю в просторі. Розподіл буде вибиратися по всьому простору середовища, враховуючи перешкоди. Коли вибирається точка потрапила на одну з перешкод, вона відкидається. Але випадковий вибір точок маршруту може призвести до того, що створений маршрут не буде оптимальним [11].

Побудова графа полягає у виборі ребер, що з'єднують шляхові точки. Ребро з'єднує дві такі точки, в разі, коли з однієї в іншу можна перемістити

по прямолінійному шляху, з обходом перешкод. Щоб побудувати всі ребра графа потрібно переглянути кожну пару шляхових точок. Ця процедура є трудомісткою, тому використовується спрощений варіант – локальний. В цьому випадку для кожної з точок в якості пари розглядаються тільки ті точки, які розташовані найближче. Побудова дорожньої карти перетворює завдання планування маршруту в завдання пошуку шляху в графі.

Метод імовірнісної дорожньої карти досліджувався за допомогою моделювання [12]. Відзначимо, що, оскільки метод сильно залежить від повноти інформації про навколишнє середовище, його важко застосовувати в умовах динамічного середовища. На основі оригінального методу імовірнісної дорожньої карти розроблено ряд модифікацій , що дозволяють покращувати результати в різних ситуаціях: підвищувати ефективність обчислення, отримувати коротший шлях і т. п.

До методів на основі випадкового вибору відноситься метод швидко досліджують випадкових дерев. При використанні цього методу процес генерації маршруту полягає в побудові дерева, що складається з опорних точок. Дерево послідовно розширюється, починаючи від початкової точки і закінчуючи цільовою. Процес починається зі стартового дерева X_{tree} з початковою вершиною і без будь-яких ребер. З кожним кроком побудови дерева вибирається нова випадкова точка маршруту X_{rand}. Генеруються такі точки до моменту, коли чергова випадкова точка не буде знаходиться поза перешкод. У поточному дереві X_{tree} визначається оптимальна по відношенню до X_{rand} точка, частіше найближча, з якої можна буде перейти в точку X_{rand}. У разі, коли така точка не буде знайдена, випадкова точка X_{rand} буде відкидатися алгоритмом. Коли оптимальна точка визначена, точка X_{rand} додається до дерева і з'єднується ребром з обраної оптимальної точкою. Процес побудови дерева закінчується в разі, коли з останньої доданої точки можна перейти до кінцевої або в тому випадку, коли досягнуто граничну кількість вершин дерева. Якщо цільова точка досягнута,

шлях відновлюється зворотним ходом від чергової точки до тієї, з якою поточна точка була з'єднана при додаванні, тобто за допомогою відносини підпорядкованості, що виникає в процесі розширення графа [13].

Метод RRT дозволяє генерувати маршрут як у фізичному, так і в конфігураційному (фазовому) просторі, яке може мати високу розмірність. Крім цього, метод може працювати в навколишньому середовищі з безліччю перешкод. Метод допускає різні модифікації, які пов'язані з інтерпретацією поняття досяжності, а також з правилами за якими вибирається чергова точка для включення в дерево. Існують варіанти алгоритму, при яких під час вибору точки береться до уваги положення кінцевої точки. Дерево може будуватися у напрямку від кінцевої точки до початкової, можливий також варіант методу в обидві сторони [14].

Методи на основі графів або дерев краще використовувати в умовах статичного простору, так як побудова графа має на увазі наявність повної інформації про цей простір. Присутність мінливих перешкод, наприклад, другого об'єкта рухається по графу, в цих методах складніше піддається інтерпретації і не дозволяє коректно будувати граф.

2.2 Методи на основі клітинної декомпозиції

З найбільш ймовірних методів, що виникають в умовах завдання планування маршруту в просторі з перешкодами можна виділити дискретизацію навколишнього середовища. У можливих реалізаціях цього методу можна виділити дві основні групи: наближена і точна клітинні декомпозиції [15]. Наближена клітинна декомпозиція створюється при використанні сітки, яка покриває собою весь простір. Метод сіток ділить навколишнє середовище з перешкодами на клітини однакового розміру, кожна з цих клітин приймає значення 0, якщо немає перешкод, або 1, в разі коли перешкоди є. Сітка проста у використанні, і її легко створити і підтримувати. Однак можна виділити головний їх недолік: при зменшенні

кроків сітки буде збільшуватися її трудомісткість. Таке збільшення буде більш помітним в просторі великого обсягу. Існують різні підходи, засновані на нерегулярних сітках, покликаних знизити обсяг всіх обчислень. Наприклад, дерево квадрантів у двовимірному середовищі або дерево кубів – у тривимірному [16].

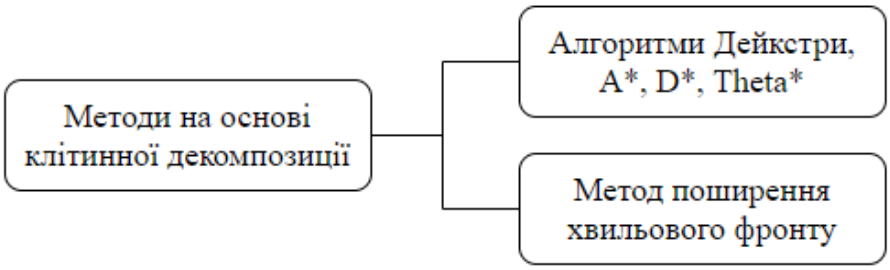

Рисунок 2.4 – Методи на основі клітинної декомпозиції

В умовах точної клітинної декомпозиції простір ділиться на клітини, використовуючи межі перешкод. Тут важливу роль відіграє завдання розкладання багатокутника на опуклі області. Відомі такі підходи до точної клітинної декомпозиції, як вертикальне або трапецієподібне розкладання [17].

Найчастіше, при вирішенні завдання планування маршруту за методами на основі клітинної декомпозиції або на основі графів є необхідність в додатковому етапі генерації маршруту, який полягає в пошуку шляхів на графі. Можна використовувати різні стратегії в задачі пошуку найбільш оптимального рішення. Ці стратегії діляться на два види: класичні (неінформовані) і евристичні (інформовані).

До класичних стратегій пошуку маршруту в навколишньому середовищі відносяться: пошук за критерієм вартості, пошук в ширину, пошук в глибину, пошук з обмеженням глибини, пошук в глибину з ітеративним поглибленням і двонаправлений пошук [18].

Евристичні стратегії пошуку маршруту не мають суворого обґрунтування, проте дають оптимальне рішення задачі в більшості випадків на практиці. З особливостей пошуку з використанням евристичної стратегії можна виділити евристичні правила, які формуються в умовах конкретного завдання. Використання правил може значно підвищити ефективність алгоритму в порівнянні з відповідною класичною стратегією. Один з поширених прикладів евристичної стратегії-алгоритм A*. Цей алгоритм є розширенням для алгоритму Дейкстри. Алгоритм A* може досягти більшої продуктивності за допомогою евристики за часом. Цільову функцію алгоритму можна представити у вигляді суми двох інших функцій: функції вартості досягнення розглянутої точки з початкової та евристичної функції оцінки відстані від розглянутої точки до кінцевої точки. Евристична функція не повинна переоцінювати відстань до цільової точки. Без другої складової алгоритм A* зводиться до алгоритму Дейкстри.

З недоліків такого алгоритму можна виділити обмеженість кількості варіантів напрямку через структуру сітки. Існують різні підходи покликані обійти цей недолік. Найбільш цінними можна виділити алгоритми D* [19] і Theta* [20]. Алгоритм D*, або динамічний алгоритм A*, являє собою розвиток алгоритму A* за допомогою перепланування. У цьому випадку використовується менша кількість клітин, ніж у випадку алгоритму A*. Це обумовлено тим, що в цьому алгоритмі не планується повний маршрут до цільової точки. У випадку D* [22] алгоритм покращений завдяки найбільш ефективним розширенням алгоритму і скороченням безлічі розширених вузлів, також D* вимагає менше часу для обчислень. Крім цих випадків, динамічне перепланування використовується також в алгоритмах D* Lite і Lifelong Planning A* (LPA*) [23]. Інший алгоритм (кінематичний A*) створений для визначення допустимого маршруту, що відповідає вимогам динамічних характеристик ЛА [24].

Іншим підходом, що використовує сітки, є метод поширення хвильового фронту, окремим випадком такого поширення є метод Fast

Marching Method (FMM) [25]. Головна особливість підходу – відсутність етапу пошуку.

Суть методу поширення хвильового фронту можна описати на прикладі розширення фронту хвилі: якщо камінь кинутий в ставок, то виникає хвиля в формі кола, далі ця хвиля буде розширюватися місця падіння каменю. Варто звернути увагу на те, що швидкість поширення хвилі буде однаковою у всіх напрямках. Хвильовий фронт буде круговим. Однак у разі неоднорідності простору хвиля буде розширюватися з різною швидкістю і хвильовий фронт буде іншим.

