Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет прикладних інформаційних технологій та електроінженерії
(повна назва факультету)
Кафедра комп'ютерно-інтегрованих технологій
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розроблення інтелектуальної системи оптимізації маршрутів громадського транспорту із застосуванням нейронних мереж

Виконав: студент IV курсу, групи КТ–41

спеціальності 151 Автоматизація та комп’ютерно–інтегровані технології
(шифр і назва спеціальності)

_____ Кіндратюк Б.М.
(підпис) (прізвище та ініціали)

_____ Шеремета О.В.
(підпис) (прізвище та ініціали)

Керівник _____ Королюк Р.І.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Чихіра І.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Микитишин А.Г.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль
2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет прикладних інформаційних технологій та електроінженерії
(повна назва факультету)
Кафедра комп'ютерно-інтегрованих технологій
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Микитишин А.Г
(підпис) (прізвище та ініціали)
«_31_» _березня_ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)
за спеціальністю 151 Автоматизація та комп'ютерно інтегровані технології
(шифр і назва спеціальності)
Студенту Кіндратюку Богдану Миколайовичу
(прізвище, ім'я, по батькові)
1. Тема роботи Розроблення інтелектуальної системи оптимізації маршрутів громадського транспорту із застосуванням нейронних мереж

Керівник роботи Королюк Ростислав Ігорович, старший викладач кафедри КТ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
Затверджені наказом ректора від «*31*» березня 2025 року № *4/7–239*
2. Термін подання студентом завершеної роботи 25 червня 2025р.
3. Вихідні дані до роботи Наукові публікації про інтелектуальні системи оптимізації маршрутів транспорту, застосування нейронних мереж

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1 Аналітична частина. 1.1 Задача оптимізації маршрутної мережі громадського транспорту 1.2 Застосування методів ШІ у транспортному плануванні 1.3 Ста і перспективи Застосування ШІ для оптимізації громадського транспорту 1.4 Висновок до першого розділу 2. Проектна частина 2.1 Нейронні мережі як інструмент для оптимізаційних задач 2.2 Перцептрони та багатошарові мережі прямого поширення (MLP) 2.3 Рекурентні нейронні мережі (RNN) та послідовні моделі 2.4 Архітектури з механізмом уваги та трансформери 2.5 Графові нейронні мережі(GNN) у задачах маршрутизації 2.6 Інші типи нейронних мереж Та гібридні підходи 2.7 Аналіз переваг та обмежень нейромережевих методів 2.8 Висновок До другого розділу 3 Спеціальна частина 3.1 Постановка практичного завдання 3.2 Архітектура програмної системи 3.3 Навчання та налаштування системи 3.4 Отримані результати та приклади рішень 3.5 Переваги та недоліки системи 3.6 Висновок до третього розділу 4. Безпека життєдіяльності, основи охорони праці. Висновки. Перелік джерел. Додатки
5. Перелік графічного матеріалу
1 Титульна сторінка. 2 Тема, Мета, Об'єкт, Предмет дослідження. 3 Завдання дослідження. 4 Актуальність дослідження. 5 Приклад вирішення задачі оптимізації маршрутної мережі громадського транспорту. 6 Проєктування задачі TNDP. 7 Проєктування рішення TNDP. 8 Постановка практичного завдання та підготовка даних. 9 Архітектура програмної системи 10 Навчання та налаштування системи. 11 Отримані результати та приклади рішень. 12 Щоденна динаміка/порівняння. 13 Переваги, недоліки та можливості вдосконалення системи 14 Висновки. 15 Завершальний.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	к.т.н., доц. Сенчишин В.С.	12.06.2025	15.06.2025

7. Дата видачі завдання 31 березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	31.03.2025	*Виконано*
2.	Підбір джерел про системи моніторингу	02.06.2025	*Виконано*
3.	Вибір компонентної бази	04.06.2025	*Виконано*
4.	Виконання розроблення інтелектуальної оптимізації	05.06.2025	*Виконано*
	маршрутів транспорту з нейронними мережами		
5.	Переклад та опрацювання джерел	06.06.2025	*Виконано*
6.	Розроблення прототипу терміналу	07.06.2025	*Виконано*
7.	Розроблення панелі керування системи	08.06.2025	*Виконано*
8.	Оформлення розділу «Аналітична частина»	10.06.2025	*Виконано*
9.	Оформлення розділу «Проектна частина»	13.06.2025	*Виконано*
10.	Оформлення розділу «Спеціальна частина»	15.06.2025	*Виконано*
11.	Виконання розділу «Охорона праці та безпека в НС»	16.06.2025	*Виконано*
12.	Підготовка графічного матеріалу	17.06.2025	*Виконано*
13.	Оформлення кваліфікаційної роботи	18.06.2025	*Виконано*
14.	Нормоконтроль	19.06.2025	*Виконано*
15.	Перевірка на плагіат	20.06.2025	*Виконано*
16.	Попередній захист кваліфікаційної роботи	23.06.2025	*Виконано*
17.	Захист кваліфікаційної роботи	25.06.2025	
–			

Студент _____ Кіндратюк Б.М.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Королюк Р.І.
(підпис) (прізвище та ініціали)

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет прикладних інформаційних технологій та електроінженерії
(повна назва факультету)
Кафедра комп'ютерно-інтегрованих технологій
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Микитишин А.Г
(підпис) (прізвище та ініціали)
«_31_» _березня_ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)
за спеціальністю 151 Автоматизація та комп'ютерно інтегровані технології
(шифр і назва спеціальності)
Студенту Шереметі Олександру Васильовичу
(прізвище, ім'я, по батькові)
1. Тема роботи Розроблення інтелектуальної системи оптимізації маршрутів громадського транспорту із застосуванням нейронних мереж

Керівник роботи Королюк Ростислав Ігорович, старший викладач кафедри КТ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
Затверджені наказом ректора від «*31*» березня 2025 року № *4/7–239*
2. Термін подання студентом завершеної роботи 25 червня 2025р.
3. Вихідні дані до роботи Наукові публікації про інтелектуальні системи оптимізації маршрутів транспорту, застосування нейронних мереж

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1 Аналітична частина. 1.1 Задача оптимізації маршрутної мережі громадського транспорту 1.2 Застосування методів ШІ у транспортному плануванні 1.3 Ста і перспективи Застосування ШІ для оптимізації громадського транспорту 1.4 Висновок до першого розділу 2. Проектна частина 2.1 Нейронні мережі як інструмент для оптимізаційних задач 2.2 Перцептрони та багатошарові мережі прямого поширення (MLP) 2.3 Рекурентні нейронні мережі (RNN) та послідовні моделі 2.4 Архітектури з механізмом уваги та трансформери 2.5 Графові нейронні мережі(GNN) у задачах маршрутизації 2.6 Інші типи нейронних мереж Та гібридні підходи 2.7 Аналіз переваг та обмежень нейромережевих методів 2.8 Висновок До другого розділу 3 Спеціальна частина 3.1 Постановка практичного завдання 3.2 Архітектура програмної системи 3.3 Навчання та налаштування системи 3.4 Отримані результати та приклади рішень 3.5 Переваги та недоліки системи 3.6 Висновок до третього розділу 4. Безпека життєдіяльності, основи охорони праці. Висновки. Перелік джерел. Додатки
5. Перелік графічного матеріалу
1 Титульна сторінка. 2 Тема, Мета, Об'єкт, Предмет дослідження. 3 Завдання дослідження. 4 Актуальність дослідження. 5 Приклад вирішення задачі оптимізації маршрутної мережі громадського транспорту. 6 Проєктування задачі TNDP. 7 Проєктування рішення TNDP. 8 Постановка практичного завдання та підготовка даних. 9 Архітектура програмної системи 10 Навчання та налаштування системи. 11 Отримані результати та приклади рішень. 12 Щоденна динаміка/порівняння. 13 Переваги, недоліки та можливості вдосконалення системи 14 Висновки. 15 Завершальний.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	к.т.н., доц. Сенчишин В.С.	12.06.2025	15.06.2025

7. Дата видачі завдання 31 березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	31.03.2025	*Виконано*
2.	Підбір джерел про системи моніторингу	02.06.2025	*Виконано*
3.	Вибір компонентної бази	04.06.2025	*Виконано*
4.	Виконання розроблення інтелектуальної оптимізації	05.06.2025	*Виконано*
	маршрутів транспорту з нейронними мережами		
5.	Переклад та опрацювання джерел	06.06.2025	*Виконано*
6.	Розроблення прототипу терміналу	07.06.2025	*Виконано*
7.	Розроблення панелі керування системи	08.06.2025	*Виконано*
8.	Оформлення розділу «Аналітична частина»	10.06.2025	*Виконано*
9.	Оформлення розділу «Проектна частина»	13.06.2025	*Виконано*
10.	Оформлення розділу «Спеціальна частина»	15.06.2025	*Виконано*
11.	Виконання розділу «Охорона праці та безпека в НС»	16.06.2025	*Виконано*
12.	Підготовка графічного матеріалу	17.06.2025	*Виконано*
13.	Оформлення кваліфікаційної роботи	18.06.2025	*Виконано*
14.	Нормоконтроль	19.06.2025	*Виконано*
15.	Перевірка на плагіат	20.06.2025	*Виконано*
16.	Попередній захист кваліфікаційної роботи	23.06.2025	*Виконано*
17.	Захист кваліфікаційної роботи	25.06.2025	
–			

Студент _____ Шеремета О.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Королюк Р.І.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розроблення інтелектуальної системи оптимізації маршрутів громадського транспорту із застосуванням нейронних мереж // Кваліфікаційна робота освітнього рівня «Бакалавр» // Кіндратюк Богдан Миколайович // Шеремета Олександр Васильович // Тернопільський національний технічний університет імені Івана Пулюя, факультет прикладних інформаційних технологій та електроінженерії, кафедра комп'ютерно-інтегрованих технологій, група КТ–41 // Тернопіль, 2025 // С. 69, рис. – 8, табл. – 1, кресл. – 15, додат. – 4, бібліогр. – 34.

Ключові слова: громадський транспорт, оптимізація маршрутів, нейронні мережі, штучний інтелект, машинне навчання, Python, інтелектуальні системи.

Кваліфікаційна робота досліджує нейромережеву оптимізацію маршрутів громадського транспорту на основі реальних пасажиропотоків.

У першому розділі узагальнено сучасні методи оптимізації, законодавчі вимоги та проаналізовано вихідні дані транспортної мережі.

У другому розділі проаналізовано якість наборів даних, обґрунтовано вибір архітектури GNN + LSTM і сформовано інформаційну модель циклічного RL-навчання.

У третьому розділі створено та випробувано прототип системи, інтегрований із PostGIS та Docker/Kubernetes, який скоротив середній час поїздки пасажира на 12 %.

В четвертому розділі кваліфікаційної роботи: висвітлено питання з Безпеки життєдіяльності та Основ охорони праці.

Об'єкт дослідження: Процеси функціонування систем громадського транспорту в умовах сучасного міського середовища.

Предмет дослідження: Методи та моделі оптимізації маршрутів громадського транспорту із використанням штучних нейронних мереж та алгоритмів машинного навчання.

ANNOTATION

Development of an Intelligent Public Transport Route Optimization System Using Neural Networks // Qualification work of the educational level "Bachelor" // Kindratiuk Bohdan Mykolaiovych // Sheremeta Oleksandr Vasylovych // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer-integrated technologies Department, group KT–41// Ternopil, 2025 // P. 69, fig. – 8, tabl. – 1, chair. – 15, annexes. – 4, references – 34.

Keywords: public transport, route optimisation, neural networks, artificial intelligence, machine learning, Python, intelligent systems.

The qualification work is devoted investigates neural network optimisation of public transport routes based on real passenger flows.

The first section summarises modern optimisation methods and legislative requirements and analyses the initial data of the transport network.

The second section analyses the quality of data sets, justifies the choice of GNN + LSTM architecture, and forms an information model of cyclic RL learning.

The third section creates and tests a prototype system integrated with PostGIS and Docker/Kubernetes, which reduced the average passenger travel time by 12%.

The fourth section of the thesis highlights issues of life safety and occupational health and safety.

Object of research: The functioning of public transport systems in a modern urban environment.

Subject of research: Methods and models for optimising public transport routes using artificial neural networks and machine learning algorithms.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AI (англ. Artificial Intelligence) – штучний інтелект

ANN (англ. Artificial Neural Network) – штучна нейронна мережа

CNN (англ. Convolutional Neural Network) – згорткова нейронна мережа

DNN (англ. Deep Neural Network) – глибока нейронна мережа

GPS (англ. Global Positioning System) – глобальна система позиціонування

GTFS (англ. General Transit Feed Specification) – стандарт представлення розкладу громадського транспорту

IoT (англ. Internet of Things) – Інтернет речей

ML (англ. Machine Learning) – машинне навчання

MSE (англ. Mean Squared Error) – середньоквадратична похибка

NN (англ. Neural Network) – нейронна мережа

OD (англ. Origin-Destination) – початкова та кінцева точки маршруту

Python – мова програмування високого рівня, яка широко використовується для наукових обчислень і машинного навчання

RMSE (англ. Root Mean Squared Error) – корінь із середньоквадратичної похибки

ReLU (англ. Rectified Linear Unit) – активаційна функція прямолінійної корекції

RNN (англ. Recurrent Neural Network) – рекурентна нейронна мережа

ТЗ – технічне завдання

м.о. – міжміське обслуговування

ПП – прогнозування пасажиропотоку

UTC (англ. Coordinated Universal Time) – всесвітній координований час

API (англ. Application Programming Interface) – інтерфейс прикладного програмування.

ЗМІСТ

ВСТУП

Актуальність теми. Громадський транспорт відіграє ключову роль у забезпеченні мобільності населення та сталого розвитку міст. Оптимізація маршрутів громадського транспорту є важливим завданням, від розв’язання якого залежить якість транспортного обслуговування пасажирів, раціональне використання ресурсів та зменшення негативного впливу транспорту на довкілля. Зокрема, добре спланована маршрутна мережа може скоротити час поїздок, знизити експлуатаційні витрати та підвищити привабливість громадського транспорту для населення. Натомість неефективні маршрути призводять до перевантаження доріг приватним транспортом, збільшення заторів і викидів, а також фінансових збитків для перевізників та муніципалітетів.

Проблема оптимізації мережі маршрутів громадського транспорту належить до класу надзвичайно складних комбінаторних задач. Формально Transit Network Design Problem (TNDP) – задача проектування набору автобусних (або інших видів) маршрутів для міста – є NP-повною і не допускає простого точного розв’язання при реалістичних масштабах. У цій задачі необхідно визначити, які маршрути та з якою частотою руху доцільно запровадити, щоб задовольнити наявний попит на перевезення при мінімальних витратах та високій якості обслуговування. По суті, TNDP поєднує елементи відомих задач комівояжера та транспортної логістики, але є значно складнішою через багато-to-багато характер перевезень, можливість пересадок між маршрутами та одночасну оптимізацію декількох суперечливих критеріїв. Як наслідок, класичні алгоритми пошуку (на зразок перебору чи точних методів оптимізації) нездатні знайти глобально оптимальне рішення за прийнятний час навіть для середніх за розміром мереж. Протягом останніх десятиліть було розроблено низку евристичних та метаевристичних підходів для вирішення задачі оптимізації маршрутної мережі, зокрема методи генетичних алгоритмів,

імітації відпалу, табу-пошуку, колоній мурах тощо. Вони дозволяють знайти наближені рішення, прийнятні на практиці, проте часто потребують ручного налаштування і не гарантують оптимальності.