2.3 Методи потенційних полів

З найпоширеніших підходів до обчислення траєкторій-використання потенційних векторних полів. Головною ідеєю таких методів є рух уздовж векторних ліній векторного поля. Потенційна функція цього поля крім мети руху, буде відображати конфігурацію перешкод і їх форм. Цей підхід підходить як у двовимірному середовищі, так і в тривимірному. За типом потенційної функції можна розділити: віртуальне силове поле [26], Ньютонівське потенційне поле [27]; супербіквадратне потенційне поле, гармонійне векторне поле [28].

Серед методів, заснованих на потенційних полях, найбільш відомим є метод штучних потенціалів (Artificial Potential Field, APF). Алгоритм цього методу має низьку обчислювальної складністю, але високою ефективністю реалізації. У цьому алгоритмі векторне поле буде розділятися на дві складові: об'єкт руху представляється притягує векторним полем, в перешкоди – відразливим. При додаванні двох таких векторних полів вирішуються завдання рух до заданої цільової точки і обхід перешкод. Варто відзначити, що відразливе векторне поле є сумою складових, кожне з яких описує конкретну перешкоду.

Розглядаємо об'єкт як матеріальну точку. Потенційна функція $U(q)$ можна записати у вигляді:

$$U(q) = U_{att}(q) + \sum U_{rep,i}(q), \tag{2.1}$$

де $U_{att}(q)$ – притягуюча потенційна функція, задана для точки q;

$\sum U_{rep,i}(q)$ – складові відразливою потенційної функції, відповідні окремим перешкодам.

Притягуюча потенційна функція має вигляд:

$$U_{att}(q) = \frac{1}{2} k_p \left(q - q_g\right)^2, \tag{2.2}$$

де k_p – коефіцієнт тяжіння, і породжує векторне поле антиградієнта.

$$F_{att}(q) = -\nabla U_{att}(q) = -k_p\left(q - q_g\right), \tag{2.3}$$

Для кожної і-го перешкоди використовуємо потенційну функцію:

$$U_{rep,i}(q) = \begin{cases} \frac{1}{2} k_{ri} \left(\frac{1}{\rho_i} - \frac{1}{p_{0i}}\right)^2 \left(q - q_g\right)^n, \\ 0 \end{cases} \tag{2.4}$$

де k_{ri} – коефіцієнт відштовхування;

$\rho_i = |q - q_i|$, q_i – центр перешкоди;

p_{0i} – радіус кругової області впливу перешкоди. Ця функція породжує векторне поле градієнта.

У цьому підході навколишній простір розглядається як статичний, незмінний. Однак в реальних умовах простір частіше буде динамічним, змінюваним. В якості модифікації методу штучних потенціалів для використання в динамічно змінюваному просторі використовуються

відносна швидкість і відносне положення досліджуваного об'єкта по відношенню до цільової точки в притягуючій потенційної функції, а також відносне положення і відносна швидкість досліджуваного об'єкта по відношенню до перешкоди у відштовхуючій потенційної функції.

Використання методу штучного потенціалу для планування шляху є простим і спрощує управління процесом руху в реальному часі. Однак головний недолік цього підходу – існування локальних мінімумів. Були запропоновані деякі рішення цього недоліку, але не існує повністю задовільного рішення. Також було зазначено, що існує невизначеність у побудові потенційних функцій, пов'язаних з вибором коефіцієнтів функцій. Два з цих факторів обмежують широке застосування методу штучного потенціалу при вирішенні практичних завдань.

Іншим недоліком методу штучних потенціалів є проблема тремтіння [29]. Існують рішення для подолання цього недоліку, наприклад, метод віртуального силового поля (Virtual Force Field, VFF), який поєднує в собі метод сіток і метод потенційного поля, який може застосовуватися для локального планування маршруту.

Метод віртуального силового поля має свої недоліки. Через значні зміни характеристик відштовхування перешкод при відборі проб не дозволяє планувати проходження вузьких ділянок типу воріт.

Метод також призводить до коливального руху в смугоподібній області (наприклад, у коридорі). Для подолання цих недоліків існує метод гістограми векторного поля (Vector Field Histogram, VFH).

Метод VFH - один з найпопулярніших методів локального планування шляху в режимах управління в реальному часі в мобільній робототехніці. При такому підході перешкоди обходяться в три етапи. На першому етапі створюється почесна гістограма, яка описує перешкоди навколо об'єкта. На другому створюється одномірна гістограма полярності на основі четвертої гістограми. Нарешті, на третьому етапі вибирається сектор, який найкраще підходить для більш низької щільності бар'єру, і розрахуйте кут повороту в

цьому напрямку. На основі методу VFH створені вдосконалені методи: VFH+ [30] и VFH* [31]. Метод VFH+ враховує розмір об'єкта, обмеження динаміки і відповідне розширення розміру перешкоди. У методі VFH* оптимальний напрямок руху вибирається за допомогою алгоритму A* з урахуванням глобальної екологічної інформації.

2.4 Оптимізаційні методи

Завдання планування маршруту в складних середовищах може бути вирішене як завдання оптимізації. Для цього необхідно відобразити переміщення об'єктів у кадрі певної моделі як динамічну систему. Перешкоди будуть описані з обмеженнями, а якість дозволеної траєкторії має оцінюватися з деякою функціональністю. Результат – оптимальне завдання управління, яке не тільки гарантує траєкторію об'єкта в обхід перешкоди, але і дозволяє вибрати найкращий в деякому сенсі варіант, по швидкості, енергоефективності тощо.

Слід зазначити, що оптимальне планування траєкторії є іншим типом завдання, ніж просте планування. Перше рішення займає значно більше часу, ніж друге. Класичний оптимальний метод управління заснований на нетривіальних аналітичних обчисленнях. Тому оптимальні засновані на управлінні способи ефективні для простих, особливо лінійних систем, і більш складні для складних нелінійних систем. Тому зусилля дослідників поступово перейшли до чисельних методів вирішення завдань оптимізації. Цей чисельний метод можна розділити на прямий і непрямий [32].

Як відомо, для оптимальних завдань управління існують необхідні умови для екстремальних значень, які встановлюються в рамках принципу максимуму Понтрягіна. Використовуючи необхідні умови, оптимальне завдання управління зводиться до вирішення диференційних рівнянь. Такі проблеми можна вирішити чисельно. Відповідний метод називається непрямим методом, оскільки він не є методом пошуку мінімального

значення для функції прямого пошуку. Звернемо увагу на те, що аналітичні рішення відповідних крайових задач часто неможливі, а численні методи чутливі до аналогічних даних або мають низьку схожість.

Прямий підхід до завдання оптимального управління пов'язаний з прямим пошуком мінімального значення цільової функції. Однак вирішення такої проблеми полягає в тому, що деякі функції можуть бути знайдені тільки чисельно у вигляді деякої дискретної аналогії. Необхідно звести вихідне завдання до завдання набору кінцевих змінних параметрів, іншими словами, до завдання нелінійного планування. Для вирішення завдання нелінійного програмування пропонується ряд численних методів. У результаті будь-який прямий метод включає в себе дві основні підзадачі: дискретизацію, спрощення завдання до завдання нелінійного програмування та вирішення цього завдання.

Загалом, оптимальні проблеми управління можуть бути сформульовані наступним чином. Потрібно знайти рішення *x(t), u(t)* динамічної системи:

$$x = f(x, u, t), \tag{2.5}$$

яка задовольняє обмеженням:

– $l_i \leq \Psi_i(x(t_0), u(t_0), t_0) \leq u_i$ (обмеження у початковій точці);

– $l_t \leq \Psi_t(x(t_t), u(t_t), t_t) \leq u_t$ (траєкторні обмеження);

– $l_f \leq \Psi_f\left(x\left(t_f\right), u\left(t_f\right), t_f\right) \leq u_f$ **(**конечні обмеження).

Опишемо кілька способів вирішення описаної оптимальної задачі.

Плоска модель. Поняття диференційної площини (планарності) нелінійних було запропоновано М. Фліссом у 1990-х роках [33]. Вона включає в себе концепцію плоского виведення, що дозволяє представляти динамічну систему у вигляді набору функціональних взаємозв'язків між входом і виходом. У разі пласкої динамічної системи завдання

оптимального управління може бути перетворено на завдання нелінійного програмування шляхом параметризації плоского вихідного сигналу. Цей метод зменшує розмір завдання, оскільки розмір вихідного простору параметричної площини менший за розмір простору стану. Друга перевага плоского підходу полягає в тому, що він дозволяє використовувати прості алгоритми для обчислення керуючих функцій, що реалізують траєкторії.