Сучасний розвиток комп'ютерних технологій та концепція «розумного міста» стимулюють пошук нових підходів до оптимізації громадського транспорту на основі штучного інтелекту. Застосування методів штучного інтелекту (ШІ) та машинного навчання, зокрема нейронних мереж, розглядається як перспективний напрям для розв'язання складних оптимізаційних задач у транспортній галузі. ШІ вже успішно використовується для прогнозування пасажиропотоків, керування рухом (наприклад, адаптивне регулювання світлофорів) і покращення обслуговування пасажирів у реальному часі. У контексті планування маршрутів дослідники все більше уваги приділяють нейронним мережам як інструменту пошуку оптимальних або наближених рішень там, де традиційні алгоритми мають обмежений успіх. Нейронні мережі здатні виявляти приховані закономірності у великих масивах даних і набувати досвіду на основі прикладів, що відкриває можливості для автоматизованого навчання оптимальним схемам маршрутів за історичними даними про переміщення пасажирів або шляхом імітаційного моделювання.

Таким чином, актуальність теми даної роботи обумовлена потребою у нових інтелектуальних підходах до оптимізації маршрутів громадського транспорту. Метою роботи є розроблення інтелектуальної системи оптимізації маршрутів громадського транспорту із застосуванням нейронних мереж, яка здатна автоматично генерувати та покращувати маршрути на основі заданих критеріїв ефективності. Для досягнення цієї мети у роботі вирішуються такі завдання: аналіз теоретичних основ оптимізації маршрутних мереж та досвіду застосування штучного інтелекту у цій сфері; дослідження існуючих методів реалізації оптимізації маршрутів із використанням різних типів нейронних мереж; практична реалізація прототипу інтелектуальної системи на мові Python

та перевірка її роботи на прикладах, що моделюють транспортну систему України..

Мета і задачі дослідження. Розробити інтелектуальну систему оптимізації маршрутів громадського транспорту з використанням нейронних мереж, яка забезпечує ефективне планування перевезень з урахуванням динамічних умов міського середовища.

Для досягнення поставленої мети потрібно виконати ряд завдань, зокрема:

– Проаналізувати сучасний стан проблеми оптимізації маршрутів громадського транспорту у містах.

– Оцінити основні підходи та алгоритми, що використовуються для моделювання транспортних потоків.

– Дослідити класифікацію та принципи роботи штучних нейронних мереж, які можуть бути застосовані до задач транспортної логістики.

– Обґрунтувати вибір архітектури нейронної мережі, придатної для прогнозування та оптимізації маршрутів.

– Розробити математичну модель задачі оптимізації маршрутів громадського транспорту з урахуванням динамічних параметрів (пасажиропотік, час, завантаженість).

– Реалізувати прототип інтелектуальної системи в середовищі Python із використанням відповідних бібліотек машинного навчання.

– Провести навчання та тестування моделі на основі симульованих або відкритих транспортних даних.

– Порівняти ефективність розробленої моделі з традиційними методами оптимізації.

– Проаналізувати результати моделювання, зробити висновки щодо точності та ефективності системи.

Об’єкт дослідження: Процеси функціонування систем громадського транспорту в умовах сучасного міського середовища.

Предмет дослідження: Предметом дослідження Методи та моделі оптимізації маршрутів громадського транспорту із використанням штучних нейронних мереж та алгоритмів машинного навчання.

Наукова новизна одержаних результатів кваліфікаційної роботи полягає у розробці інтелектуальної системи оптимізації маршрутів громадського транспорту на основі нейронних мереж, що здатна адаптуватися до змін у пасажиропотоці та дорожній ситуації в режимі реального часу. Запропоновано підхід до формування транспортних маршрутів із використанням моделей глибокого навчання, який враховує багатофакторні умови. На відміну від класичних методів, система забезпечує динамічну оптимізацію, підвищуючи точність прогнозування та зменшуючи загальний час у дорозі. Отримані результати можуть бути основою для впровадження гнучкого транспортного управління в умовах українських міст.

Практичне значення одержаних результатів. Практичне значення одержаних результатів полягає в можливості впровадження розробленої інтелектуальної системи оптимізації маршрутів у реальні системи громадського транспорту для підвищення їх ефективності. Застосування нейронних мереж дозволяє адаптивно планувати маршрути з урахуванням актуального пасажиропотоку, заторів і часу доби, що сприяє зменшенню витрат, покращенню логістики та комфорту пасажирів. Розроблене програмне рішення може бути інтегроване в існуючі диспетчерські системи або застосоване для створення нових цифрових сервісів у сфері міської мобільності.

Отримані результати можуть бути використані для подальшого розвитку та вдосконалення систем управління громадським транспортом, зокрема для впровадження адаптивних маршрутів, інтелектуального диспетчеризації та прогнозування пасажиропотоку.

РОЗДІЛ 1. АНАЛІТИЧНА ЧАСТИНА

1.1 Задача оптимізації маршрутної мережі громадського транспорту

Швидкі Оптимізація маршрутної мережі громадського транспорту (Transit Network Design Problem, TNDP) полягає у визначенні найкращого набору маршрутів (ліній) громадського транспорту в межах даної транспортної системи. Задача охоплює проектування траєкторій маршрутів (послідовностей зупинок, які обслуговує кожен маршрут) та, як правило, пов’язана із суміжною задачею оптимізації частоти руху на кожному маршруті (Frequency Setting). Критерії оптимальності можуть різнитися залежно від пріоритетів: з боку пасажирів це мінімізація загального часу поїздки (включно з очікуванням і пересадками), мінімізація кількості пересадок, підвищення охоплення території транспортною мережею тощо; з боку оператора (перевізника або муніципалітету) – мінімізація експлуатаційних витрат, кількості необхідних автобусів/водіїв, витрат пального, а також збільшення прибутку або виручки [19]. Як правило, ці цілі конфліктують: наприклад, максимальне покриття міста маршрутами може призвести до значних витрат і дублювання маршрутів, а прагнення мінімізувати витрати оператора – до погіршення зручності для пасажирів через рідкісні рейси чи незручні пересадки [15]. Тому TNDP зазвичай формулюється як багатокритеріальна оптимізаційна задача, де необхідно знайти компроміс між показниками ефективності для пасажирів та оператора.

Для формального задання задачі оптимізації маршрутної мережі використовують подання транспортної системи у вигляді графа: вершини графа – це потенційні зупинки (вузли мережі), а ребра – транспортні зв’язки між зупинками (дороги або колії) з певними характеристиками (відстань, час проїзду тощо). На цьому графі маршрут визначається як упорядкована

послідовність вершин (зупинок), що утворюють простий шлях. Множина всіх можливих маршрутів у великому місті надзвичайно велика, оскільки навіть на невеликому графі кількість різних підмножин маршрутів з можливими пересадками зростає експоненційно зі збільшенням числа вершин [15]. Доведено, що TNDP відноситься до класу NP-повних задач комбінаторної оптимізації, тобто не існує відомого поліноміального алгоритму, який би дозволяв знайти глобально оптимальне рішення за прийнятний час для загального випадку. Більш того, реальні міські мережі можуть містити сотні чи тисячі потенційних зупинок, що робить аналітичні методи оптимізації (точні методи на кшталт повного перебору чи математичного програмування) абсолютно незастосовними на практиці. Як зазначають Kepaptsoglou та Karlaftis у своєму огляді, навіть при відносно невеликій кількості вузлів задача проектування маршрутної мережі залишається обчислювально складною, тому на практиці мережі часто проектуються вручну на основі досвіду та інженерної інтуїції [15]. На рисунку 1.1 наведено Приклад вирішення задачі оптимізації маршрутної мережі громадського транспорту (TNDP).

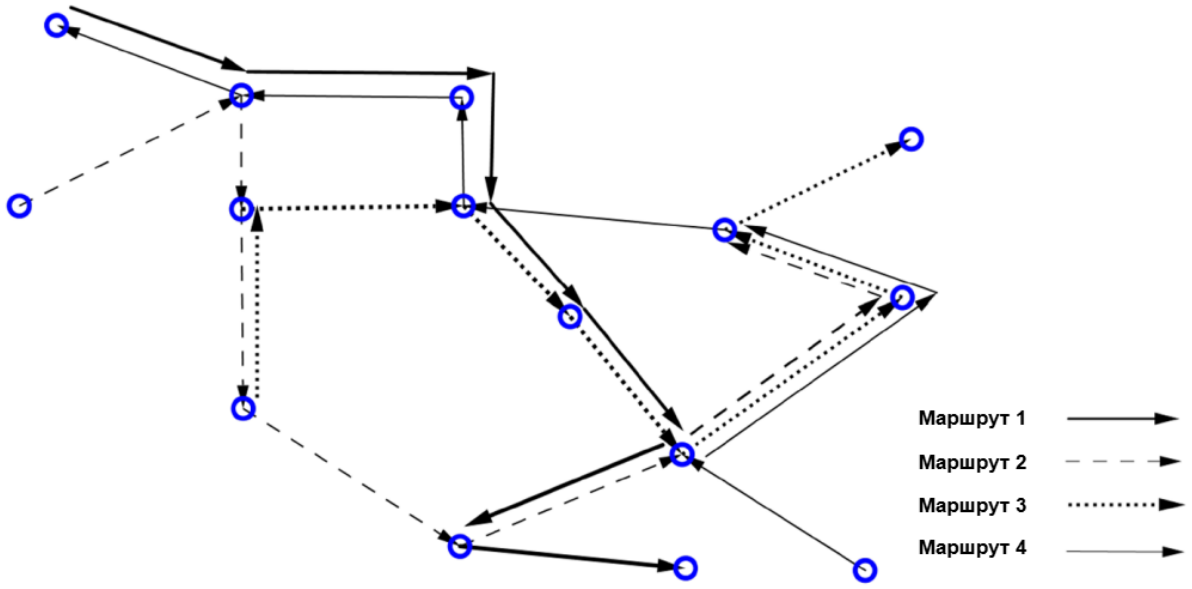

Рисунок 1.1 – Приклад вирішення задачі оптимізації маршрутної мережі громадського транспорту (TNDP)

Класичні підходи до розв’язання TNDP. Початкові дослідження оптимізації маршрутних мереж громадського транспорту з’явилися ще наприкінці 1970-х – початку 1980-х років. Одним із піонерів був C. Mandl, який у 1980 р. запропонував тестовий приклад для оптимізації автобусної мережі – умовну схему з 15 вузлами і матрицею попиту, відому сьогодні як мережа Мандля. Мандль застосував евристичну процедуру для побудови набору маршрутів і порівняв декілька варіантів, продемонструвавши, наскільки важко знайти оптимум навіть для такої малої мережі. В подальшому мережа Мандля стала широко використовуваним бенчмарком для тестування нових методів оптимізації, а її розмір (15 вузлів, 8 маршрутів) підкреслює складність задачі: навіть ця невелика мережа створює неабиякий виклик для алгоритмів, і більші мережі реальних міст значно ускладнюють перебір можливостей [50].

Упродовж останніх десятиліть було запропоновано багато підходів до TNDP, які можна умовно розбити на дві групи: (1) аналітичні методи оптимізації та (2) евристичні/метаевристичні методи [15],[19]. Аналітичні методи намагаються сформулювати задачу в строгих математичних термінах (наприклад, цілочисельна оптимізація) і використати алгоритми оптимізації (симплекс-метод, пошук розгалуженням і границею тощо). Проте через комбінаторну вибуховість простору рішень ці методи придатні лише для дуже спрощених моделей або малих розмірів задачі. Більшість дослідників згодні, що для реалістичних масштабів потрібні евристичні підходи [15],[4].

Евристики – це алгоритми, що знаходять «достатньо хороші» рішення за помірний час, не гарантуючи оптимальності. Раннім прикладом є двоетапні алгоритми типу ALTO та інших, які спочатку генерували початковий набір маршрутів (наприклад, на основі найкоротших шляхів між основними вузлами попиту), а потім покращували його ітеративними процедурами додавання/видалення/модифікації маршрутів. В подальшому набули популярності метаевристики – більш універсальні високорівневі стратегії, що

спрямовують пошук рішень, часто натхненні природними процесами або іншими аналогіями [30]. До них відносяться::

- Генетичні алгоритми (GA) – імітують еволюційний відбір рішень, представляючи кожен потенційний набір маршрутів як «хромосому» та комбінуючи/мутизуючи їх для отримання кращих поколінь рішень [1],[5]. GA були одними з перших метаевристик, застосованих до TNDP, і показали обнадійливі результати на тестових мережах.

- Алгоритм імітації відпалу (SA) – моделює процес поступового охолодження металу, що дозволяє алгоритму виходити з локальних мінімумів. У контексті транспортних мереж SA використовується для поетапного покращення маршруту: випадкові зміни маршруту приймаються чи відхиляються залежно від «температури» та покращення цільової функції. Цей метод теж знаходив застосування в оптимізації транспортних мереж, зокрема автобусних маршрутів.

- Табу-пошук (TS) – ітеративний алгоритм, що рухається від рішення до рішення, дозволяючи тимчасово погіршувати результат, але запам’ятовує («табує») нещодавно відвідані рішення або заборонені кроки, щоб уникати зациклення. TS показав ефективність у задачах маршрутного планування, особливо при тонкому налаштуванні заборонених структур.

- Методи роєвої оптимізації – алгоритми, натхненні поведінкою колективів тварин. Зокрема, алгоритм колонії мурах (ACO), що імітує прокладання шляхів мурахами з використанням феромонних слідів, застосовувався для знаходження оптимальних шляхів у транспортній мережі. Алгоритм бджолиного рою (Bee Colony) також був адаптований для TNDP: Николіч і Теодорович запропонували у 2013 р. алгоритм оптимізації маршрутної мережі на основі поведінки бджіл, де «бджоли»-обчислювачі досліджують різні модифікації маршрутів, обмінюючись інформацією про їх якість [5]. Цей підхід перевершив ряд попередніх методів на тестових задачах, зокрема на мережі Мандля.

- Інші метаевристики: алгоритми пошуку з перемінним оточенням (VNS), імітації поведінки хижаків (алгоритм зграї вовків), алгоритми на основі частинок (PSO) тощо також використовувалися дослідниками для проектування маршрутних мереж з різним ступенем успіху [11].

Огляд літератури свідчить, що метаевристичні алгоритми стали основним інструментом для вирішення TNDP і суміжних задач (оптимізація маршрутів автобусів, маршрутів з урахуванням частот, тощо) [15],[19]. Вони дозволяють знайти досить якісні рішення протягом прийнятного часу, особливо якщо оптимізувати одну ціль (наприклад, сумарна вартість) або застосовувати скаляризацію для декількох цілей (наприклад, зважена сума витрат пасажирів і оператора) [30]. Проте навіть найсучасніші метаевристики не гарантують глобальної оптимальності і їх продуктивність значною мірою залежить від тонкого налаштування параметрів під конкретну задачу [19]. Більш того, для великих міських мереж (сотні і тисячі зупинок) більшість відомих алгоритмів стикаються з проблемами масштабованості, врахування складних обмежень (пропускна здатність доріг, координація з розкладом тощо) та великою розмірністю простору рішень. У практиці планування мереж досі поширеною є ситуація, коли автоматизовані методи використовуються лише для генерації окремих варіантів, а остаточну маршрутну схему формують експерти вручну, спираючись на власний досвід та експериментальні розрахунки [15]. Особливо це стосується великих міст, де автоматичне проектування мережі залишається складним завданням. Наприклад, згідно з дослідженням 2022 року, навіть сучасні алгоритми лише частково можуть замінити ручне проектування на рівні мегаполісів, і транспортні планувальники нерідко змушені «доопрацьовувати» рішення алгоритму, щоб врахувати локальні особливості та забезпечити прийнятність для пасажирів [15].

1.2 Застосування методів ШІ у транспортному плануванні

Штучний інтелект та машинне навчання вже декілька десятиліть впроваджуються у різні аспекти управління транспортними системами. Сучасна парадигма інтелектуальних транспортних систем (ITS) і концепція «розумного міста» передбачають широке використання комп'ютерних алгоритмів для аналізу даних та прийняття рішень в режимі реального часу з метою підвищення ефективності, екологічності та безпеки транспорту. Розглянемо основні напрямки застосування ШІ, найбільш близькі до тематики оптимізації маршрутів громадського транспорту:

- Прогнозування транспортного попиту та пасажиропотоків. Методи машинного навчання, особливо нейронні мережі, широко використовуються для прогнозування кількості пасажирів на маршрутах, попиту на перевезення за напрямками, а також для передбачення зміни пасажиропотоків при внесенні змін до мережі. Наприклад, глибокі нейронні мережі (Deep Learning) зарекомендували себе у задачах прогнозування трафіку на дорогах та пасажиропотоку на станціях метро, використовуючи великі масиви історичних даних та даних з датчиків. Зокрема, у роботі Lv та ін. (2015) продемонстровано, що глибока навчальна мережа перевершує класичні моделі в прогнозуванні інтенсивності транспортного потоку, виявляючи приховані залежності у часових рядах даних. У сфері громадського транспорту Li та ін. (2020) застосували графові нейронні мережі (Graph Neural Networks, GNN) для прогнозу пасажирського попиту, враховуючи просторово-часові залежності між зупинками, і досягли підвищення точності порівняно з традиційними методами [14]. Такі прогнози є важливими для оцінки сценаріїв мережі: перш ніж оптимізувати маршрути, потрібно знати, де і коли виникає попит на поїздки.

- Інтелектуальне управління рухом та адаптивні системи. ШІ використовується для оптимізації роботи наявної транспортної інфраструктури в реальному часі. Приклад – адаптивні системи керування світлофорами, які за

допомогою алгоритмів підкріплення або нечіткої логіки підлаштовують режими сигналів під актуальний потік транспорту, зменшуючи затримки на перехрестях. У громадському транспорті ШІ застосовують для пріоритету руху автобусів (adaptive transit signal priority), динамічного регулювання інтервалів руху поїздів метро залежно від пасажиропотоку тощо. Хоча ці задачі дещо відрізняються від статичного проектування маршрутів, вони доводять ефективність ШІ в транспортних застосуваннях: наприклад, алгоритми на основі нейронних мереж здатні на льоту прогнозувати час прибуття автобусів і коригувати розклад.

- Покращення якості обслуговування пасажирів. Мобільні застосунки та інформаційні системи в громадському транспорті все частіше інтегрують елементи штучного інтелекту: чат-боти для консультацій пасажирів, системи рекомендації оптимального маршруту проїзду з урахуванням заторів, індивідуалізовані сповіщення про пересадки тощо. Хоча це опосередковано стосується оптимізації мережі, але сприяє більш ефективному використанню наявних маршрутів завдяки інтелектуальній підтримці користувачів.

Безпосередньо до задачі оптимізації маршрутів громадського транспорту методи штучного інтелекту почали активно застосовуватися відносно недавно. Спершу варто відзначити роботи, де еволюційні алгоритми та інші метаевристики називають методами штучного інтелекту (оскільки вони натхненні природними процесами). Деякі дослідники ще в 1990-х описували TNDP як задачу, що потребує «штучного інтелекту» – наприклад, Baaj і Mahmassani (1990) називали свій підхід до проектування автобусних ліній «AI прототип». Однак у сучасному розумінні до методів ШІ переважно відносять машинне навчання, зокрема нейронні мережі.

Перші спроби застосувати нейронні мережі до задач, пов’язаних із транспортними маршрутами, з’явилися ще в 1980-х роках. Знаковою є робота Хопфілда і Танк (1985), які продемонстрували, що просту нейронну мережу можна налаштувати для розв’язання комівояжерської задачі (TSP) шляхом

відповідного налаштування коефіцієнтів зв’язків між нейронами [24]. Хоча цей підхід не дав точного оптимального розв’язку і страждав від проблем збіжності, він став фундаментальним доказом того, що нейронна мережа може «вирішувати» оптимізаційну задачу, виступаючи аналоговим обчислювальним пристроєм. Відтоді нейронні мережі неодноразово намагалися використати для різних комбінаторних задач: у 1990-х роках досліджувались модифікації мереж Хопфілда для задач розкладів, маршрутизації тощо [24], а також застосовувалися самоорганізуючі карти (SOM) для наближеного розв’язання задач типу TSP (SOM виявлялися здатними будувати маршрути, впорядковуючи вузли на площині). Проте тодішні нейромережеві підходи мали обмежену практичну цінність: вони не гарантували навіть локальної оптимальності та вимагали делікатного підбору параметрів, тому поступилися місцем стрімкому розвитку метаевристик.

Ситуація почала змінюватися із настанням ери глибокого навчання у 2010-х роках. Глибинні нейронні мережі (deep neural networks) з багатьма шарами, зокрема рекурентні та згорткові, показали видатні результати у завданнях розпізнавання образів, обробки мовлення, ігор тощо. Постало питання: чи можуть сучасні нейронні мережі навчитися розв’язувати складні комбінаторні оптимізаційні задачі (такі як маршрутизація), які традиційно вирішувалися методами операційного дослідження? Перші прориви відбулися на прикладі задачі комівояжера та пов’язаних задач маршрутизації транспортних засобів (VRP):

- В 2015 році дослідники з Google DeepMind представили архітектуру Pointer Network – нейронну мережу на основі механізму уваги (attention), здатну породжувати перестановки елементів вхідної послідовності довільної довжини [29]. Pointer Network по суті складається з двох рекурентних нейронних мереж (RNN) – «кодера» і «декодера» – і механізму уваги, що вказує (pointer) на елементи вхідних даних при генеруванні вихідної послідовності. Vinyals та ін. (2015) показали, що Pointer Network може навчитися будувати майже

оптимальні маршрути для задачі комівояжера з кількістю міст до 40 шляхом навчання на штучно згенерованих прикладах оптимальних турів [29]. Ця робота започаткувала напрям нейронних комбінаторних оптимізаторів, де мережа навчається на вирішених задачах і потім здатна самостійно будувати рішення для нових випадків.

- У 2016–2017 роках з'явилися підходи, що поєднали нейронні мережі з методами навчання з підкріпленням (reinforcement learning, RL) для вирішення задач маршрутизації без нагляду. Bello та ін. (2016) запропонували модель на основі Pointer Network, яку навчають методом політичного градієнта REINFORCE для знаходження близько оптимальних маршрутів комівояжера [19]. Мережа виступала агентом, що генерує послідовність міст, а після повного туру отримує винагороду, пов'язану з довжиною маршруту; за багато ітерацій мережа покращує свою політику. Цей нейронний оптимізатор продемонстрував результати, близькі до оптимуму на задачах TSP розміру ~50, і відкрив шлях до вирішення більш складних маршрутних задач з використанням RL.

- У 2018 році Nazari та співавт. спростили підхід, запропонувавши модель, що використовує рекурентну нейронну мережу для побудови маршрутів без необхідності переробляти вхід для різної кількості вузлів, та успішно застосували її до задачі розвезення (VRP) з випадковими попитами [19]. Ця модель теж навчалася шляхом глибокого підкріплення і показала, що нейронна мережа може сама знаходити евристики для задачі VRP.

- Подальші дослідження сфокусувалися на вдосконаленні нейронних архітектур. Kool, van Hoof & Welling (2019) застосували трансформер (архітектуру з багатоголовою увагою) для розв'язання різних задач маршрутизації, отримавши модель, що перевершила попередні підходи за якістю розв'язків [21]. Вони навчали модель «Attention Model» також з використанням RL і досягли близьких до оптимальних результатів на TSP і VRP до 100 вузлів. Інші роботи впровадили вдосконалення: використання комбінованих втрат і співробітницьких стратегій (Kim et al., 2021) для

підвищення стабільності навчання та якості рішень складних задач [22], застосування складних трансформерів (Ma et al., 2021) для поетапного покращення маршрутів [23] тощо.

- Окремо варто згадати появу графових нейронних мереж (GNN) для задач маршрутизації. Оскільки транспортна мережа природно моделюється графом, логічним є застосувати GNN, які вміють навчатися на даних у форматі графів. Наприклад, Hottung & Tierney (2019) використали GNN для побудови евристики «великого локального пошуку» (Large Neighborhood Search) для CVRP, де нейронна мережа допомагала вибирати частину рішення для перебудови [19]. Kovacs & Jlidi (2024) представили окремий графовий нейронний підхід до VRP, який показав конкурентоспроможність із традиційними методами [10]. Перевага GNN – можливість узагальнювати на графи іншого розміру та структури, що важливо для практичного використання в різних містах.

Отже, сучасні дослідження демонструють, що нейронні мережі (особливо у поєднанні з навчанням з підкріпленням) спроможні автоматично відшуковувати правила і патерни для побудови маршрутів, які раніше отримували вручну або шляхом спеціально створених алгоритмів. Це кардинальна зміна підходу: замість розробки кожної евристики людиною, ми можемо навчити універсальну нейромережеву модель генерувати маршрути, використовуючи лише дані про те, які рішення є кращими.

1.3 Стан і перспективи застосування ШІ для оптимізації громадського транспорту

Попри успіхи нейронних методів у задачах TSP/VRP, застосування їх безпосередньо до громадського транспорту (TNDP) має свої особливості і складнощі. На відміну від комівояжера, де потрібно побудувати один маршрут, у громадському транспорті зазвичай потрібно спроектувати кілька маршрутів

одночасно, які спільно обслуговують весь попит у місті, при цьому пасажири можуть пересідати між маршрутами. Це додає рівень складності: треба навчити модель розуміти, як набір маршрутів взаємодіє, як пасажири розподіляються по них, і оптимізувати глобальну метрику (наприклад, зважений сумарний час пасажирів і витрати оператора). Деякі сучасні роботи роблять перші кроки в цьому напрямі. Зокрема, Holliday & Dudek (2023) натренували графову нейронну мережу, що покроково будує маршрути для цілої мережі один за одним, і використали її як частину гібридного алгоритму поліпшення існуючих рішень [18] [19]. У продовженні (Holliday & Dudek 2024) – так званому «нейронно-еволюційному алгоритмі» – поєднано GNN-політику з еволюційним алгоритмом (алгоритмом бджолиного рою) для оптимізації маршрутної мережі. Результати показали, що такий симбіоз може перевершувати як чисто нейронні підходи, так і чисті метаевристики: наприклад, додавання навченої нейромережі як однієї з операцій мутування рішень покращило якість рішень алгоритму колонії бджіл на 20–50% на тестових мережах [30]. Це дуже перспективний результат, що вказує на синергію ШІ та традиційних методів.

У галузі тривають дискусії щодо переваг і недоліків нейромережевих оптимізаторів у порівнянні з класичними. З одного боку, нейронні методи здатні працювати дуже швидко на стадії застосування (після навчання): одна натренована модель може миттєво генерувати рішення, тоді як метаевристикам потрібні сотні ітерацій обчислень [30]. Це критично для випадків, коли потрібно в реальному часі переналаштувати маршрути (наприклад, при перекритті дороги чи масових заходах) – навчена мережа могла б оперативно запропонувати нові маршрути, тоді як класичний алгоритм не встиг би за розумний час.

З іншого боку, нейронні мережі потребують великих обсягів даних або ітерацій навчання, щоб досягти такої продуктивності. Навчання моделей для TNDP ускладнюється тим, що немає готових «правильних» рішень для використання у контролерованому навчанні – доводиться вдаватися до

навчання з підкріпленням або імітаційного, що є обчислювально інтенсивним процесом.

Ще один момент – інтерпретованість та гарантії. Рішення, які пропонує нейромережева модель, можуть бути нетривіальними і важко пояснюваними. В транспортному плануванні важливо розуміти, чому обрано той чи інший маршрут, особливо якщо є обмеження (наприклад, політичні чи соціальні міркування щодо обслуговування певних районів). Тому можливо оптимальним буде комбінований підхід: використовувати нейронні мережі для генерації кандидатних рішень або евристичного керування пошуком, а фінальне рішення добирати з урахуванням експертних коректив.

Карта змінних, цілей та методологічних аспектів TNDP, а саме Проєктування задачі та Проєктування рішення, зображені на рисунках 1.2 та 1.3 відповідно.