Оптимальна траєкторія може бути побудована в три етапи з використанням диференційної площини:

– визначення плоского виходу. Не існує спільного методу вибору плоского виходу. При вирішенні практичної задачі необхідно побудувати плоский вихід на основі природи розглянутої системи та її значення;

– налаштування вихідних даних площини. Вибрана площина повинна мати тимчасову функцію. Така функція визначає сам вихід, стан системи і введення (управління) у часі. Якщо ви бачите, як лінійна комбінація базисних функцій з невідомими коефіцієнтами, ви можете скористатися базовими функціями для вибору тимчасових функцій:

$$z_i(t) = \sum_j A_{ij}\lambda_j(t). \qquad (2.6)$$

Тут $\lambda_j(t), i = 1, \dots, N$ вибрані базисні функції, а A_{ij} – невідомі коефіцієнти, які необхідно вибрати, враховуючи існуючі обмеження та вимоги до оптимальної траєкторії щодо зазначеної функції маси. Базисні функції можуть бути обрані з різних причин (наприклад, простота представлення вихідних даних). Можливі параметри базових функцій: поліноми, тригонометричні поліноми, сплайні;

– побудова графіка траєкторії. Параметризація виходу перетворює задачу вибору оптимальної траєкторії на завдання нелінійного

програмування, де коефіцієнти лінійної комбінації заданої планарної вихідної функції невідомі.

Тому параметризація вихідних даних площини зводить завдання оптимального управління (планування траєкторії) до завдання нелінійного планування. Для вирішення цієї проблеми можна використовувати стандартні методи, реалізовані в програмних комплексах, таких як SNOPT [34], NPSOL [35].

Двоїчно-цілочисельне програмування. Один з підходів до вирішення оптимізаційних завдань – зведення їх до завдань двомісячного програмування, що є приватним випадком завдань цілочисельного програмування, в яких змінні можуть приймати лише два значення 0 і 1. Методи вирішення подібних завдань добре розроблені, наприклад, можуть бути використані добре відомі метод гілок і меж і метод відсікання Гоморі.

Двійкове цілочисельне програмування. Одним із способів вирішення завдань оптимізації є скорочення їх до двійкових цілочисельних завдань програмування, що є приватним випадком цілочисельних завдань, де змінні можуть приймати тільки два значення 0 і 1. Вирішення таких завдань були добре розроблені, як, наприклад, добре відомі метод гілок і кордонів і метод відсікання Гоморі.

Завдання двійкового цілочисельного планування можуть виникати в методах клітинної декомпозиції. Наприклад, використовують триангуляцію Делоне як одиничну декомпозицію для планування плоского руху. Кожному елементу секціонування (трикутнику) потім призначається змінна з значенням 0 (маршрут не проходить через трикутник) або 1 (маршрут проходить через трикутник). Як рішення вибирається послідовність суміжних трикутників, починаючи з початкової точки і закінчуючи кінцевою. Умова суміжності трикутника формується у вигляді лінійної нерівності за допомогою матриці суміжності трикутника.

Оптимізація забезпечується відповідною цільовою функцією. Метод близький до стандартного методу клітинної декомпозиції, оскільки опис

суміжних трикутників фактично зводиться до побудови спеціального графа і пошуку маршруту в ньому, але пошук здійснюється не стандартним методом графа, а як вирішення завдання цілочисельного програмування. Після розміщення ланцюжка суміжних трикутників вибирається траєкторія, що з'єднує медіанні перетини цих трикутників. Слід зазначити, що такі шляхи гарантовано відокремлені від перешкод і зручні для згладжування та використання з урахуванням відхилень.

Спосіб зведення завдання оптимізації до завдання двійкового цілочисельного планування може бути використано у тривимірному випадку. Також важливо відзначити, що виникаюча може бути переформульовано як завдання з обмеженнями у вигляді лінійних матричних нерівностей. Для вирішення цих завдань пропонуються ефективні алгоритми, які доводять, що вони зручніші у формуванні завдань, ніж цілочисельні алгоритми програмування, доводячи при цьому, що вони швидше [36].

2.5 Методи на інтелектуальних алгоритмах.

Завдання планування шляхів є підзавданням автоматичного управління об'єктами. Об'єкт повинен мати здатність вирішувати проблему планування маршрутів в реальних умовах навколишнього середовища без втручання людини. Природно, в цьому випадку використовуються поведінкові алгоритми, знайдені в живій природі, – так звані біоінспіровані алгоритми. Ці алгоритми імітують поведінку або мислення людини, а також деяких біологічних спільнот.

Серед біоінспірованих алгоритмів можна виділити: інтелект рою, алгоритм мураха, метод рою частинок, бджолиний алгоритм і алгоритм генетичної еволюції. Зазвичай ці алгоритми виражаються як алгоритми мінімізації (або пошуку), які пов'язані з евристичними алгоритмами, тому

що вони не мають суворого математичного доказу. Розгляньмо деякі алгоритми, засновані на інтелектуальних алгоритмах.

Мурашиний алгоритм. Перший варіант алгоритму, названий алгоритмом оптимізації колонії мурашок (Ant Colony Optimization, ACO), був запропонований Марко Доріго в 1992 році для пошуку оптимального шляху в графі [37].

У реальному світі мурахи (спочатку) пересуваються у випадковому порядку до того, як знайдуть їжу, після чого повертаються у свої колонії, маркуючи шлях феромонами. Якщо інші мурахи знайдуть такі сліди, вони, швидше за все, підуть за ними. Якщо ці мурахи зрештою знаходять джерело живлення, то, повертаючись, вони повторно маркують шлях. З часом феромони починають випаровуватися, знижуючи їх привабливість. Чим довше час до і від мети, тим більше феромонів випаровується. У короткостроковій перспективі алгоритм буде швидшим, і отримана сила феромона залишиться високою [38].

Мурашині алгоритми фокусуються на вирішенні трудомістких комбінаторних завдань, використовуючи такі алгоритми для пошуку майже оптимальних рішень за прийнятний час. Таке завдання виражається як визначення в даному кінцевому наборі S (простір пошуку), яке задовольняє елемент даного обмеження Ω і забезпечує мінімальне значення для даної функції f.

Для мурашиного алгоритму завдання переводиться в завдання пошуку шляху в графі. Для кожного ребра приписується обсяг феромона τ_i^j (i тут – вершина з якої виходить ребро, а j – вершина в яку входить ребро). На основі цих обсягів обчислюється ймовірність проходження кожного ребра. Кожна з кінцевої колекції мурашок будує свій власний маршрут, випадковим чином вибираючи наступне ребро на основі ймовірності його знаходження. Побудований маршрут потім служить основою для перерахунку обсягу феромона, а потім повторює конструкцію шляху. Процес ітерації триває до

тих пір, поки не буде виконано умову завершення роботи. Початкове значення вибраного об'єму феромона дорівнює деякій константі.

Основна формула алгоритму - розрахунок ймовірності переходу від i-го вузла до j-го на основі наявних обсягів феромона:

$$P_{i,j} = \tau_{i,j}^{\alpha}\eta_{i,j}^{\beta}, \tag{2.7}$$

де $\tau_{i,j}^{\alpha}$ – кількість феромона для ребра, що з'єднує i-й вузел с j-м;

α – параметр, що контролює вплив $\tau_{i,j}^{\alpha}$;

$\eta_{i,j}^{\beta}$ – привабливість ребра, що з'єднує i-й вузол з j-м, яку зазвичай беруть рівній величині, зворотній відстані $d_{i,j}$ між вказаними вузлами, тобто $\eta_{i,j}^{\beta} = 1/d_{i,j}$;

β – параметр, що контролює вплив $\eta_{i,j}^{\beta}$.

На кожній ітерації алгоритму для кожного ребра графа обсяг феромона перераховується за формулою:

$$\tau'_{i,j} = (1-\rho)\tau_{i,j} + \rho\Delta\tau_{i,j}, \tag{2.8}$$

де $\tau_{i,j}$ та $\tau'_{i,j}$ – вихідний і новий обсяг феромона для ребра $i \to j$;

ρ – параметр, що регулює швидкість випаровування феромона;

$\Delta\tau_{i,j}$ – кількість відкладеного феромона всіх мурахів, зазвичай визначається формулою:

$$\Delta\tau_{i,j} = \sum_{k=1}^{m}\Delta\tau_{iij}^{k}, \tag{2.5}$$

де

$$\Delta\tau_{i,j}^{k} = \begin{cases} Q \,/\, L_k, \\ 0 \end{cases}, \tag{2.6}$$

де Q – константа;

L_k – довжина шляху k-го мураха.