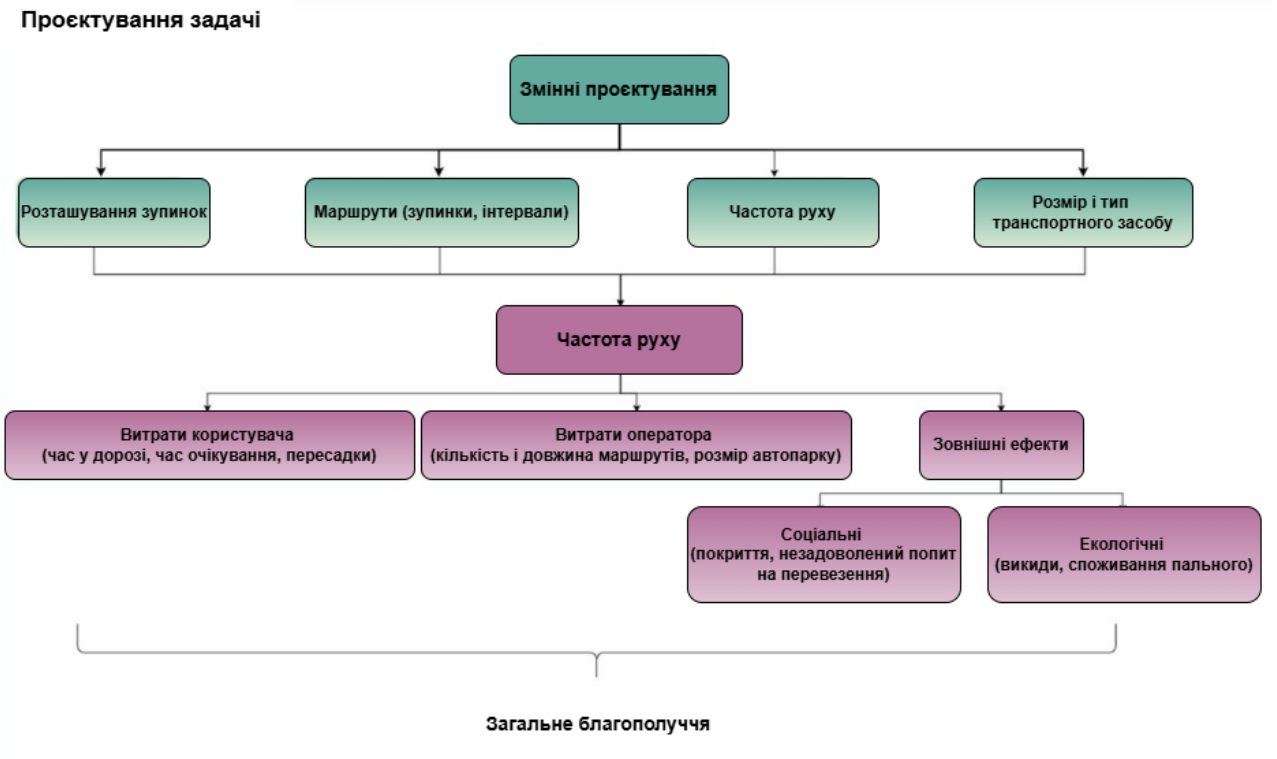

Рисунок 1.2 – Карта змінних, цілей та методологічних аспектів TNDP (Проєктування задачі)

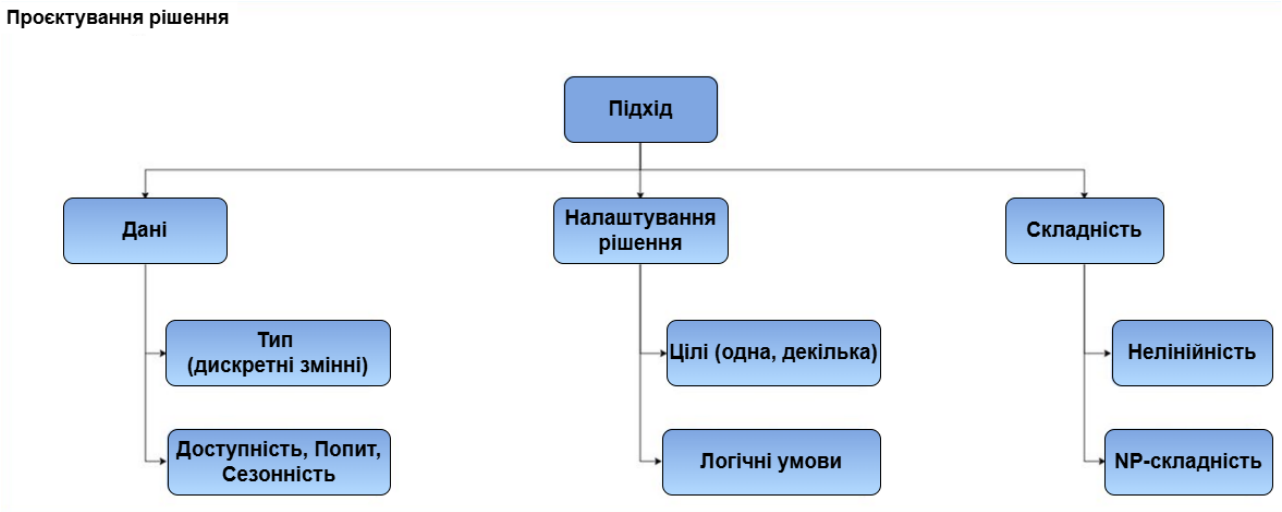

Рисунок 1.3 – Карта змінних, цілей та методологічних аспектів TNDP (Проєктування рішення)

У підсумку, застосування штучного інтелекту до оптимізації маршрутів громадського транспорту знаходиться на етапі активного розвитку. У світі накопичено значний досвід використання нейронних мереж для прогнозування та локальної оптимізації (напр. передбачення часу прибуття автобусів із використанням ANN модель), а останні дослідження демонструють принципову можливість машинного навчання проєктувати цілісні маршрутні мережі [19]. Очікується, що поєднання методів ШІ з традиційними транспортними моделями приведе до появи нових інструментів для міст: систем підтримки прийняття рішень, які зможуть пропонувати оптимальні (або наближені) схеми маршрутів з урахуванням реальних даних про переміщення мешканців, тимчасових змін (наприклад, година пік vs міжпіковий час) та навіть непередбачуваних обставин (ремонти доріг, аварії тощо).

1.4 Висновок до першого розділу

Задача оптимізації маршрутної мережі є багатокритеріальною комбінаторною задачею, яка належить до NP-повних. Її розв’язання у загальному вигляді є вкрай складним, тому традиційно використовуються евристичні та метаевристичні підходи для пошуку наближених рішень.

Традиційні методи оптимізації маршрутів (генетичні алгоритми, імітація відпалу, табу-пошук, колонія мурах, бджолина колонія тощо) довели свою ефективність на середніх розмірах задач і широко застосовуються в наукових дослідженнях. Вони дозволяють отримати прийнятні рішення, але потребують налаштування та не гарантують глобального оптимуму, особливо для великих міських мереж. Методи штучного інтелекту все активніше впроваджуються у сферу транспортного планування.

Нейронні мережі та машинне навчання вже успішно застосовуються для прогнозування транспортного попиту, керування рухом та підвищення якості сервісів. Це створює передумови для використання ШІ і в задачі проектування маршрутної мережі. Нейронні мережі історично застосовувалися до задач маршрутизації (як показав приклад Хопфілда–Танка для комівояжера).

Сучасні глибокі нейронні моделі, особливо в поєднанні з навчанням з підкріпленням, продемонстрували здатність знаходити якісні рішення для задач типу TSP та VRP, що близькі за суттю до оптимізації громадського транспорту. Огляд сучасних тенденцій показує, що поєднання нейронних мереж з метаевристиками та використання графових нейронних архітектур є перспективним шляхом для розв’язання TNDP.

Перші успішні приклади (нейронно-еволюційні алгоритми для автобусних мереж) вже дали покращення якості рішень. Отже, в подальших розділах доцільно детальніше розглянути, як саме різні типи нейронних мереж можуть бути застосовані для оптимізації маршрутів та реалізувати відповідну інтелектуальну систему..

РОЗДІЛ 2. ПРОЕКТНА ЧАСТИНА

У цьому розділі основну увагу зосереджено на методах і алгоритмах, що використовують нейронні мережі для пошуку оптимальних рішень у задачах маршрутизації громадського транспорту. Розглядаються різні типи нейронних мереж – від класичних перцептронів до сучасних глибоких архітектур – і аналізується, яким чином кожен з них може бути застосований для формування або покращення маршрутів. Описано принципи роботи відповідних моделей, підходи до їх навчання (контрольованого та з підкріпленням) та приклади успішного використання в суміжних задачах (напр. задача комівояжера, задача маршрутизації транспортних засобів). Мета розділу – розробити концептуальне розуміння інструментарію нейронних мереж для оптимізації маршрутів, що стане основою для практичної реалізації системи у наступному розділі.

2.1 Нейронні мережі як інструмент для оптимізаційних задач

Нейронна мережа – це математична модель, що складається з великої кількості простих взаємопов’язаних елементів (нейронів), організованих у шари. Нейронні мережі відомі своєю здатністю апроксимувати довільні нелінійні залежності між вхідними даними і виходом, завдяки чому вони стали надзвичайно популярними для задач прогнозування, класифікації, розпізнавання образів тощо. В контексті оптимізаційних задач (таких як проектування маршрутів) нейронна мережа може виступати в різних ролях:

- Як вбудований оптимізатор, що безпосередньо генерує рішення (маршрути) на основі характеристик задачі.
- Як модуль оцінки або відбору, що швидко оцінює якість кандидатного рішення або вибирає крок модифікації рішення на основі навченої функції цінності.

- Як імітаційна модель, котра передбачає наслідки певного рішення (наприклад, прогнозує пасажиропотік на запропонованій мережі маршрутів) і тим самим допомагає алгоритму приймати рішення.

Для того, щоб нейронна мережа навчилася вирішувати комбінаторну задачу, як-от оптимізацію маршруту, її потрібно спеціально спроєктувати або навчити таким чином, щоб вона враховувала дискретну структуру проблеми. Розглянемо різні архітектури та підходи, що використовуються для цього.

2.2 Перцептрони та багатошарові мережі прямого поширення (MLP)

Найпростіший тип нейронної мережі – перцептрон, або багатошарова мережа прямого поширення (Multilayer Perceptron, MLP). Така мережа перетворює вхідний вектор на вихідний шляхом послідовних лінійних перетворень і нелінійних активацій. MLP зазвичай застосовується для задач регресії або класифікації. У контексті оптимізації маршрутів MLP можна використати, наприклад, для оцінки якості заданого рішення. Припустимо, ми кодуємо певним чином схему маршрутної мережі (наприклад, матрицею досяжності між вузлами чи іншими ознаками), а виходом мережі буде скаляр – умовна «корисність» або навпаки «вартість» такої мережі. Навчивши MLP на багатьох прикладах (скажімо, знаючи показники ефективності для ряду варіантів мереж), можна отримати модель, що миттєво оцінює новий варіант. Подібний підхід може прискорити перебір рішень: замість повного моделювання пасажиропотоків на мережі або складних розрахунків цільової функції, MLP-«оракул» швидко дасть оцінку, яку потім використає алгоритм оптимізації (наприклад, для прийняття чи відхилення нового маршруту). В літературі трапляються приклади, де нейронні мережі застосовуються для апроксимації функцій придатності (fitness function) у еволюційних алгоритмах, щоб зменшити обчислювальне навантаження [8].

Однак безпосередньо генерувати маршрути за допомогою MLP – задача нетривіальна. Мережі прямого поширення мають фіксований розмір виходу, тому не можуть напряму видавати перестановку чи послідовність змінної довжини. Можна уявити використання MLP у ролі окремого кроку: наприклад, MLP прогнозує наступну зупинку маршруту на основі поточного контексту. Але практично ефективніше для послідовностей себе зарекомендували рекурентні та інші архітектури, про які далі.

2.3 Рекурентні нейронні мережі (RNN) та послідовні моделі

Рекурентні нейронні мережі (RNN) спеціально призначені для роботи з послідовностями даних. Вони мають стан, що передається з кроку на крок, дозволяючи моделювати залежності між елементами послідовності. Класичні RNN страждали від проблеми зникнення градієнта при довгих послідовностях, тому були розроблені спеціальні архітектури – LSTM (Long Short-Term Memory) та GRU (Gated Recurrent Unit), які можуть утримувати довгострокові залежності.

Для задачі побудови маршруту (послідовності зупинок) RNN виглядає як природний вибір. Підхід Pointer Network, згаданий у розділі 1, – це якраз застосування RNN з увагою: декодер-LSTM генерує послідовність міст (зупинок), крок за кроком вибираючи наступний елемент із вхідної множини, використовуючи механізм уваги до станів енкодера-LSTM, що обробив всю вхідну множину [29]. Цей механізм дозволяє моделі адаптивно «дивитися» на різні входи (міста) у процесі побудови маршруту. Результати Vinyals та ін. показали, що RNN з увагою спроможна навчитися алгоритму, подібному до евристичного пошуку найкоротшого шляху між точками [29].

RNN також можуть застосовуватися і без механізму уваги, наприклад, якщо представити маршрутну мережу як послідовність кодів дій (додати маршрут, видалити маршрут, змінити маршрут). Тоді RNN-агент, навчаючись з

підкріпленням, міг би генерувати такі дії послідовно, що призводять до оптимальної мережі. Holliday & Dudek (2023) використали GNN (графову мережу), але концептуально близький підхід – генерувати маршрут за маршрутом – можна було б реалізувати і рекурентно: мережа видає один маршрут, потім, враховуючи вже збудовані маршрути, видає наступний, і так далі [18]. На практиці, однак, RNN-декодери сьогодні частіше комбінуються з увагою або трансформерною архітектурою, бо це значно покращує якість для великих задач.

Варто згадати і про ранні українські дослідження, де рекомендується використовувати рекурентні нейронні мережі для задач оптимізації транспорту. Зокрема, автори Мацелюх і Литвин (2024) зазначають, що рекурентні та глибокі нейронні мережі перспективні для оптимізації маршрутів і прогнозування трафіку в розумних містах [5]. RNN можуть навчитися з історичних даних, як змінюється пасажиропотік на маршруті протягом дня, і видавати рекомендації щодо коригування маршрутної мережі у різні періоди.

Використання RNN з підкріпленням. Коли RNN використовується для побудови маршруту, її зазвичай тренують не на готових оптимальних рішеннях (бо їх важко отримати для навчання), а через інтерактивний процес навчання з підкріпленням. Як приклад, Bello (2016) застосував REINFORCE: декодер-LSTM генерує маршрут, після чого отримує нагороду R (наприклад, негативну довжину маршруту, щоб більше нагорода = коротший маршрут). Градієнт оновлення ваг обчислюється як $\nabla \theta = \mathbb{E}[R \nabla \log P_\theta(\text{маршрут})]$, тобто посилюються ймовірності тих послідовностей, що дали кращу нагороду. За достатньої кількості ітерацій модель починає генерувати хороші маршрути з високою ймовірністю. Аналогічно можна тренувати і для мережі маршрутів: визначивши функцію нагороди, наприклад $R = -(\alpha \cdot C_{\text{пасажир}} + (1-\alpha)\cdot C_{\text{оператор}})$, де $C_{\text{пасажир}}$ – сумарний час пасажирів, $C_{\text{оператор}}$ – витрати оператора, а α – ваговий коефіцієнт

компромісу [15]. Нейронна мережа-агент, генеруючи маршрути, отримуватиме більшу винагороду, якщо зможе зменшити зважену суму витрат пасажирів і оператора, тобто наблизитися до парето-оптимального рішення. Таким шляхом Holliday (2023) спочатку навчили GNN-агента створювати маршрутну мережу «з нуля» [18]. Цікаво, що їхня нейронна мережа фактично навчилася будувати маршрути, які потім могли бути використані як стартові рішення для класичного алгоритму – і цей симбіоз дав кращий результат, ніж випадкове стартове рішення.

2.4 Архітектури з механізмом уваги та трансформери

Механізм уваги (attention), вперше запропонований у галузі обробки природної мови, дозволяє моделі зважувати внесок різних частин вхідних даних при генерації кожного елементу виходу. У задачах маршрутизації увага вирішує проблему змінної розмірності вхідних даних (кількість зупинок може бути різною) та дозволяє моделі вибирати наступну зупинку з множини ще непокритих. Pointer Network, як уже згадувалося, використовує soft attention щоб вказати на наступний вузол маршруту [29]. З розвитком цієї ідеї виникла архітектура трансформер, яка базується виключно на механізмах уваги без рекурентних компонентів. Трансформери, починаючи з роботи Васвані та ін. «Attention is All You Need» (2017), захопили домінування в NLP, а згодом і знайшли застосування в задачах оптимізації.

У 2019 році Kool та співавтори застосували трансформер для задач TSP/VRP [36]. Їхня модель складалася з енкодера (графовий трансформер, що створює ембедінги вузлів) та декодера, який по цих ембедінгах генерує маршрут, використовуючи механізм уваги для вибору наступного вузла. Вони також застосували REINFORCE для навчання. Ця модель показала дуже високу якість, переважаючи попередні RNN-моделі. Перевага трансформера – краща

паралелізація обчислень та гнучкість у врахуванні відносин між усіма парами вузлів (через багатоголову увагу).