Варіант мурашиного алгоритму використовується для планування шляхів у середовищі перешкод на основі карт або графів. Цей алгоритм використовується у зерном середовищі і тривимірному середовищі. В цілому, алгоритм показав себе ефективним алгоритмом пошуку, який дозволяє знайти оптимальний шлях у кілька ітерацій.

Штучні нейронні мережі – сукупність математичних моделей, що моделюють функцію нейронних клітинних мереж в живих організмах. Така мережа являє собою сукупність однорідних елементів (нейронів), кожен з яких має один або більше входів і один або більше виходів, як описано у першому розділі [39].

Нейрони мережі пов'язані з точками в просторі конфігурації, якщо відповідні точки знаходяться поза бар'єром, взаємодія нейронів привабливо або іншим чином відштовхується. Під час навчання нейромережа розширюється шляхом додавання нових елементів. Самоорганізація нейронної мережі і лінії взаємодії нейронів представляють шлях у просторі з перешкодами.

У рамках цієї роботи розглядатимуться глибокі нейронні мережі з генетичним алгоритмом навчання.

3 НЕЙРОЕВОЛЮЦІЙНИЙ ПІДХІД У ЗАВДАННІ ОБХОДУ ПЕРЕШКОД

Разом з моделлю багатошарового персептрона, пізніше з'явилися й інші види нейронних мереж, які відрізняються один від одного по будові окремо взятих нейронів, з топології зв'язків між нейронами і за алгоритмами, що використовуються для навчання.

З найбільш відомих моделей можна назвати НМ зі зворотним поширенням помилки, які засновані на радіальних базисних функціях, самоорганізуються карти Кохонена, узагальнено-регресійні мережі, НМ Хопфілда і Хеммінга, узагальнено-регресійні мережі тощо.

Існують роботи з рекурентних мереж, які містять зворотні зв'язки, що ведуть від більш далеких до ближніх нейронів. Такі мережі можуть мати більш складну динаміку поведінки, ніж традиційні. Також починають ефективно використовуватися самоорганізовані (зростаючі або еволюціонуючи) нейронні мережі, які часто виявляються найбільш кращими, ніж традиційні повнозв'язні НМ.

3.1 Алгоритм зворотного поширення помилки

Алгоритм зворотного поширення помилок – алгоритм для навчання прямого розповсюдження НМ з плоскими шарами. Алгоритми відносяться до парадигми навчання з учителями, тому на прикладі навчання необхідно задати цільове значення. Метод зворотного поширення помилок також є одним з найбільш відомих і часто використовуваних алгоритмів машинного навчання на практиці.

У корені алгоритму лежить застосування вихідної помилки НМ для подальшого обчислення величин корекції вагів нейронів в її прихованих шарах:

$$E = \frac{1}{2}\sum_{i=1}^{k}(y - y')^2, \quad (3.1)$$

де k – кількість вихідних нейронів мережі;

y – цільове значення;

y′ – фактичне вихідне значення.

Алгоритм використовує ітеративний принцип і навчається «крок за кроком» в online-режимі, коли ваги нейронів навченої мережі коригуються після подачі на її вхід одного навчального прикладу.

На кожній ітерації відбувається два проходи мережі – прямий і зворотний. На прямому вхідний вектор поширюється від входів мережі до її виходів і формує деякий вихідний вектор, що відповідає поточному (фактичному) стану терезів. Потім обчислюється помилка нейронної мережі як різниця між фактичним і цільовим значеннями. На зворотному проході ця помилка поширюється від виходу мережі до її входів, і проводиться корекція терезів нейронів відповідно до правила:

$$\Delta\omega_{j,i}(n) = -\eta\frac{\partial E_{av}}{\partial\omega_{ij}}, \quad (3.2)$$

де $\omega_{j,i}$ – вага i-го зв'язку j-го нейрона;

η – параметр швидкості навчання, який дозволяє додатково керувати величиною кроку корекції $\Delta\omega_{j,i}$ з метою більш точного налаштування на мінімум помилки та підбирається експериментально у процесі навчання (змінюється в інтервалі [0,1]).

Враховуючи, що вихідна сума j-го нейрона дорівнює формулі:

$$s_j = \sum_{i=1}^{n}\omega_{ij}x_i, \quad (3.3)$$

можна показати, що

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial S_j}\frac{\partial S_j}{\partial w_{ij}} = x_i \frac{\partial E}{\partial S_j}. \quad (3.4)$$

З останнього виразу випливає, що диференціал ∂S_j активаційної функції нейронів мережі *f(s)* повинен існувати і бути рівним нулю у будь-якій точці, тобто. активаційна функція має бути деференційована по всій числової осі. Тому для застосування методу зворотного поширення використовують сигмоїдні активаційні функції, наприклад, логістичну або гіперболічний тангенс.

Тому алгоритм використовує так званий стохастичний (випадковий) градієнтний спуск для переміщення в зворотному напрямку градієнта в багатовимірному зваженому просторі для досягнення функції мінімальної помилки.

На практиці навчання триває не до точного налаштування мережі на мінімальне значення функції помилки, а для досягнення досить точного наближення. Це дозволить скоротити кількість ітерацій навчання, уникаючи при цьому перенавчання мережі.

Зараз алгоритм зворотного розповсюдження багаторазово модифікований. Наприклад, навчання використовує не «покрокове» (при обчисленні помилок виведення і коригування терезів для кожного прикладу), а «за епохами» в автономному режимі (коли всі приклади навчального набору подаються на мережевий вхід, проводиться зміна терезів, і всі приклади мають середні помилки).

Для багатьох прикладів навчання за епохами є більш стійким до викидів і відхилень цільової функції за рахунок усереднення помилок. Але при цьому зростає ймовірність «заклинювання» алгоритму в локальному мінімумі. Це менш ймовірно для покрокового навчання, тому що за

допомогою одного прикладу створюється «шум», який «виштовхує» алгоритм з ямок градієнтного рельєфу.

Переваги алгоритмів поширення помилок включають простоту реалізації і стійкість до аномалій даних і викидів. До недоліків відносяться:

– невизначений процес навчання;

– можливість «мережевого паралічу», коли робоча точка функції активації відображається значною мірою в області S-насичення і похідна у вираженні стає близькою до 0, корекція ваги фактично не відбувається, і процес навчання «завмирає»;

– вразливість алгоритму до потрапляння в локальний мінімум функції помилки.

3.2 Генетичний алгоритм

Генетичний алгоритм (англ. genetic algorithm) – евристичний алгоритм пошуку, що використовує безперервний відбір, наступні комбінації і дисперсію необхідних параметрів. Алгоритми належать до різних еволюційних обчислень. Генетичні алгоритми відрізняють використання операторів схрещування. Ця операція проводить рекомбінації у рішень (кандидатів), аналогічно схрещуванню в природі.

Такі алгоритми використовують методи випадкового пошуку, призначені для знаходження всіх найкращих доступних рішень. Це обумовлено основною необхідністю пошуку задовільного рішення в разі складних систем. У такому випадку складність досягнення оптимуму не є необхідною.

Інші методи, метою яких є пошук оптимального рішення виявляються неможливими через складність завдання. З іншого боку, як і будь-який інший метод, цей підхід не буде оптимальним для вирішення абсолютно будь-яких завдань. Ще однією властивістю цих алгоритмів є відсутність

впливу людини в процесі пошуку. Людина може тільки задавати конкретні параметри.

Через складність завдання інші способи пошуку найкращого рішення застосовуватися не будуть. З іншого боку, як і в разі будь-якого іншого методу, такий підхід не є оптимальним для повного вирішення будь-якого завдання. Ще однією характеристикою цих алгоритмів є відсутність впливу людини під час пошуку. Людина може вказати тільки конкретні параметри.

У списку основних відмінностей генетичних алгоритмів від традиційних можна виділити наступне:

– генетичні алгоритми працюють з кодами, в яких буде представлено набір параметрів, які безпосередньо залежать від аргументів цільової функції. Інтерпретація таких кодів завжди відбуватиметься до початку роботи алгоритму і після його завершення. Протягом роботи маніпуляція з кодами відбувається незалежно від інтерпретації, код визначається як простий бітовий рядок;

– алгоритм використовує для пошуку декілька точок пошукового простору в один момент часу. Він не переходить від однієї точки до іншої, як це пошукає у виконанні традиційних методах. Така можливість дозволяє обійти нестачу таких алгоритмів – небезпеку потрапляння в локальний екстремум цільової функції, якщо вона не є вгамованою (має кілька екстремумів). Використання декількох точок одночасно значно знижує таку можливість;

– додаткові дані в процесі роботи генетичного алгоритму не використовується, це впливає на підвищення швидкості роботи. Всі дані, які можуть використовуватися алгоритмом – область допустимих значень параметрів та цільової функції в довільній точці;

– крім імовірнісних правил, що використовуються для породження нових точок аналізу, використовуються детерміновані правила відповідають за перехід від одних точок до інших. Одночасне використання

детермінованості та випадковості дає найбільш відчутний ефект на відміну від роздільного.