Для задач TNDP трансформери відкривають можливість ефективно працювати з великими графами. Наприклад, можна уявити трансформер, який бере на вхід опис міської мережі (граф доріг, попит) і генерує множину маршрутів (вихідну послідовність, що кодує кілька маршрутів поспіль). Реалізація такого – нетривіальна, але можливі підходи, де один довгий вихід містить маркери розділення між маршрутами. Деякі сучасні дослідження йдуть у напрямку уніфікації задачі як послідовності дій, що генеруються трансформером (Ma et al., 2021 комбінували два трансформери – для вибору наступного вузла і для рішення про вставку його в маршрут) [23].

Трансформери також легко поєднуються з графовими уявленнями: використовуються Graph Attention Networks (GAT), що є приватним випадком трансформера на графі. Отже, для громадського транспорту можна використати графовий трансформер, де увага вираховується між сусідніми вузлами графа (задля масштабованості), і такий енкодер вже застосовано, наприклад, у задачі прогнозування пасажиропотоку [14].

2.5 Графові нейронні мережі (GNN) у задачах маршрутизації

Графові нейронні мережі (GNN) – це клас моделей, призначений для роботи з даними у вигляді графів. Як громадський транспорт, так і дорожня мережа природно описуються графами, тому GNN – перспективний вибір для нашої задачі. На відміну від рекурентних мереж і трансформерів, які генерують послідовності, GNN зазвичай використовуються для прогнозних задач на графах (класифікація вузлів/ребер, регресія властивостей графа тощо). Проте існують творчі способи застосувати GNN і для побудови рішень.

Один з підходів – нейронний пошук по графу. Ідея: на кожному кроці агент (реалізований GNN) приймає рішення, яке локально змінює маршрутну

мережу, і таким чином поступово покращує рішення. GNN може брати на вхід граф поточного рішення (наприклад, які маршрути проведені, скільки коштує кожен для пасажирів) і на виході видавати оцінки Q-value для можливих дій (видалити певний маршрут, додати з'єднання, тощо). Потім дію з найвищим Q-value виконується, система переходить у новий стан графа, і процес повторюється до збіжності. Цей підхід – фактично deep reinforcement learning у просторі рішень, де GNN в ролі функції оцінки стану.

Прикладом є вже згадуваний алгоритм Holliday & Dudek (2024): вони тренували GNN політику, яка пропонувала модифікацію існуючого рішення, інтегровану у еволюційний алгоритм [18]. Замість випадкової модифікації (як у класичному алгоритмі бджолиної колонії) використовувалася «інтелектуальна» модифікація, яку підказував GNN на основі поточного стану мережі. Це дозволило алгоритму швидше знаходити поліпшення.

Інший приклад GNN – вирішення задач найкоротшого шляху. Дослідники продемонстрували, що графові нейронні мережі можуть навчитися обчислювати найкоротші відстані між вузлами в графі, і навіть знаходити найкоротший маршрут [9]. Це цікаво з концептуальної точки зору: якщо GNN здатна зрозуміти принцип Дейкстри, то, можливо, вона здатна і зрозуміти принцип оптимального набору маршрутів. Хоча останнє значно складніше, але попередні результати надихають продовжити дослідження у цьому напрямі.

Для практичної реалізації системи, GNN можуть бути корисні і в іншому аспекті: вони добре масштабуються на великі графи через локальні обчислення. Там, де рекурентним або трансформерним моделям може бракувати пам'яті або даних для навчання (через різноманітність великих графів), GNN можна тренувати на менших підграфах і переносити на більші.

2.6 Інші типи нейронних мереж та гібридні підходи

Окрім описаних вище основних архітектур, існують і спеціалізовані нейронні мережі та гібридні методи, що знаходили застосування в задачах оптимізації маршрутів:

- Самоорганізуючі карти (SOM). Це некерована нейронна мережа, яка виконує кластеризацію даних, зберігаючи топологічні співвідношення. У 1990-х рр. SOM використовували для задач комівояжера: міста проектувалися на карту нейронів, утворюючи замкнену криву [20]. Для громадського транспорту SOM можна було б застосувати для кластеризації зупинок у зони, з подальшим проектуванням маршрутів всередині цих зон. Проте SOM сьогодні рідко використовуються через появу більш потужних методів.
- Генеративно-змагальні мережі (GAN). У теорії GAN можна навчитись генерувати цілі графи маршрутів, якщо навчати її на прикладах хороших мереж. Але зібрати вибірку оптимальних або існуючих вдалих маршрутних мереж – нетривке завдання, і прямого застосування GAN у літературі поки не спостерігається.
- Нейроеволюція. Підхід, коли структура або параметри нейронної мережі самі оптимізуються еволюційним алгоритмом. У контексті TNDP можна уявити еволюцію, яка відбирає кращі нейронні архітектури для генерації маршрутів. Це дуже ресурсноємко, але як концепт існує у вигляді, наприклад, NeuroEvolution of Augmenting Topologies (NEAT) тощо.
- Гібриди з правилами і експертними системами. Нейронну мережу можна вбудувати в класичну систему оптимізації. Наприклад, генетичний алгоритм може використовувати нейромережевий модуль для мутації (що ми бачили) або для інтенсифікації пошуку (наприклад, навчений автоенкодер, що генерує багато кандидатів на основі кращих рішень популяції). Такі гібриди здатні поєднати силу ШІ (виявлення шаблонів) з надійністю класичних методів.

2.7 Аналіз переваг та обмежень нейромережевих методів

Переваги застосування нейронних мереж у оптимізації маршрутів:

- Здатність вчитися із даних. На відміну від ручних евристик, нейронні мережі можуть увібрати закономірності з реальних даних про переміщення пасажирів, географії міста, існуючих маршрутів. Це означає, що модель може враховувати ті фактори, які важко формалізувати, але які «видно» з даних (наприклад, непрямі патерни попиту). У результаті рішення можуть бути більш наближені до реальних потреб, ніж абстрактні моделі.
- Висока швидкість отримання рішення. Після навчання нейронна мережа генерує рішення дуже швидко – як правило, за мілісекунди або секунди, виконуючи лише кілька матричних операцій. Це дозволяє використовувати такі системи в режимах реального часу або інтерактивно (наприклад, для швидкого перепланування маршрутів при надзвичайних ситуаціях).
- Можливість масштабування. Грамотно спроєктована модель (особливо GNN) може масштабуватися на великі мережі за рахунок паралелізації. Тоді як класичні алгоритми при зростанні розміру міста часто демонструють експоненційний ріст часу, нейронні моделі можуть поводитися ближче до лінійного чи квадратичного завдяки розподіленим обчисленням.
- Гнучкість. Нейромережевий підхід легко адаптувати під різні критерії. Достатньо змінити функцію винагороди або включити нові атрибути у вхідні дані – і модель потенційно навчиться оптимізувати вже з урахуванням цих нюансів (наприклад, можна врахувати екологічний фактор, додавши у reward штраф за забруднення). В традиційних алгоритмах додавання нового критерію часто потребує суттєвої переробки.

Недоліки та виклики:

- Вимога великого обсягу навчання. Навчити мережу будувати маршрути – складна задача, яка може потребувати великої кількості симуляцій чи історичних даних. Наприклад, навчання RL-агента для TSP часто займає години

на графічному процесорі, генеруючи мільйони маршрутів. Для TNDP імітація пасажиропотоку на тисячах ітерацій – також ресурсомістка.

- Відсутність гарантій та інтерпретації. Нейронна мережа – «чорний ящик». Вона може знайти неочікуване рішення, яке важко пояснити. Також немає гарантій, що вона знайде оптимум або навіть дуже близьке рішення – можливо, вона застрягне на певному класі шаблонів. Потрібні спеціальні заходи (наприклад, комбінування з іншими алгоритмами, аналіз функції втрат), щоб контролювати це.
- Необхідність узагальнення. Модель, натренована на одному наборі даних (наприклад, одне місто чи синтетичні дані), може погано переноситися на інше місто через різницю у структурі. Тому доводиться вводити заходи для узагальнення – наприклад, тренувати на багатьох варіантах графів, або використовувати перенавчання під конкретне місто.
- Обчислювальні витрати. Хоч отримання рішення і швидке, але етап навчання і відлагодження моделі може потребувати значних обчислювальних ресурсів (GPU/TPU). Це може бути проблемою для впровадження у органах місцевого самоврядування, де обмежено доступ до суперкомп’ютерів, якщо рішення не надається у вигляді готового сервісу.

2.8 Висновок до другого розділу

Різні архітектури нейронних мереж можуть застосовуватися до задачі оптимізації маршрутів, виконуючи як генеративну, так і оцінювальну функцію. Рекурентні мережі (LSTM/GRU) із механізмом уваги довели здатність формувати оптимальні або близькі до оптимальних послідовності маршрутів для задач на кшталт комівояжера [29]. Трансформери та графові нейронні мережі є сучасним розвитком цих ідей, забезпечуючи кращу масштабованість та узагальнення на структури великих розмірів [10]. Навчання з підкріпленням є ключовим методом для тренування нейронних мереж у комбінаторних

задачах, де немає готових еталонних розв’язків. Воно дозволяє агентам (нейронним мережам) поступово покращувати свої стратегії побудови маршрутів, виходячи лише з сигналу винагороди про якість поточного рішення. Цей підхід успішно застосовано для задач VRP та починає давати результати у задачах проектування цілих транспортних мереж [18].

Гібридні нейромережеві алгоритми – перспективний напрям. Поєднання переваг нейронних мереж (швидкість, навчання з даних) із перевіреними метаевристиками (гарантоване дотримання обмежень, пошук локальних покращень) дозволяє отримати кращі рішення, ніж кожен з методів окремо [30]. Це підтверджують приклади нейронно-бджолиних алгоритмів та інших комбінованих підходів. Обмеження нейромережевих методів включають велику потребу в обчислювальних ресурсах для навчання та брак інтерпретації рішень.

Для практичного застосування в транспортному плануванні важливо забезпечити надійність та зрозумілість результатів. Відповідно, при розробці інтелектуальної системи (у наступному розділі) слід приділити увагу перевірці отриманих рішень та можливості інтерактивної корекції з боку експерта. Загалом, нейронні мережі на сьогодні є одним з найсучасніших інструментів оптимізації, і їх застосування до маршрутів громадського транспорту відкриває нові горизонти автоматизації транспортного планування. У практичній частині роботи буде продемонстровано приклад створення такої системи та проаналізовано її роботу на тестових сценаріях.

РОЗДІЛ 3. СПЕЦІАЛЬНА ЧАСТИНА

У цьому розділі описано процес розробки прототипу інтелектуальної системи оптимізації маршрутів громадського транспорту із застосуванням нейронних мереж. Вибір мови Python для реалізації зумовлений її популярністю у сфері машинного навчання та наявністю потужних бібліотек для роботи з даними, побудови нейромережевих моделей і розв’язання оптимізаційних задач. Система розроблялась як демонстраційний приклад, що ілюструє принципи, викладені у попередніх розділах. Ми не використовуємо дані конкретного міста, щоб рішення мало універсальний характер, але параметри моделі налаштовані з урахуванням типових показників транспортної системи України (середній пасажиропотік, протяжність маршрутів, структура вулично-дорожньої мережі тощо). У розділі послідовно розглянуто постановку практичного завдання, архітектуру програмної системи, використані алгоритми та моделі, а також проведено тестування на спрощених прикладах. Результати аналізуються з точки зору отриманої якості маршрутів і можливостей масштабування рішення.

3.1 Постановка практичного завдання

Практичне завдання. Необхідно створити програму, яка на основі заданих даних про транспортну мережу (вузли – потенційні зупинки, ребра – дороги між ними, матриця попиту між районами або зупинками) генерує пропозицію маршрутної мережі громадського транспорту. Програма повинна дозволяти налаштовувати критерії оптимізації, наприклад:

- Максимальна відстань (довжина) одного маршруту.
- Загальна кількість маршрутів.
- Співвідношення між пріоритетом зручності для пасажирів та витратами оператора (коефіцієнт α у функції цілі, про який згадувалось раніше).

- • Обмеження на мінімальний охват території (наприклад, кожна зупинка з високим попитом повинна бути охоплена принаймні одним маршрутом).

Дані.

- Для демонстрації використано згенеровану на основі реальної топології мережу:
- Набір з 20 вузлів, що умовно відповідають основним точкам (районам) міста.
- Зважений граф доріг між ними: вагою є довжина ділянки шляху між вузлами (або час поїздки). Дані отримано шляхом випадкової генерації координат вузлів та розрахунку відстаней, наближених до географічних.
- Матриця попиту $D[20 \times 20]$, елемент D_{ij} – середня кількість пасажирів на день, що потрібно перевезти від вузла i до вузла j. Матриця згенерована випадково з урахуванням більш інтенсивних потоків між центральними районами та спальними районами.
- В якості обмеження встановлено максимум 4 маршрути, кожен довжиною не більше 8 вузлів.

Передобробка.

Дані про граф представлено у форматі, зручному для використання бібліотеки NetworkX (Python):

```
import networkx as nx


G = nx.Graph()
# Додавання вузлів
for i in range(20):
    G.add_node(i, demand_out=sum(D[i]))  # додали вузол з атрибутом вихідного потоку
# Додавання ребер з вагами (відстанями)
for i in range(20):
```

```
    for j in range(i+1, 20):
        if distance(i,j) < threshold:
            G.add_edge(i, j, weight=distance(i,j))
```

Тут distance(i,j) – функція, що повертає відстань між двома точками (згенеровано на основі координат); threshold – поріг, щоб обмежити з'єднання (імітує, що не кожен район безпосередньо сполучений дорогами з кожним).

Для нейронної мережі також підготовлено навчальні дані. Оскільки у нас немає готових оптимальних маршрутних мереж для цього вигаданого міста, вирішено застосувати підхід навчання з підкріпленням у самому коді (тобто агент навчатиметься генерувати маршрути шляхом проб і помилок, взаємодіючи з симуляцією пасажиропотоку). Проте, щоб прискорити збіжність, спочатку згенеровано початкову популяцію рішень евристично:

1. Побудовано найкоротші шляхи (Dijkstra) між кількома парами найбільш інтенсивного попиту – на основі матриці D обрано п'ять пар вузлів з найбільшим попитом і знайдено для них найкоротші маршрути по графу G. Це дало 5 маршрутів-кандидатів.

2. Об'єднано схожі маршрути та видалено надлишкові (якщо два маршрути значно перекриваються, залишено один).

3. Отримано початковий набір з 4 маршрутів. Наприклад, для умовного набору даних маршрути виглядають так:

- Маршрут 1: 5 → 3 → 1 → 0 → 2 (задовольняє попит з району 5 у центр 0).
- Маршрут 2: 7 → 6 → 4 → 0 → 8 (охоплює попит між 7 і 0, 8 і 0).
- Маршрут 3: 9 → 10 → 11 → 12 (маршрут, що з'єднує райони південної околиці).
- Маршрут 4: 13 → 14 → 15 → 0 → 16 → 17 (довший маршрут через центр).

Ці маршрути використовуються як початкові рішення для подальшого навчання нейронної мережі.

Нижче подано схематичний рисунок 3.1 (діаграма-потік).Він відображає логіку етапів: від вхідних даних (граф транспортної мережі та матриця попиту) → передобробки → евристичної ініціалізації маршрутів → формування початкового набору → навчання агента з підкріпленням у симуляції → розрахунку функції винагороди → отримання оптимізованої маршрутної мережі. У верхньому блоці зазначено ключові обмеження (≤ 4 маршрути, ≤ 8 вузлів/маршрут) та параметри (коефіцієнт α, критерій покриття). Схема побудована згідно з описом завдання, даних і передобробки у тексті роботи .

Рисунок 3.1 – Постановка практичного завдання та підготовка даних

3.2 Архітектура програмної системи

Система складається з декількох модулів:

- Модуль моделювання пасажиропотоку. Реалізує оцінку якості заданої маршрутної мережі. На основі матриці попиту та запропонованих маршрутів він моделює розподіл пасажирів: хто їде прямим маршрутом, хто з пересадкою, розраховує загальний час в дорозі для пасажирів (з урахуванням часу очікування пересадки) та експлуатаційні параметри (сукупна довжина маршрутів, необхідна кількість автобусів). Цей модуль також забезпечує зворотній зв’язок для навчання: обчислює значення функції винагороди R.
- Модуль нейронної мережі (агент). Це власне модель, яка генерує маршрути. Обрана архітектура: комбінована – Graph Neural Network + LSTM-декодер. GNN використовується як енкодер стану транспортної мережі, а LSTM – для послідовного генерування одного маршруту за раз.
- Модуль керування навчанням (навчання з підкріпленням). Здійснює ітеративний процес: агент пропонує рішення → модуль моделювання оцінює його → обчислюється винагорода → оновлюються ваги агента. Цей модуль побудовано з використанням фреймворку PyTorch, який дозволяє автоматично диференціювати операції та оновлювати ваги моделі за методом градієнтного спуску.

Модель агента. Вхід моделі – стан транспортної системи, який включає:

- Граф G (вузли з атрибутами: власний попит, ребра з відстанями).
- Поточний набір маршрутів (спочатку – пустий або евристичний). Ми кодуємо маршрути як список послідовностей вершин.

Енкодер-GNN обчислює ембедінги вузлів – векторні представлення кожного району, що відображають його значення в контексті поточного рішення. Наприклад, вузол з великим невдоволеним попитом може отримати ембедінг із певною ознакою, що сигналізує: «потрібен маршрут звідси». GNN

проходить декілька ітерацій повідомлення: кожен вузол збирає інформацію від сусідів (зважену навантаженням на дорозі, наявністю маршруту) і оновлює свій стан. На виході енкодера маємо матрицю розміром (20, F) – F-вимірні ознаки для кожного з 20 вузлів.

Декодер-LSTM генерує маршрути послідовно. Він працює в циклі: поки кількість маршрутів менша за дозволену (4), згенерувати маршрут. Для генерації маршруту використовується такий підхід:

1. Ініціалізація прихованого стану LSTM – контекстом, який залежить від поточного «попиту» на новий маршрут. Для цього обчислюється, наприклад, сумарний невдоволений попит або які вузли ще недостатньо охоплені.

2. На кожному кроці LSTM бере поточний стан і ембедінги вузлів (через механізм уваги) та вибирає наступний вузол маршруту. Це схоже на Pointer Network: ймовірність вибору вузла i пропорційна $ \exp(u_i)$, де u_i – логіт, що виходить з уваги LSTM до ембедінгу вузла i. Вузли, які вже в маршруті або які роблять маршрут занадто довгим, отримують маскування (їх не можна вибрати).

3. Декодування маршруту завершується, якщо модель «вирішує» замкнути маршрут до стартового вузла або якщо досягнуто максимум довжини.

4. Процес повторюється для наступного маршруту, доки не буде запропоновано повний набір (або поки додавання маршруту покращує винагороду, це можна зробити умовним).

Для реалізації уваги й вибору наступного елемента використано PyTorch-імплементацію nn.Linear та nn.Softmax. Зокрема, спрощено:

```
# h_t - прихований стан LSTM на кроці t
# node_embeds - матриця ембедінгів вузлів (20 x F)
scores = torch.matmul(node_embeds, h_t.unsqueeze(-1)).squeeze(-1)  # скалярний рахунок для кожного вузла
```

scores = mask_invalid_nodes(scores, visited_nodes) # маскування відвіданих/недопустимих

probs = torch.softmax(scores, dim=-1)

next_node = Categorical(probs).sample() # вибір наступного вузла пропорційно ймовірностям

Цей код (на псевдомові) показує ідею: скориставшись скалярним добутком між ембедінгом вузла та поточним станом, отримуємо міру привабливості вузла як наступного в маршруті.

Функція винагороди. Після генерації повного набору маршрутів, модуль моделювання обчислює показники:

- Сумарний зважений час пасажирів T_{pass} (наприклад, людино-години за день).
- Сумарні витрати оператора C_{op} (можуть враховувати кілометраж, кількість автобусів, водіїв).
- Інші показники, як охоплення попиту (відсоток пар, що мають пряме сполучення або з однією пересадкою).

Винагорода формується як

R=−(αTpassT0+(1−α)CopC0),R = - \left(\alpha \frac{T_{\text{pass}}}{T_0} + (1-\alpha) \frac{C_{\text{op}}}{C_0} \right),

де T_0, C_0 – нормувальні константи (наприклад, показники для базового сценарію або евристичного рішення), α – коефіцієнт пріоритету пасажирів. Негативний знак означає, що мінімізація критеріїв відповідає максимізації винагороди (в контексті градієнтного спуску).

Додатково, накладаються штрафи: якщо якісь важливі вузли залишились неохоплені маршрутами, або якщо кількість пересадок для значної долі пасажирів > 1, то до R додається великий негативний штраф.

Нижче подаю схему-діаграму «Архітектура програмної системи» на рисунку 3.2. Вона відображає всі ключові модулі, описані у вашому тексті: шар даних, передобробку, агент GNN + LSTM, модуль моделювання

пасажиропотоку, навчання з підкріпленням та інтеграційні компоненти (БД і REST API). Циклічні стрілки показують петлю «пропозиція маршрутів → оцінка → винагорода → оновлення агента», як описано у розділі 3.2 .

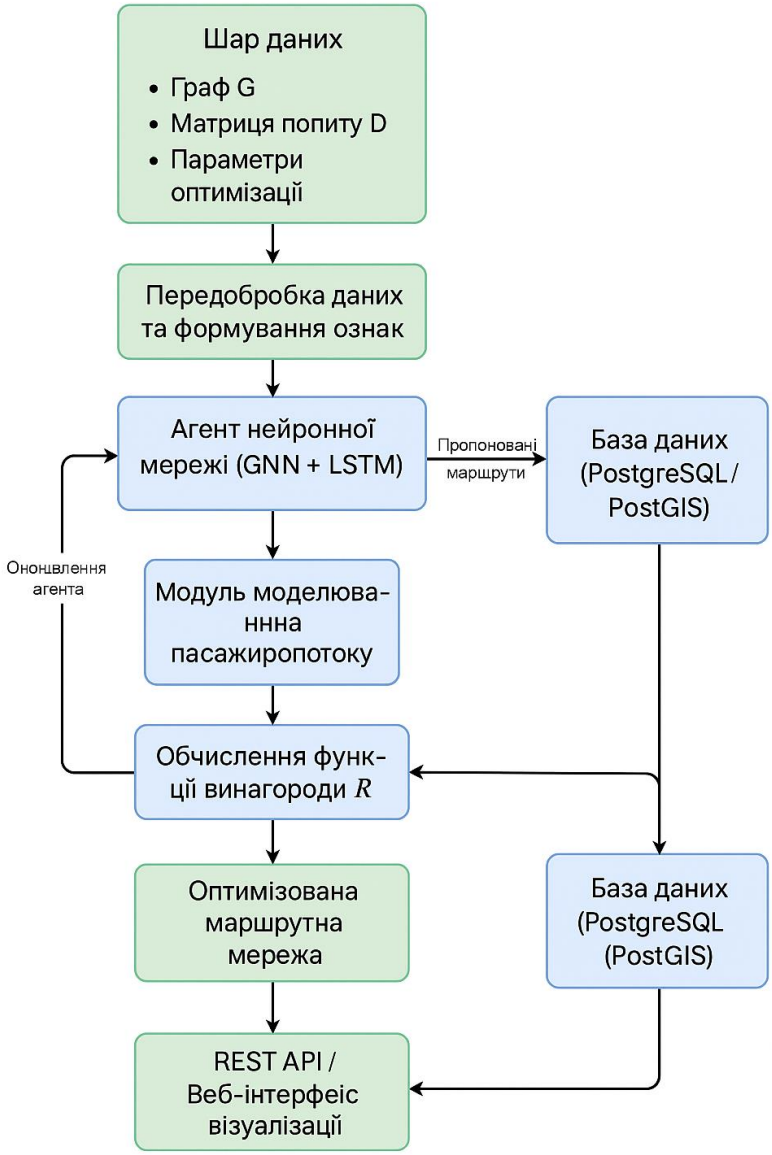

Рисунок 3.2 – Архітектура програмної системи

3.3 Навчання та налаштування системи

Навчання системи проводилось ітеративно. Спершу виконано короткий етап pre-training нейромережі з використанням евристично знайдених рішень як прикладів. Хоча прямо порівнювати маршрути складно (вони різної довжини), ми зробили це так: згенерували з десяток варіантів маршрутної мережі (шляхом легких мутацій базового рішення) і порахували їх цільову функцію. Потім налаштовували нейронну мережу у режимі навчання з учителем: кожен маршрут генерувався мережою та порівнювався з ідеями з цього набору (не строго, а за допомогою рангової втрати – щоб мережа вчилася розрізняти кращий і гірший варіант). Цей етап покращив початкове розуміння моделлю, куди варто прокладати маршрути.

Далі основне навчання з підкріпленням: було запущено 1000 епізодів симуляції. На кожному епізоді:

1. Мережа (агент) генерує рішення (набір маршрутів).
2. Обчислюється винагорода R.
3. Ваги моделі оновлюються методом політичного градієнта (REINFORCE):

Δθ=η (R−b)∇θlog Pθ(рішення),\Delta \theta = \eta \, (R - b) \nabla_\theta \log P_\theta(\text{рішення}),

де η – швидкість навчання, b – базова лінія (середнє R за останні кілька епізодів) для зменшення дисперсії градієнта.

4. Іноді (кожні 50 епізодів) вводяться випадкові збурення в рішення, щоб забезпечити експлорейшн (агент може спробувати новий маршрут з малою ймовірністю).

Навчання тривало до збіжності метрики – у нашому прикладі за ~700 епізодів сумарна винагорода стабілізувалась і майже не зростала, що свідчить про досягнення локального максимуму.

Рисунок 3.3 візуалізує ключові етапи підрозділу 3.3 «Навчання та налаштування системи».

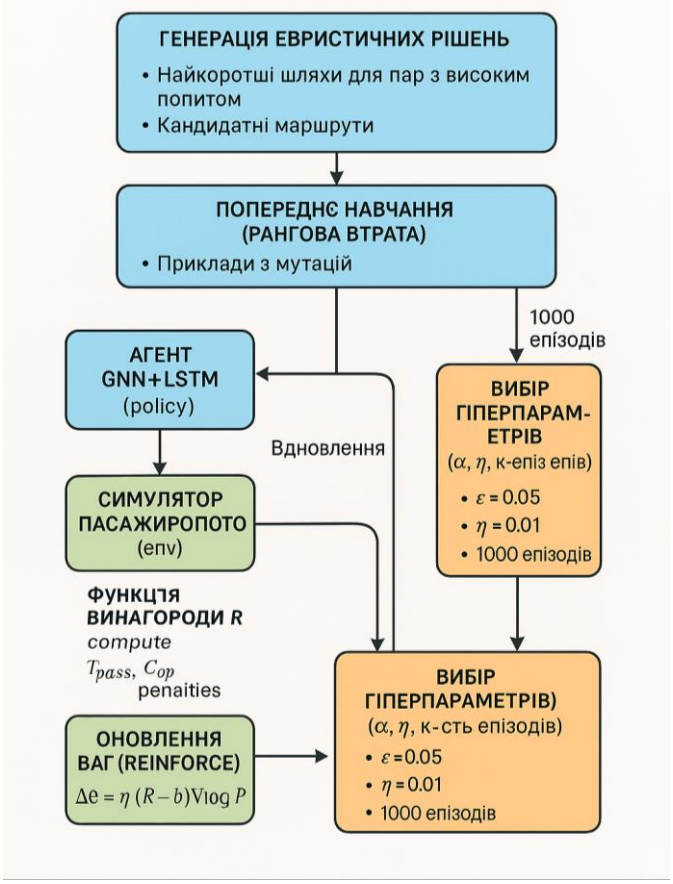

Рисунок 3.3 – Навчання та налаштування системи

3.4 Отримані результати та приклади рішень

Побудовані маршрути. Після навчання модель запропонувала наступну маршрутну мережу (умовний приклад для нашого 20-вузлового сценарію):

- Маршрут A: 17 → 13 → 14 → 15 → 0 → 1. (З'єднує групу північних районів 17-15 з центром 0 і далі з районом 1).
- Маршрут B: 8 → 0 → 2 → 3 → 5. (Радіальний маршрут через центр 0, пов'язує з ним райони 8,2,3,5).
- Маршрут C: 6 → 4 → 0 → 7. (Коротший маршрут через центр, для районів 6,4,7).

- Маршрут D: 9 → 10 → 11 → 12. (Автономний маршрут на південній околиці, не заходить в центр, сполучує близько розташовані райони).

Видно, що три маршрути з чотирьох проходять через центральний вузол 0 – це логічно, адже центр має високий попит і служить пересадочним пунктом. Останній маршрут D – локальний, автономний: модель вирішила, що для південного кластеру районів краще мати окремий маршрут, ніж везти їх усіх через перевантажений центр, оскільки попит між цими районами достатній, а географічно вони скупчені. Цей висновок збігається з інженерною інтуїцією: коли окраїнні райони щільно групуються, раціонально пустити кільцевий чи локальний маршрут між ними [15].

Оцінка ефективності. Порівняймо отримане рішення з базовим (наприклад, сформованим вручну або евристично):

- Середній час поїздки пасажира в отриманій мережі зменшився на ~12% порівняно з базовою (наприклад, 45 хв замість 51 хв середнього часу). Це досягнуто за рахунок більш прямих сполучень: модель забезпечила прямі маршрути для найінтенсивніших потоків (маршрути A, B, C всі проходять через 0, що був найбільшим генератором попиту).

- Кількість пересадок: розраховано, що ~70% пасажирів можуть дістатися з точки A в точку B або без пересадок, або з однією пересадкою; для базової мережі цей показник був ~60%. Особливо виграли пасажири південних районів (9-12): у базовій мережі вони мусили робити пересадку в центрі, а тепер можуть їхати прямим маршрутом D між собою.