Алгоритм працює з кодами незалежно від їх смислової інтерпретації. Завдяки цьому код і його структура будуть описуватися поняттям генотипу. Інтерпретація – поняттям фенотипу. У своїй суті кожен код представляє конкретну точку в просторі. Для максимального наближення до біологічної термінології екземпляр коду носить назву хромосому, індивідуум або особину. Далі для позначення рядка коду буде використовуватися термін «особина».

Генетичний алгоритм на кожному кроці використовує одночасно кілька точок пошуку. Сукупність всіх таких точок є набором особин, який також називається популяцією. Розмір популяції – особа в ній особин. Під час кожного кроку роботи алгоритм оновлює популяцію створюючи нові особини і знищує непотрібні. Для відмінності популяцій на кожному з кроків і під час кожного кроку, вони називаються поколіннями і мають номер для ідентифікації. Наприклад, популяція, яка була отримана з вихідної популяції після першого кроку роботи алгоритму, буде першим поколінням, після наступного кроку – другим тощо.

На рисунку 3.1 зображена блок-схема, що представляє роботу генетичного алгоритму.

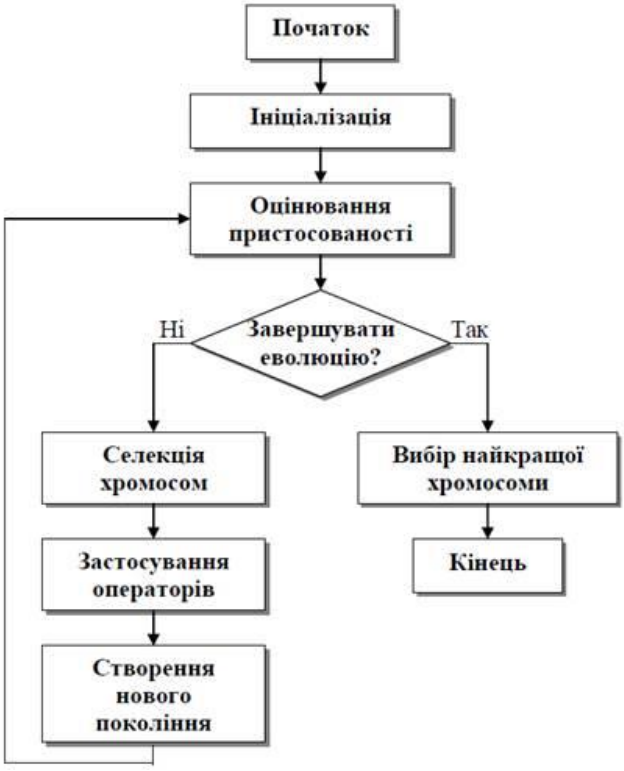

Рисунок 3.1 – Робота генетичного алгоритму

У процесі виконання генетичного алгоритму генеруються нові особини на основі моделювання природного процесу розмноження. Породжуючи особини носять ім'я батьків, а породжені – нащадками, за аналогією з біологічними концепціями. Зазвичай пара батьків породжує іншу пару нащадків. Генерація нових кодових рядків з двох обраних відбувається завдяки оператору схрещування. Оператор схрещування може бути застосований у разі породження нової популяції, проте не до всіх пар батьків. Частина таких пар безпосередньо може полягати в популяції подальшого покоління. Частота виникнення такої ситуації залежатиме від ймовірності застосування цього оператора.

Моделювання процесу мутації нових особин може бути можливим за рахунок роботи оператора мутації. Головний параметр оператора мутації – ймовірність мутації. У зв'язку з тим, що розмір популяції є фіксованим породження нащадків має супроводжуватися знищенням інших особин.

Оператор відбору визначає пари батьків з популяції під час породження нащадків, а оператор редукції – особини для знищення. Вибір пар батьків з популяції для породження нащадків виробляє оператор відбору, а вибір особин для знищення - оператор редукції. Якість особи, визначена значенням цільової функції, є головним параметром для роботи цих операторів.

Тепер можна перерахувати всі основні поняття і терміни, що застосовуються в області використання генетичних алгоритмів: генотип, фенотип, особина, популяція, покоління, батьки і нащадки.

До характеристик, що визначають генетичний алгоритм, можна віднести розмір популяції і такі генетичні оператори, як: оператор схрещування, мутації, відбору та редукції.

3.3 Ефективність генетичних алгоритмів

Ефективність генетичних алгоритмів у вирішенні конкретних завдань залежить від багатьох факторів, особливо від вибору генетичних операторів і відповідних значень параметрів, і від того, як завдання представлено на хромосомі. Оптимізація цих факторів покращує швидкість і стабільність пошуку, а також має велике значення для застосування генетичного алгоритму.

Швидкість генетичних алгоритмів оцінюється за часом, який потрібен для виконання користувацьких ітерацій. Якщо критерієм зупинки є якість популяції або її схожість, то швидкість буде оцінюватися залежно від часу, за який алгоритм досягає одна з подій.

Стабільність пошуку оцінюється ступенем алгоритмічної стійкості до потрапляння в локальні екстремальні точки і здатністю безперервно покращувати якість передачі з покоління в покоління.

Від цих факторів – швидкості і стабільності – залежить ефективність генетичних алгоритмів, для вирішення конкретного завдання з конкретними умовами.

3.3.1 Швидкість генетичних алгоритмів

Основний спосіб підвищення швидкості генетичних алгоритмів – паралелізація. Крім того, цей процес можна розглядати з двох точок зору. Паралелізація може відбуватися на першому рівні роботи організаційного генетичного алгоритму і на першому рівні прямої реалізації на комп'ютерах.

У другому випадку використовуються такі особливості генетичного алгоритму: У цьому процесі багаторазово обчислюється значення цільової функції для кожного університету, перетворюється оператор кросовера, мутується кілька пар батьків тощо. Всі ці процеси можуть бути реалізовані одночасно на декількох паралельних системах або процесорах, що пропорційно збільшить швидкість виконання алгоритму.

У першому випадку структурування популяції застосовується на основі одного з двох способів:

– популяція ділиться на кілька різних підгруп і згодом розвивається незалежно і паралельно. Тобто кросовер зустрічається тільки серед учасників однієї популяції. На одному етапі операції деякі особини обмінюються між підгрупами на основі випадкової вибірки. І це може тривати до тих пір, поки алгоритм не буде виконаний. Ця практика відома як концепція острова;

– для кожної особи визначається її просторове положення в популяції. Схрещування при експлуатації відбуваються між найближчими з них. Ця практика відома як концепція локального перетину.

Очевидно, що ці два методи можуть бути ефективно реалізовані на паралельних комп'ютерах. Крім того, практика показала, що навіть при

використанні традиційних обчислювальних засобів структурування популяцій підвищує ефективність генетичних алгоритмів.

Ще одним засобом підвищення продуктивності є кластеризація. Принцип полягає в двоетапній роботі генетичних алгоритмів. На першому етапі генетичні алгоритми працюють традиційним чином для отримання більш «хороших» груп рішень. Після завершення алгоритму вибирається найближча група прийняття рішення з кінцевої сукупності. Ці групи як одне ціле формують початкову популяцію для другої фази роботи генетичного алгоритму. Розмір такої популяції природно значно менший, тому алгоритм шукає далі швидше. У цьому випадку простір пошуку не скоротиться, тому що тільки кілька дуже схожих особин не матимуть істотного впливу на нове рішення і не будуть розглядатися.

3.3.2 Стійкість роботи генетичних алгоритмів

Стабільність або здатність генетичних алгоритмів ефективно формувати кращі рішення залежить насамперед від того, як працюють генетичні оператори (оператори селекції, кросовера, мутації та відновлення). Механізм цього ефекту розглядається більш докладно.

Загалом масштаби впливу можна оцінити шляхом розгляду випадків деградації генетичних операторів.

Вироджена форма операторів проходження є, з одного боку, точною реплікацією батьківського потомства і, з іншого боку, поколінням потомства, яке найбільш відрізняється від них.

Перший варіант має перевагу пошуку кращого рішення на ранньому етапі. Браком, у свою чергу, є те, що алгоритм не знайде кращого рішення, ніж вже міститься у вихідній популяції. Тому що в цьому випадку алгоритм не генерує принципово нові особини, а лише копіює вже існуючі. Щоб як і раніше використовувати переваги цієї кінцевої форми перехресного

оператора в дійсному генетичному алгоритмі, використовується елітарність.