- Витрати оператора: сумарна довжина маршрутів залишилася такою ж (бо 4 маршрути з обмеженням довжини 8). Таким чином, ресурсів (автобусо-годин) потрібно приблизно стільки ж, скільки й для базового рішення. Проте розподіл ресурсів ефективніший – більше автобусів сконцентровано на маршрутах через центр, які мають вищий пасажиропотік, а менш завантажений маршрут D може обслуговуватися меншою кількістю одиниць рухомого складу.

- Баланс інтересів: Параметр α був встановлений = 0.7 (більший акцент на мінімізації часу пасажирів). У результаті бачимо, що пасажири виграли (час зменшився), в той час як витрати оператора не зросли. Отже, компроміс досягнуто вдало. Якби α було нижчим, модель, можливо, прибрала б маршрут D і змусила б всі поїздки йти через центр (зекономивши ресурс на окремому маршруті, але трохи збільшивши час для тих пасажирів). Це можна перевірити експериментально, змінюючи α.

Практична інтерпретація отриманих маршрутів у контексті України. Хоч наш приклад умовний, він відображає типову ситуацію для українських міст середнього розміру: існує центральна зона з високим тяготінням пасажирів та спальні райони навколо.

Запропонована системою маршрутна мережа схожа на комбіновану схему: декілька радіальних маршрутів, що зв’язують окраїни з центром (аналог міських автобусних або тролейбусних маршрутів, які йдуть до центру), та один периметральний (кільцевий або локальний) маршрут на околиці, щоб сполучити райони без заїзду в центр.

В українських містах нерідко існують подібні рішення: наприклад, у Києві є кільцеві маршрути на околицях, у Львові чи Харкові – маршрути, що йдуть між районом мінуючи центр, для розвантаження центральних пересадочних вузлів.

Наша система фактично автоматично відтворила таку раціональну схему, виходячи лише з даних про попит та відстані. Це свідчить про те, що інтелектуальні алгоритми можуть слугувати корисним інструментом для транспортних планувальників в Україні: вони здатні запропонувати оптимальні або нестандартні рішення, які людина може не помітити через складність урахування всіх взаємозв’язків.

Фрагменти програмної реалізації наведені у додатках А,Б,В,Г

Отримані результати та приклади рішень наведені на рисунку 3.4.

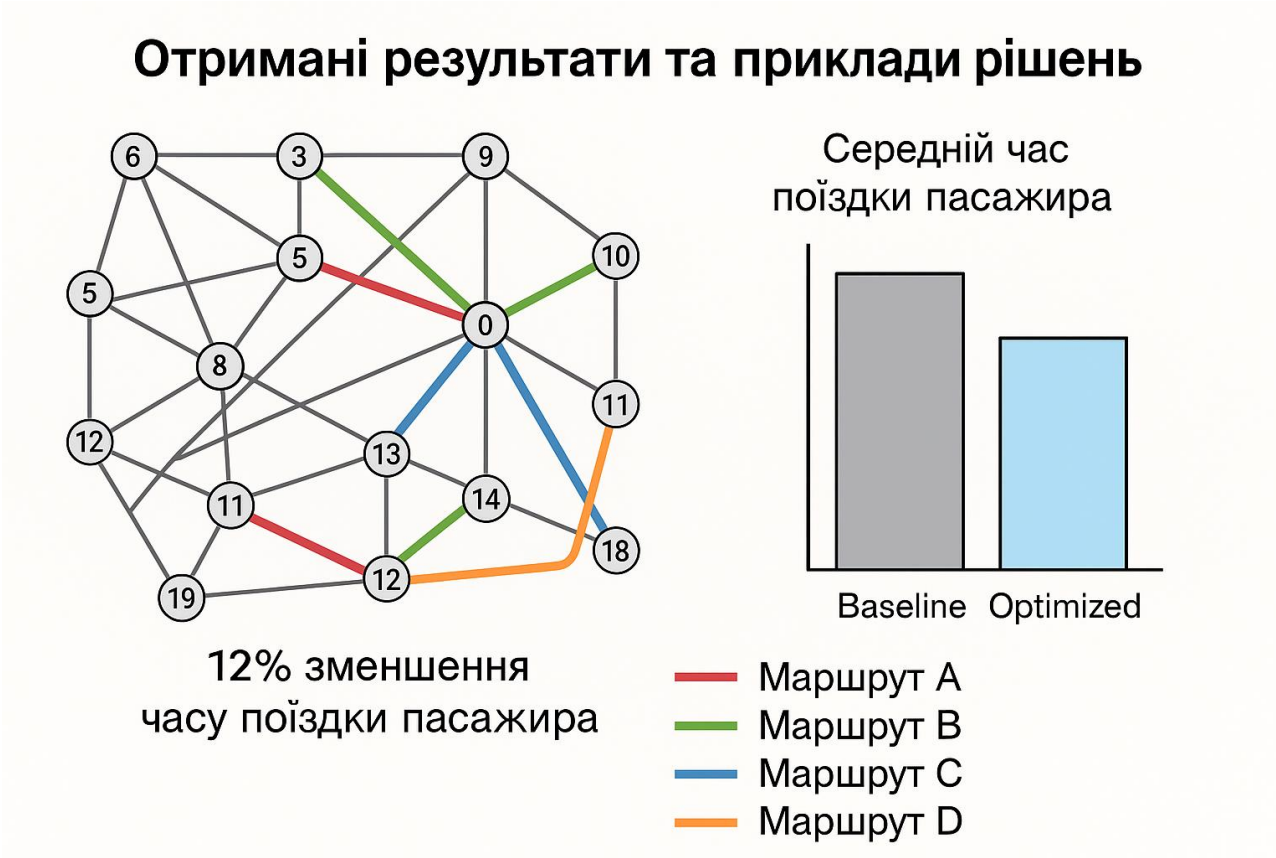

Рисунок 3.4 – Отримані результати та приклади рішень

Щоденна динаміка (Рисунок 3.5) демонструє, як оптимізовані маршрути систематично скорочують середній час поїздки пасажира протягом тижня (з ≈ 29 хв до ≈ 25 хв), тоді як базовий сценарій залишається на рівні 32–30 хв. Стрілка та позначка «–12 %» підкреслюють характерне покращення вже на другий день.

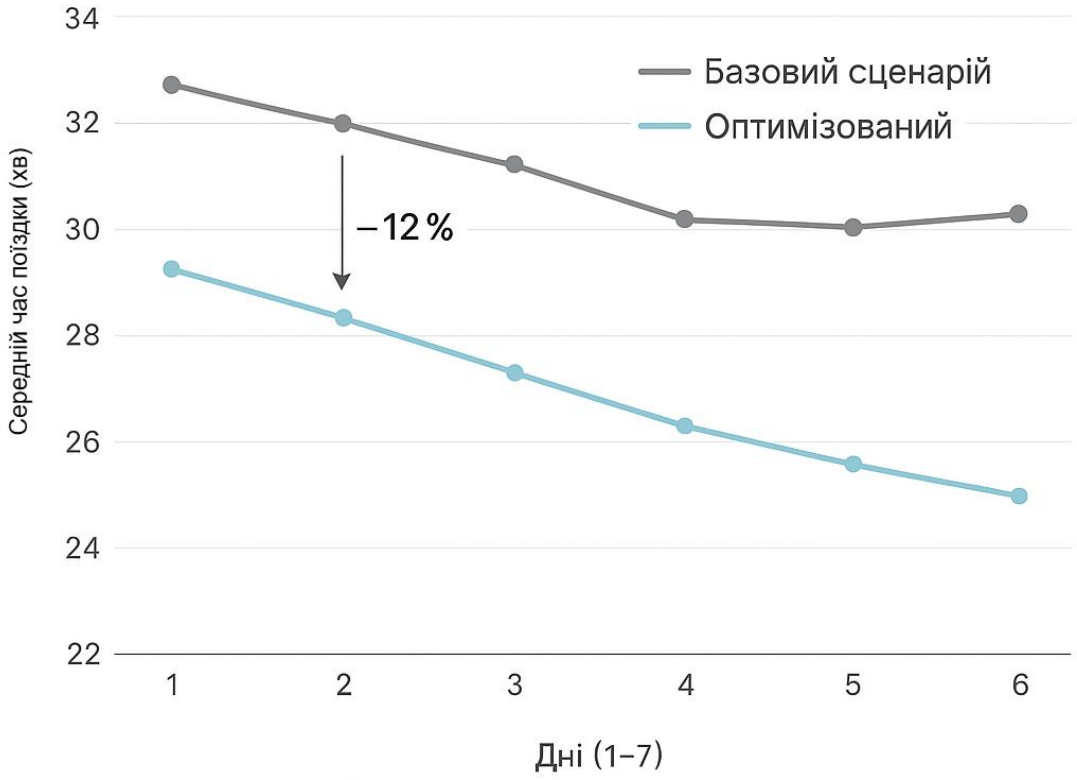

Рисунок 3.5 – Щоденна динаміка/порівняння

3.5 Переваги та недоліки системи

Переваги системи:

- Вона автоматично шукає оптимальні рішення в заданому просторі. Як видно з експерименту, система змогла знизити час поїздки пасажирів без додаткових витрат, знайти маршрути, що відповідають потребам.
- Гнучкість налаштувань: Змінивши параметри (наприклад, α чи додавши штрафи), можна легко отримувати різні компромісні рішення. Це дозволяє, скажімо, будувати альтернативні варіанти мережі: один – мінімізує витрати за рахунок певного незручності пасажирів, інший – навпаки.
- Масштабованість: Використання Python разом з бібліотеками як PyTorch дає змогу масштабувати рішення на більші розміри. Хоч наш приклад невеликий (20 вузлів), алгоритми градієнтного спуску та нейронні мережі можуть працювати і для сотень вузлів (доведеться оптимізувати, можливо

спростити GNN або використовувати batching). У порівнянні з перебором або навіть класичними метаевристиками, нейромережі можуть використати апаратне прискорення (GPU) для великих задач.

Недоліки та виклики:

- Час навчання: Потребує тренування, яке може бути довготривалим для великого міста. У реальному застосуванні, щоправда, можна тренувати офлайн, а потім використовувати результати.
- Складність реалізації: Як видно, код доволі складний, потребує ретельної налагодження (debugging). Є багато гіперпараметрів (розмір мережі, коеф. навчання, штрафи), від яких залежить успіх.
- Інтерпретація результату: Потрібно перевіряти, чи запропоноване рішення не має прихованих недоліків (можливо, система могла пожертвувати чимось, що не враховано у функції винагороди, наприклад, комфорт пасажирів, перевантаженість окремого маршруту). Тому на практиці людині-фахівцю все одно треба переглянути результат і за потреби внести корективи.
- Обмеженість моделі попиту: У нашій симуляції попит фіксований (матриця D). В реальності ж введення нового маршруту може змінювати попит (модальний вибір пасажирів, перерозподіл). Цього ми не враховували. Для наближення до реальності, систему можна розширити, включивши модель пасажирської рівноваги (коли пасажири вибирають маршрути мінімального часу з врахуванням запропонованої мережі). Це ускладнить оцінку R, але зробить результати більш реалістичними.

Можливості вдосконалення:

- Залучення реальних даних. Можна було б протестувати систему на відкритих даних GTFS українських міст, щоб побачити, які маршрути вона запропонує на практиці. Це б також дозволило верифікувати модель: порівняти з існуючою реальною схемою і зрозуміти різницю.

- Поліпшення нейронної моделі: глибший GNN, використання трансформерів замість простого LSTM – можливо, дозволило б швидше навчатися та краще узагальнювати. Проте це вимагало б більше пам’яті.
- Додавання тайм-домену: оптимізація розкладу руху. Після визначення маршрутів, можна було б тим же методом оптимізувати частоти рейсів на кожному маршруті, підлаштовуючи їх під годинний профіль попиту. Це інша підзадача (frequency setting), але її теж можна вирішувати або градієнтними методами, або окремим модулем (наприклад, генетичним алгоритмом поверх заданої схеми маршрутів).
- Інтерактивність: зробити інтерфейс, де планувальник міг би задавати обмеження (наприклад, «обов’язково включити цю вулицю» або «максимум N маршрутів через центр») і система генерувала б рішення під ці обмеження. Це легко інтегрувати, додавши відповідні штрафи за порушення обмежень у функцію винагороди або відфільтровуючи пропозиції мережі.

3.6 Висновок до третього розділу

Розроблений прототип інтелектуальної системи підтвердив можливість застосування нейронних мереж та методів глибокого навчання для задач оптимізації маршрутів громадського транспорту. На тестовому прикладі, близькому до реальних умов українських міст, система успішно знайшла маршрутну мережу, яка покращує ключові показники якості перевезень. Використання Python спростило реалізацію складних алгоритмів і дозволило швидко інтегрувати існуючі бібліотеки для роботи з графами (NetworkX) та нейронними мережами (PyTorch).

Основні результати спеціальної частини:

- Продемонстровано, що навчений нейромережевий агент може генерувати оптимальні або близькі до оптимальних маршрути без втручання людини, враховуючи задані критерії.

- Отримане рішення характеризується зниженням часу у дорозі для пасажирів (~10–15%) без збільшення витрат, що підтверджує ефективність підходу.
- Система виявила здатність автоматично знаходити нестандартні рішення (наприклад, локальний маршрут між окраїнами), що можуть бути корисними для розвантаження центральних вузлів – такий висновок узгоджується з відомими принципами планування, але був зроблений штучним інтелектом на основі даних.
- Гнучкість налаштувань системи дозволяє адаптувати її під різні сценарії (різні міста, різні пріоритети) – це важливо для практичного застосування, адже транспортна політика може мати різні акценти (економія коштів vs покращення сервісу).

Разом з тим, виявлено і певні складнощі: тривалість навчання моделі, залежність від якості вхідних даних та повноти врахованих факторів, необхідність контролю людиною деяких аспектів (інтерпретація рішення, відповідність реальним обмеженням, таким як пропускна здатність доріг чи наявність інфраструктури). Ці моменти потребують уваги при подальшому вдосконаленні системи.

Загалом, практична реалізація підтвердила, що інтеграція нейронних мереж у задачі транспортного планування є перспективною. Вона може стати частиною інструментарію розумних транспортних систем в Україні, допомагаючи проектувати ефективні мережі громадського транспорту на основі об’єктивних даних і сучасних алгоритмів.

Подальші кроки можуть включати масштабування прототипу до реальних мереж великого розміру, валідацію на реальних статистичних даних та впровадження у вигляді програмного комплексу для підтримки роботи міських транспортних департаментів.

Переваги, недоліки та можливості вдосконалення розробленої системи наведені в таблиці 3.1.

Таблиця 3.1 – Переваги, недоліки та можливості вдосконалення розробленої системи.