У другому випадку алгоритм розглядає максимальну кількість різних особин і розширює область пошуку, що призводить до кращих результатів. У цьому випадку недоліком є значне уповільнення пошуків. Однією з причин цього є те, що потомство, зокрема, значно відрізняється від батьків і не успадковує їх вигідне майно.

Як справжні оператори кросовера використовуються проміжні варіанти. Зокрема, батьківська репродукція з мутацією, а також з рекомбінацією і мутацією. Батьківське відтворення вказує на те, що батьківський рядок копіюється до дочірнього рядка. У першому випадку після цього потомство зазнавало впливу мутації. У другому випадку після реплікації відбувається обмін потомством, процес, відомий як рекомбінація, який описаний у попередньому абзаці. Після рекомбінації потомство також піддається мутаціям. Останній підхід найбільш поширений в області генетичних алгоритмів.

У цьому випадку найбільш поширені одномісні, одноточковий, двоточковий і рівномірний оператори кросоверів. Їх назва походить від принципу поділу кодових рядків на підрядки. Рядки можна розділити на підрядки в одному або двох розташуваннях відповідно. Або рядки можуть утворювати нащадки, які чергують свої елементи.

Основним параметром оператора мутації є ймовірність його впливу. Зазвичай його вибирають досить маленьким. Щоб розширити область пошуку з одного боку, а з іншого боку, це не змушує нащадку занадто серйозно змінюватися, всупереч вигідним батьківським параметрам спадкування. Характер мутаційного ефекту зазвичай визначається фенотипом і не надає особливого впливу на ефективність алгоритму.

Існує також стратегія розширення простору пошуку, звана стратегією різноманітності. Якщо генетичні алгоритми використовують цю стратегію, кожне породжене потомство зазнає невеликої випадкової зміни.

Різноманітність і мутація різняться тим, що оператори мутації вносять досить значні зміни в хромосоми, а оператори різноманітності – навпаки. Це є основною причиною застосування абсолютної ймовірності рознесення. Адже якщо ви часто вносите невеликі зміни в хромосоми вашого потомства, вони можуть бути корисні при розширенні пошукового простору і успадкуванні корисних властивостей. Слід зазначити, що стратегії різноманітності застосовуються не до всіх генетичних алгоритмів, оскільки вони є лише засобом підвищення ефективності.

Іншим важливим фактором, що впливає на ефективність генетичних алгоритмів, є вибір операторів. Сліпе слідування принципу «виживання на найсильнішому» може призвести до більш вузької області пошуку і рішення, знайденого в локальній крайній області об'єктивної функції. З іншого боку, занадто слабкий оператор вибору може уповільнити зростання якості населення, що призведе до більш повільних швидкостей пошуку. Крім того, чисельність населення не тільки не поліпшується, а й погіршується. Найбільш поширеними батьківськими операторами вибору є:

– випадковий рівномовірний відбір;

– рангово-пропорційний відбір;

– відбір пропорційний значенню цільової функції.

Види операторів редукції особин для збереження розміру популяції практично збігаються з видами операторів відбору батьків. Серед них можна перерахувати:

– випадкове рівноімовірне вилучення;

– вилучення k найгірших;

– вилучення, зворотно пропорційне значенню цільової функції.

Оскільки процедури відбору батьків і редукції рознесені за часом дії і мають різне значення. Проводяться активні дослідження, щоб визначити, як узгодженість цих процедур впливає на ефективність генетичних алгоритмів.

Одним з параметрів, що впливають також на стабільність і швидкість пошуку, є загальний розмір роботи алгоритму. Класичні генетичні

алгоритми припускають, що розміри популяції повинні бути фіксованими. Такі алгоритми називаються алгоритмами стаціонарного стану.

Однак практика показує, що іноді корисно змінювати чисельність популяції в певних межах. Подібний алгоритм називається поколенським. При цьому усічення популяції не відбувається після наступного покоління потомства. Таким чином, у декількох ітераціях розмір популяції може зростати до досягнення певного порогу. Після цього населення було усічене до початкового розміру. Такий підхід допомагає розширити область пошуку, але не призводить до значного зниження швидкості, оскільки усічення популяції, хоча і менш поширені, все ж відбуваються.

4 ІМІТАЦІЙНЕ МОДЕЛЮВААНЯ ТА ВИРІШЕННЯ ЗАДАЧІ ОБХОДУ ПЕРЕШКОД

У рамках імітаційного моделювання необхідно побудувати функціональну глибоку нейронну мережу, здатну до навчання за допомогою еволюції, а точніше генетичного алгоритму. Код є розширюваним, що дозволяє при необхідності вносити зміни в архітектуру нейронної мережі. Іншими словами, це дозволить легко змінювати спосіб роботи мережі.

В ході моделювання буде налаштована функція зворотного зв'язку, всі необхідні масиви і навчання на основі мутацій.

В якості інструменту для імітаційного моделювання обрано багатоплатформне середовище розробки Unity, основними перевагами якого є наявність візуального середовища розробки для кращої візуалізації процесу навчання нейронної мережі, багатоплатформна підтримка і модульна система компонентів. Як мову програмування обрано C# через її сумісність з Unity.

4.1 Імітаційне моделювання багатошарової нейронної мережі з використанням генетичного алгоритму.

В рамках експерименту буде створена проста модель багатошарової нейронної мережі, що складається з 5-и нейронів на вхідному шарі, 4-х нейронів прихованого шару і 3-х вихідних. Розмір шарів і кількість прихованих нейронів можуть бути довільними. У рамках зазначеної візуальної моделі було обрано менший розмір.

Ідея нейронної мережі полягає в тому, що значення кожного вхідного вузла передається через кожен дендрит (з'єднує лінія на моделі) і помножено на вагу відповідного дендриту. Потім результат передається в нейрон в наступному шарі. Цикл триватиме доти, доки не буде отримано висновок:

$$\sigma(w_1 a_1 + w_2 a_2 + \cdots + w_n a_n + b), \tag{4.1}$$

де b – зсув, доданий до вузла;

w – вага дендриту;

σ – функція активації;

а – активація кожного нейрона у попередньому шарі.

Щоб забезпечити оптимальну продуктивність мережі використовується серія масивів.

Як розмір мережі оголошується вхідний масив [5, 4, 3], аналогічно моделі створюваної нейронної мережі. Також потрібен масив нейронів для зберігання значень, які будуть генеруватися в ході алгоритму у вигляді 2-зубчастого масиву. Ще один подібний масив зберігатиме зміщення в кожному шарі.

Кожен раз під час створення класу розміри мережі повинні бути визначені і всі масиви ініціалізовані. Для цього створена функція ініціалізації, яка приймає вхідний масив мережевих вимірювань і заповнює кожен з масивів.

З усіма описаними функціями можна перейти до алгоритму прямого зв'язку і навколишніми його концепціями.

Зважена сума – ті дані, які передаються у функцію активації. Це кожен з вузлів у попередньому шарі, помножений на вагу в дендриті, за яким дані переносяться в поточний нейрон.

Як функцію активації можна вибрати будь-яку, хоча в різних завданнях корисними можуть виявитися різні функції активації.

Серед найбільш використовуваних функцій активації можна виділити:

– лінійну;

– ступеневу;

– сигмоїдальну;

– Tanh;

– ReLu;

– Leaky ReLu;

– Softmax.

У рамках цієї роботи буде використовуватися функція активації Tanh для позитивних і негативних значень.

Tanh можна представити в наступному вигляді:

$$tanh(x) = \frac{2}{\left(1+e^{(-2x)}\right)\ -1} \tag{4.2}$$

Програма виконує ітерацію по кожному з нейронів, отримує значення в кожному з нейронів наступного шару, множить значення на вагу, виконуючи цей крок через функцію активації і потім зупиняє власне значення для активації.

Для реалізації створеної нейронної мережі використовується генетичний алгоритм. Виділимо основні етапи цього алгоритму:

– задати цільову функцію для особин популяції;

– створити початкову популяцію;

– запустити цикл навчання, що складається з 5 кроків: розмноження, мутації, обчислення значення цільової функції для кожної особини, створення нового покоління на основі отриманих результатів і зупинити цикл при отриманні задовольняючого результату, або повернутися в початок циклу.

Навіть якщо перша популяція виявиться абсолютно неробочою, генетичний алгоритм шляхом проб і помилок приведе її в життєздатну популяцію, здатну виконувати поставлене завдання. Таким чином, на першому кроці достатньо підрахувати функцію пристосованості (фітнес-функцію) першої популяції, щоб вибрати частку тих особин, що залишаться «в живих».

На етапі відбору особин з популяції в рамках поставленого завдання використовується та особина, яка володіє найкращим значенням функції пристосованості.

Оскільки головним завданням є обхід перешкод, багато особин почнуть крутитися на місці і вибиратися алгоритмом, як найбільш успішні. В обхід подібного розвитку подій на шляху у особин будуть розставлені контрольні точки, а найуспішнішою особиною буде вважатися та, яка пройде більшу кількість контрольних точок з унікальними ідентифікаторами.

При використанні середовища розробки Unity було створено навколишнє середовище з перешкодами у формі лабіринту. Метою в такій ситуації буде пройти якомога далі інших особин, не стикаючись зі стінами і знайти вихід.

Сценою в Unity керуватиме менеджер зображений на рисунку 4.1. За допомогою цього інструменту будуть створюватися популяції, мутувати і руйнуватися.

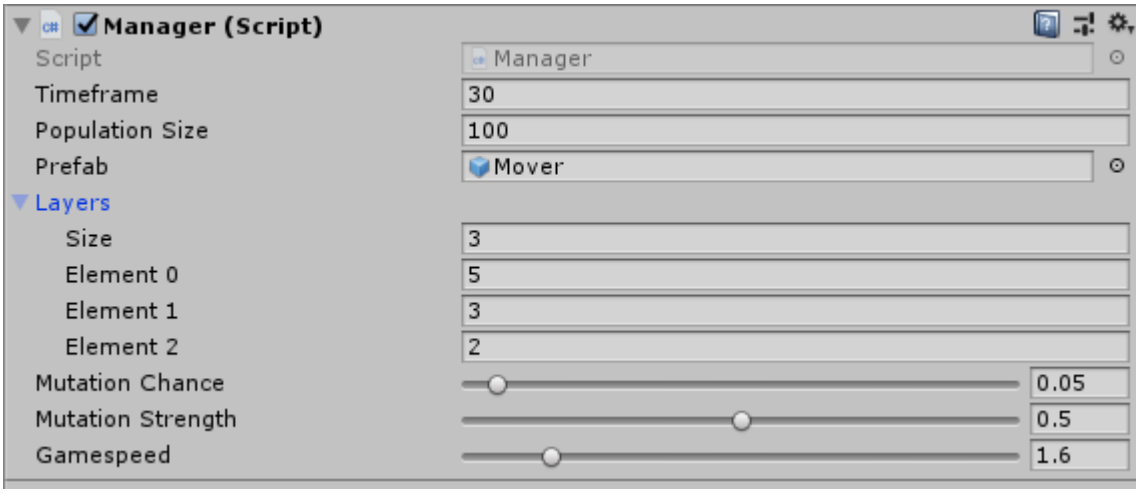

Рисунок 4.1 – Менеджер для управління ходом навчання

З основних понять, що використовуються у створенні середовища з перешкодами, варто виділити такі:

– таймфейм (Timeframe) – час, протягом якого будуть навчатися покоління досліджуваних особин;

– розмір популяції (Population size) – кількість особин з однієї популяції, які проходять навчання в один момент часу;

– швидкість навчання (Gamespeed) – частота оновлення даних, підвищення цього параметра буде зменшувати час, що витрачається на навчання мережі, проте вимагати більше продуктивних ресурсів;

– префаб (Prefab) – шаблон створюваного об'єкта, що виконує роль особини в навчанні нейронної мережі.

Менеджер створює групу префабів, що використовують нейронну мережу, потім розгортає мережу в кожному з них. Через деякий час тестування буде завершено, і мережі будуть відсортовані таким чином, що найефективніші з них будуть збережені, а найгірші мутують у кращі. Потім розгортання відбувається знову, і цикл навчання триває до тих пір, поки результат навчання не буде задовільним.

Особина (префаб) здатна пересуватися з фіксованою швидкістю і повертати в разі виявлення перешкоди. Виявити перешкоду особина може завдяки 5-и датчикам відстані, створених за допомогою Raycast з бібліотек вбудованих в Unity. Raycast – деякий промінь, що випускається з особини в певному напрямку нескінченної довжини або фіксованої, як в даному випадку. Raycast розраховує відстань до перешкоди в п'яти напрямках і з деякою ймовірністю перенаправляє об'єкт, щоб не зіткнутися з ним. Особини не вважають перешкодами один одного, що не заважає навчанню нейронної мережі.

На рисунках 4.2-4.5 зображено процес навчання нейронної мережі з моделюванням особин в умовах лабіринту.

Рисунок 4.2 – Навчання НМ на першій ітерації

Рисунок 4.3 – Навчання НМ через декілька ітерацій

Рисунок 4.4 – Навчання НМ через 10-20 ітерацій

Рисунок 4.5 – Навчання НМ через 20-30 ітерацій

4.2 Результати імітаційного моделювання

В якості аналізу отриманих результатів розглянемо інші методи планування шляху і обходу перешкод для створення порівняльної характеристики.

Методи засновані на графах легко реалізувати в комп'ютерному моделюванні. Базові графічні підходи призначені для статичних середовищ, хоча деякі з цих методів можуть бути перетворені на уявлення, що підходять для динамічних середовищ. Оскільки графіки пов'язані з опорними точками, результати цих методів являють собою ламані лінії. Крім того, отриманий результат не є оптимальним: він припустимий або відносно оптимальний.

Метод потенційного поля дозволяє створювати гладкі маршрути. Реалізація способу відносно проста. Перевагою цього методу є короткий час обчислення. Тому він часто використовується в локальному плануванні, коли розрахунки повинні виконуватися в режимі реального часу. Але при такому підході об'єкти можуть досягти локального мінімуму і в кінцевому підсумку не досягти саму мету.

Перевага методу оптимізації полягає в тому, що він може враховувати різні обмеження, такі як динамічні або керуючі обмеження. Також можуть бути включені обмеження в безпеці, надійності та інших факторів. Однак у міру збільшення розміру простору стану і площі середовища модель швидко ускладняється в міру збільшення ступеня деталізації, що уповільнює процес розрахунку. Алгоритми оптимізації не завжди можуть бути реалізовані в реальному часі.

Методи, засновані на інтелектуальних алгоритмах, допомагають вирішувати дуже складні завдання з динамічними обмеженнями. Але найбільший недолік полягає в тому, що цикл планування може бути занадто довгим для використання в реальному часі. Однак у статичному середовищі такі методи показують досить оптимальні результати.

Порівняльні характеристики методів планування маршруту і обходу перешкод вказані на табл. 4.1.

Таблиця 4.1 – Порівняльні характеристики методів планування маршруту та обходу перешкод.

Методи	Робота в реальному часі	Тип навколишнього середовища	Характер шляху	Оптимальність рішення
На основі графа або дерева	Нормально	Статичне	Негладкий	Відносно допустима
На основі клітинної декомпозиції	Нормально	Статичне	Негладкий	Відносно допустима
Метод потенційних полів	Добре	Будь-яке	Гладкий	Наближено допустима
Оптимізаційні	Добре	Будь-яке	Гладкий	Наближено допустима
Інтелектуальні	Слабко	Статичне	Негладкий	Допустима

У рамках цієї роботи навіть невелика багатошарова нейронна мережа з використанням генетичного алгоритму навчання задовільно справляється із завданням обходу перешкод і пошуку маршруту в заданому лабіринті. Однак, чим більше буде навколишнє середовище, тим більше часу буде йти на навчання НМ. Також навчання можна прискорити за наявності великої обчислювальної потужності.

ВИСНОВКИ

В ході даної роботи було досліджено методи вирішення задач обходу перешкод та пошуку маршруту.

Перший розділ роботи присвячений аналізу предметної галузі та основним поняттям нейронний мереж. Було розглянуто принцип навчання штучних нейронних мереж з вчителем та без. Проаналізовано проблеми навчання НМ і способи їх уникнення. Також проведено аналіз предметної області та постановка задачі дослідження.

У другому розділі розглянута класифікація основних методів обходу перешкод та пошуку маршруту. Серед яких можна визначити методи на основі графів, клітинної декомпозиції, методи потенційних полів, оптимізаційні методи та методи на інтелектуальних алгоритмах, один з яких й розглядується у темі роботи.

Третій розділ присвячено концепціям і принципам навчання нейронних мереж за допомогою генетичного алгоритму та його відмінності від алгоритму зворотного поширення помилки. Також було проаналізовано ефективність генетичних алгоритмів.

Четвертий розділ описує результати імітаційного моделювання за використовуванням середовища розробки Unity, що дозволило візуалізувати процес навчання НМ. Була створена проста глибока нейронна мережа із застосуванням генетичного алгоритму та навчена для пошуку маршруту з обходом перешкод у лабіринті, який описував навколишнє середовище.

Отримані результати роботи можуть бути використані для обходу перешкод у статичному просторі, наприклад, для роботів-пилососів, прототипів транспортних засобів, імітації штучного інтелекту у комп’ютерних іграх тощо.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Хайкин С. Нейронные сети: полный курс, Neural Networks: A Comprehensive Foundation. 2-е изд. — М.: Вильямс, 2006. – 1104 с.

2. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечёткие системы. // Д. Рутковская, пер. с польского И. Д. Рудинского. -М.: Горячая линия–Телеком, 2007. – 452 с.

3. Круглов В. В., Борисов В. В. Искусственные нейронные сети. Теория и практика. — М.: Горячая линия - Телеком, 2001. – 382 с

4. Горбань А.Н. Обучение нейронных сетей. – М.: СССР-США СП «Параграф», 1990. – 160 с.

5. Круглов В. В., Дли М. И., Голунов Р. Ю. Нечеткая логика и искусственные нейронные сети: Учеб. пособие. М.: Издательство физико-математической литературы. 2001.

6. Каллан Р. Основные концепции нейронных сетей. Пер. с англ. М.: Издательский дом «Вильямс». 2001.

7. Meng Wang, Liu J.N.K. Fuzzy logic-based real-time robot navigation in unknown environment with dead ends. Robotics and Autonomous Systems, 2008, vol. 56, no. 7, pp. 625–643.

8. Мурашиний алгоритм URL: https://ru.wikipedia.org/wiki/Муравьиный_алгоритм (дата звернення: 30.11.2021).

9. Mohamad M.M., Dunnigan M.W., Taylor N.K. Ant colony robot motion planning. EUROCON 2005: Intern. conf. on computer as a tool (Belgrade, Serbia, November 21-24, 2005): Proc. N.Y.: IEEE, 2005. Vol. 1. pp. 213–216. DOI: 10.1109/EURCON.2005.1629898

10. Штучна нейронна мережа URL: https://ru.wikipedia.org/wiki/ Искусственая_ нейронная_сеть (дата звернення: 01.12.2021)

11. Janet J.A., Luo R.C., Kay M.G. The essential visibility graph: An approach to global motion planning for autonomous mobile robots. IEEE intern.

conf. on robotics and automation (Nagoya, Japan, May 21-27, 1995): Proc. Vol. 2. N.Y.: IEEE, 1995. Pp. 1958–1963.

12. Habib M.K., Asama H. Efficient method to generate collision free paths for an autonomous mobile robot based on new free space structuring approach. IEEE/RSJ intern. workshop on intelligent robots and systems: IROS'91 (Osaka, Japan, November 3-5, 1991): Proc. Vol. 2. N.Y.: IEEE, 1991. Pp. 563–567.

13. Wallgrun J. O. Voronoi graph matching for robot localization and mapping. Transactions on computational science IX. B.: Springer, 2010. Pp. 76–108.

14. Ladd A.M., Kavraki L.E. Measure theoretic analysis of probabilistic path planning. IEEE Trans. on Robotics and Automation, 2004, vol. 20, no. 2, pp. 229–242.

15. Geraerts R., Overmars M.H. A comparative study of probabilistic roadmap planners. Algorithmic foundations of robotics. B.: Springer, 2004. Pp. 43–57.

16. Sleumer N.H., Tschichold-Gurman N. Exact cell decomposition of arrangements used for path planning in robotics. Zurich: Inst. of Theoretical Computer Science, 1999.

17. Elfes A. Using occupancy grids for mobile robot perception and navigation. Computer, 1989, vol. 22, no. 6, pp. 46–57.

18. Yahja A., Stentz A., Singh S., Brumitt B.L. Framed-quadtree path planning for mobile robots operating in sparse environments. IEEE intern. conf. on robotics and automation (Leuven, Belgium, May 20, 1998): Proc. N.Y.: IEEE, 1998. Vol. 1. Pp. 650–655.

19. Kitamura Y., Tanaka T., Kishino F., Yachida M. 3-D path planning in a dynamic environment using an octree and an artificial potential field. IEEE-RSJ intern. conf. on intelligent robots and systems: IROS’95 (Pittsburgh, PA, USA, Aug. 5-9, 1995): Proc. N.Y.: IEEE, 1995. Vol. 2. Pp. 474–481.

20. Redding J., Amin J., Boskovic J., Kang Y., Hedrick K., Howlett J., Poll S. A real-time obstacle detection and reactive path planning system for

autonomous small-scale helicopters. AIAA Guidance, navigation and control conf. and exhibit (Hilton Head, USA, Aug. 20–23, 2007): Proc. Wash.: AIAA, 2007. Pp. 989–1010.

21. Chazelle B., Palios L. Triangulating a nonconvex polytope. Discrete and Computational Geometry, 1990, vol. 5, no. 5, pp. 505–526.

22. Russell S.J., Norvig P. Artificial intelligence: A modern approach. 3rd ed. Upper Saddle River: Prentice Hall, 2010. 1132 pp.

23. Ferguson D., Stentz A. Using interpolation to improve path planning: The field D* algorithm. J. of Field Robotics, 2006, vol. 23, no. 2, pp. 79–101.

24. Koenig S., Likhachev M., Furcy D. Lifelong planning A*. Artificial Intelligence, 2004, vol. 155, no. 1-2, pp. 93–146.

25. Koenig S., Likhachev M. D* lite. 18th national conf. on artificial intelligence (Edmonton, Alberta, Canada, July 28–August 1, 2002): Proc. Menlo Park: AAAI Press, 2002. pp. 476–483.

26. De Filippis L., Guglieri G., Quagliotti F. Path planning strategies for UAVs in 3D environments. J. of Intelligent and Robotic Systems, 2012, vol. 65, no. 1–4, pp. 247–264.

27. De Filippis L. Advanced path planning and collision avoidance algorithms for UAVs: Doct. diss. Torino: Ist. Politecnico di Torino, 2012. 142 p.

28. Alvarez D., Gomez J.V., Garrido S., Moreno L. 3D robot formations path planning with fast marching square. J. of Intelligent and Robotic Systems, 2015, vol. 80, no. 3-4, pp. 507–523.

29. Fujimura K., Samet H. A hierarchical strategy for path planning among moving obstacles (mobile robot). IEEE Trans. on Robotics and Automation, 1989, vol. 5, no. 1, pp. 61–69.

30. Masoud A.A. Solving the narrow corridor problem in potential field-guided autonomous robots. IEEE intern. conf. on robotics and automation: ICRA 2005 (Barcelona, Spain, April 18-22, 2005): Proc. N.Y.: IEEE, 2005. Pp. 2909–2914.

31. Fan Xiao-ping, Li Shuang-yan, Chen Te-fang. Dynamic obstacle-avoiding path plan for robots based on a new artificial potential field function. Control Theory and Applications, 2005, vol. 22, no. 5. Pp. 703–707.

32. Borenstein J., Koren Y. Real-time obstacle avoidance for fast mobile robots. IEEE Trans. on Systems, Man, and Cybernetics, 1989, vol. 19, no. 5, pp. 1179–1187.

33. Ulrich I., Borenstein J. VFH+: Reliable obstacle avoidance for fast mobile robots. IEEE intern. conf. on robotics and automation (Leuven, Belgium, May 20, 1998): Proc. N.Y.: IEEE, 1998. Vol. 2. Pp. 1572–1577.

34. Ulrich I., Borenstein J. VFH*: Local obstacle avoidance with look-ahead verification. IEEE intern. conf. on robotics and automation: ICRA'00 (San Francisco, CA, USA, April 24-28, 2000): Proc. N.Y.: IEEE, 2000. Vol. 3. Pp. 2505–2511.

35. Betts J.T. Survey of numerical methods for trajectory optimization. J. of Guidance, Control and Dynamics, 1998, vol. 21, no. 2, pp. 193–207.

36. Culligan K., Valenti M., Kuwata Y., How J.P. Three-dimensional flight experiments using on-line mixed-integer linear programming trajectory optimization. Amer. control conf.: ACC'2007 (New York, NY, USA, July 9-13, 2007): Proc. N.Y.: IEEE, 2007.
Pp. 5322–5327.

37. Schouwenaars T., De Moor B., Feron E., How J. Mixed integer programming for multivehicle path planning. Eur. control conf.: ECC 2001 (Porto, Portugal, Sept. 4-7, 2001): Proc. N.Y.: IEEE, 2001. Pp. 2603–2608.

38. Earl M.G., D'Andrea R. Iterative MILP methods for vehicle-control problems. IEEE Trans. on Robotics, 2005, vol. 21, no. 6, pp. 1158–1167.

39. Masehian E., Habibi G. Robot path planning in 3D space using binary integer programming. Intern. J. of Computer, Information, Systems and Control Engineering, 2007, vol. 1, no. 5, pp. 1240-1245.