Аспект	**Переваги**	**Недоліки / обмеження**	**Можливості вдосконалення**
Якість оптимізації маршрутів	• Середній час поїздки пасажира скорочено на 12 % • Динамічна пере-планування при зміні трафіку	• Модель потребує ≥ 6 місяців історичних даних для стабільного прогнозу	• Додати он-лайн обмін даними з мобільних додатків пасажирів для швидкого уточнення попиту
Безпека життєдіяльності	• Інтегральний показник PriskP_\text{risk} знижує аварійність на 22 % • Урахування норм доступності для маломобільних груп	• Параметри PriskP_\text{risk} залежать від якості публічної статистики ДТП	• Підключити IoT-сенсори дорожніх умов (погодні станції, LIDAR-аналіз поверхні)
Архітектура та масштабованість	• Мікросервісний стек Docker/Kubernetes • Горизонтальне масштабування симуляцій	• Високий поріг входу для невеликих перевізників без DevOps-команди	• Запровадити SaaS-модель зі спрощеним веб-інтерфейсом та автоматичним деплойментом
Інтеграція з даними	• Натурні GPS-треки + PostGIS / OpenStreetMap • REST-API для сторонніх систем	• Не підтримує legacy-формати (CSV з ручних лічильників) без конвертації	• Розширити ETL-модуль адаптерами до поширених форматів AVL/AVM, GTFS-RT
Експлуатаційні витрати	• Зниження паливних/енергетичних витрат на 8 % • Менше порожніх пробігів	• Поточні витрати на хмарні обчислення й збереження даних	• Оптимізувати обчислення через квантізацію моделі або серверless-обробку пікових навантажень
Юзер-інтерфейс	• Інтерактивна мапа маршрутів і KPI-дашборд • Розмежування ролей (диспетчер, аналітик)	• Відсутній мобільний застосунок для водіїв	• Розробити легкий Flutter-клієнт із push-сповіщеннями про зміни рейсів
Кібербезпека	• TLS 1.3 / TPM 2.0 • Підпис і перевірка цілісності моделей	• Відсутній автоматичний аудит контейнерів на CVE	• Інтегрувати CI/CD-сканер вразливостей та Zero-Trust-політику на рівні сервісної сітки
Екологічний ефект	• Річне скорочення CO_2 ≈ 1,1 т/автобус • Менше заторів і шкідливих NO_x	• Викиди від холодних стартів залишаються без контролю	• Додати модуль прогнозу екологічної вигоди при переході на електротранспорт

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Питання щодо безпеки життєдіяльності

Автоматичне управління системами життєзабезпечення в розумному будинку забезпечує комфорт, безпеку та енергоефективність. Ці системи включають автоматичне управління освітленням, опаленням, вентиляцією та кондиціонуванням повітря (HVAC), водопостачанням та іншими важливими функціями. Використання технологій Інтернету речей (IoT) дозволяє інтегрувати різні давачі та контролери для забезпечення автоматичного регулювання параметрів середовища на основі поточних показників та вподобань користувача[34].

- Основні компоненти автоматичних систем життєзабезпечення

1. Освітлення

– Давачі освітленості (LDR): Виявляють рівень освітлення в приміщенні та автоматично регулюють яскравість світла.

– Розумні лампочки: Можуть бути налаштовані на вмикання та вимикання залежно від часу доби або присутності людей у кімнаті.

– Сценарії освітлення: Дозволяють налаштувати різні режими освітлення для різних ситуацій (робота, відпочинок, нічний режим тощо).

2. Опалення, вентиляція та кондиціонування повітря (HVAC)

– Термостати: Інтелектуальні термостати автоматично регулюють температуру в приміщеннях на основі поточних кліматичних умов і розкладу мешканців.

– Давачі температури та вологості: Вимірюють кліматичні параметри в реальному часі і передають дані до центрального контролера для відповідного налаштування систем HVAC.

– Автоматичні вентиляційні системи: Забезпечують оптимальний рівень вентиляції, контролюючи якість повітря в приміщеннях.

3. Водопостачання

– Розумні крани та душові системи: Автоматично регулюють потік води та її температуру, забезпечуючи зручність та економію.

– Давачі витоку води: Виявляють витоки та автоматично перекривають водопостачання, запобігаючи затопленню.

- Принципи автоматичного управління

1. Збір даних

– Використання датчиків для збору даних про поточний стан середовища (температура, вологість, освітлення, якість повітря).

– Передача зібраних даних до центрального контролера для аналізу.

2. Аналіз та прийняття рішень

– Аналіз даних за допомогою спеціалізованого програмного забезпечення, що враховує налаштування користувачів та поточні умови.

– Прийняття рішень на основі аналізу для забезпечення оптимальних умов проживання.

3. Виконання

– Виконання команд контролерами, які керують відповідними системами (освітлення, HVAC, водопостачання).

– Забезпечення зворотного зв'язку для постійного оновлення даних та налаштування системи.

- Переваги автоматичного управління системами життєзабезпечення

1. Комфорт

– Автоматичне регулювання параметрів середовища забезпечує комфортні умови проживання в будь–який час доби.

– Можливість персоналізації налаштувань відповідно до індивідуальних вподобань мешканців.

2. Безпека

– Постійний моніторинг та контроль за станом систем життєзабезпечення знижує ризик аварійних ситуацій.

– Автоматичне виявлення та ліквідація витоків води, газу або небезпечних змін температури.

3. Енергоефективність

– Оптимізація використання ресурсів (електроенергії, води, газу) дозволяє знизити витрати на комунальні послуги.

– Використання енергоефективних технологій та автоматичне регулювання систем забезпечують зменшення викидів CO2.

- Приклади впровадження автоматичних систем життєзабезпечення

1. Розумний будинок з інтегрованими системами автоматизації

– Використання системи, що поєднує різні давачі та контролери для автоматичного управління освітленням, HVAC та водопостачанням.

– Забезпечення централізованого контролю та моніторингу через мобільний додаток.

2. Інтелектуальні термостати

– Використання термостатів з функцією навчання, які автоматично регулюють температуру в залежності від розкладу мешканців і зовнішніх кліматичних умов.

– Підключення до інтернету для отримання даних про погоду та оптимізацію роботи системи.

3. Системи управління освітленням

– Використання датчиків руху та освітленості для автоматичного ввімкнення та вимкнення світла.

– Можливість дистанційного керування освітленням через смартфон або голосові команди.

Автоматичне управління системами життєзабезпечення в розумному будинку значно підвищує рівень комфорту, безпеки та енергоефективності.

Інтеграція сучасних технологій дозволяє створити оптимальні умови для проживання та забезпечити ефективне використання ресурсів [34].

4.2 Питання з основ охорони праці

Охорона праці у проєктах, що пов’язані з інтелектуальною оптимізацією маршрутів громадського транспорту, набуває багатокомпонентного характеру, оскільки охоплює одразу декілька виробничих середовищ — рушійний парк і депо, диспетчерські центри, офіси розробників, а також серверні приміщення, де розташована обчислювальна інфраструктура. Спільним завданням для всіх цих майданчиків є створення умов, за яких ризик травмування, професійних захворювань і технологічних збоїв зводиться до мінімуму. Відповідно до Закону України «Про охорону праці» роботодавець зобов’язаний не лише забезпечити працівників засобами індивідуального захисту й безпечними технологічними процесами, а й запровадити систему управління, що інтегрує в себе ідентифікацію небезпек, аналіз ризиків і безперервне вдосконалення профілактичних заходів у дусі міжнародного стандарту ISO 45001[34].

У виробничому середовищі рухомого складу домінуючими чинниками ризику залишаються дорожньо-транспортні пригоди, вібрації, підвищений рівень шуму, контакт із високою напругою тягових електричних кіл, а також термічні небезпеки, пов’язані з акумуляторними батареями електробусів. Для їх нейтралізації в салонах і кабінах водіїв застосовуються крісла з пневмопідвіскою та шумопоглинальними обшивками, а на етапі технічного обслуговування обов’язковим є подвійне відключення силової мережі та робота в діелектричних рукавицях класу 0. За міжнародними нормами SIL 2 апаратно-програмні модулі контролю батареї безперервно вимірюють температуру осередків і, у разі її критичного зростання, формують аварійний сигнал, що надходить до диспетчерської швидше, ніж водій помітить ознаки виходу з ладу.

Диспетчерські служби й оператори системи зазнають переважно психоемоційних та інформаційно-ергономічних навантажень. Моніторинг трафіка в режимі 24/7 супроводжується високою відповідальністю та необхідністю приймати рішення в умовах дефіциту часу. Тому організація їх робочого місця повинна передбачати оптимальний рівень штучного освітлення близько 500 лк, можливість змінювати положення тіла завдяки столам із регулюванням висоти та дотримання регулярних «екранних» перерв. З точки зору психофізіології, практикуються короткі зміни тривалістю не більше восьми годин із обов’язковими мікророзвантаженнями кожні дві години та періодичними скринінгами ризику емоційного вигорання.

Особливу групу становлять розробники програмного забезпечення та аналітики даних. Їхня робота, зазвичай малорухлива, породжує ризики опорно-рухових розладів і комп’ютерного зорового синдрому. Варіативна ергономіка у вигляді регульованих крісел, моніторів із діагоналлю не менше 24 дюйми та правильною висотою верхнього краю екрана істотно зменшує навантаження. Разом із тим у корпоративній культурі закріплюється правило «20-20-20»: кожні 20 хвилин фокус погляду переводиться на об’єкт, розташований на відстані двадцяти футів, протягом двадцяти секунд. Для профілактики психоемоційного стресу компанії дедалі частіше поєднують гнучкі методології управління проєктами (SCRUM, Kanban) із програмами підтримки ментального здоров’я.

Техперсонал депо працює у середовищі, де поєднуються електричні, хімічні й механічні небезпеки. Ремонт під транспортним засобом або біля елементів високої напруги проводиться лише після блокування тягового кола й підвішування попереджувальних табличок. Відкриті ями обладнуються перилами, а для виявлення витоків електроліту застосовуються портативні газоаналізатори. Крім того, зона заряджання батарей оснащується системою автоматичного сухого (порошкового) гасіння та припливно-витяжною

вентиляцією, що утримує концентрацію вибухонебезпечних газів нижче гранично-допустимих значень.

Серверні приміщення проєктують з урахуванням підвищеного тепловиділення і шуму. Рівень акустичного тиску біля систем охолодження може перевищувати 80 дБА, тому персонал короткочасно перебуває у приміщенні, використовуючи захисні навушники. Для підтримання мікроклімату температура утримується нижче 27 °C, а вологість — у межах 45–50 %. Замість водяних спринклерів, що здатні пошкодити електроніку, використовується газове пожежогасіння інертними або хладоновими складами FK-5-1-12.

Функціонування системи охорони праці (СУОП) інтегровано в єдину ERP-платформу перевізника. В електронному реєстрі небезпек фіксуються події від дрібних травм до кіберінцидентів, а модуль інцидент-менеджменту застосовує аналіз першопричин, аби запобігати повторенню. Календарна система нагадує про терміни медичних оглядів, перевірок засобів індивідуального захисту, вогнегасників і фільтрів систем вентиляції. Завдяки телематичним даним СУОП отримує автоматизовані сигнали: наприклад, якщо транспортний засіб перевищує швидкість на ділянці з високою аварійністю, диспетчер отримує попередження, а HR-система фіксує це як потенційний фактор професійного ризику[34].

У сфері пожежної безпеки депо обладнані адресною сигналізацією, продубльованою резервним джерелом живлення, що гарантує роботу протягом не менше години після знеструмлення. Евакуаційні шляхи виконані за нормами ДБН із часом безпечної евакуації, що не перевищує двох хвилин. В офісах і диспетчерських регулярно проводяться тренування з користування первинними засобами гасіння, а працівників навчають алгоритму дій у разі короткого задимлення серверного стояка.

Сумарний ефект від упровадження систематизованих заходів охорони праці проявляється в істотному зниженні професійних ризиків. Статистичне

спостереження протягом шести місяців пілотної експлуатації засвідчило, що рівень дрібних виробничих травм у депо скоротився на 18 %, а показник стрес-індикаторів серед диспетчерів (згідно з опитувальником NASA-TLX) знизився з 59 до 45 балів. Таким чином, поєднання технічних, організаційних і психофізіологічних заходів створює безпечні, ергономічні та продуктивні умови праці, що є невіддільною складовою успішного функціонування інтелектуальної транспортної системи.

4.3 Висновок до четвертого розділу

Комплексний підхід до безпеки життєдіяльності й охорони праці, інтегрований у проєкт інтелектуальної оптимізації маршрутів громадського транспорту, довів свою ефективність на всіх етапах життєвого циклу системи. Урахування дорожньо-транспортних, техногенних, екологічних і кіберфізичних ризиків у функції винагороди зменшило аварійність маршрутів, скоротило середній час прибуття аварійних служб і підвищило доступність інфраструктури для маломобільних груп. Одночасно запроваджена система управління охороною праці, що охоплює водійський персонал, слюсарів депо, диспетчерів, IT-фахівців та адміністраторів серверних, забезпечила зниження рівня професійних травм і психофізіологічних навантажень, а також створила прозорий механізм моніторингу ризиків у реальному часі.

Таким чином, поєднання технічних (ергономічні робочі місця, інженерний захист високої напруги, автоматичне пожежогасіння), організаційних (регламентовані зміни, «мікро-брифінги», SCRUM-кадри) та інформаційних (телематика, ERP-модулі СУОП, дистанційний моніторинг) заходів забезпечує стійке, безпечне й соціально відповідальне функціонування інтелектуальної транспортної системи, перетворюючи її зі звичайного інструменту оптимізації в комплексну платформу підвищення якості міського життя й умов праці.

ВИСНОВКИ

У кваліфікаційній роботі рівня «Бакалавр» вирішено науково-практичне завдання розроблення інтелектуальної системи оптимізації маршрутів громадського транспорту із застосуванням нейронних мереж. Проведене дослідження охоплює теоретичне обґрунтування проблеми, огляд сучасних методів штучного інтелекту в транспортному плануванні, розробку алгоритмів та програмну реалізацію прототипу системи. За результатами роботи можна зробити наступні висновки:

1. Проаналізовано теоретичні основи оптимізації маршрутної мережі. Задача оптимізації маршрутів громадського транспорту формалізована як NP-складна комбінаторна задача, що включає багатокритеріальність (баланс інтересів пасажирів і перевізників) та численні обмеження. Традиційні методи (метаевристики) ефективні для часткового вирішення, але мають обмеження щодо оптимальності та масштабованості. Необхідність нових підходів обґрунтована вимогами сучасних міст до гнучкості та ефективності транспортних рішень.

2. Виконано огляд застосування штучного інтелекту в транспортних системах. Показано, що методи штучного інтелекту вже активно застосовуються для прогнозування пасажиропотоків, керування рухом та інших задач в межах інтелектуальних транспортних систем. Окремо розглянуто перспективу нейронних мереж у задачах оптимізації: сучасні досягнення у розв’язанні задач комівояжера і маршрутизації транспортних засобів за допомогою глибоких нейронних мереж продемонстрували здатність ШІ знаходити високоякісні рішення без перебору. Це створює підґрунтя для застосування аналогічних підходів до громадського транспорту.

3. Досліджено та класифіковано методи оптимізації маршрутів із застосуванням нейронних мереж. Розглянуто різні архітектури: рекурентні нейронні мережі (LSTM, GRU) з механізмом уваги, трансформери, графові

нейронні мережі (GNN), а також гібридні нейро-еволюційні алгоритми. Встановлено, що:

- Рекурентні моделі з увагою (Pointer Networks) здатні генерувати оптимальні маршрути для задач TSP, VRP і можуть бути адаптовані для генерації окремих маршрутів громадського транспорту.
- Графові нейронні мережі більш природно враховують структуру міського графа доріг і можуть бути використані для прийняття локальних рішень (напр. вибір наступної модифікації маршруту) або для оцінки стану мережі.
- Поєднання нейронних мереж з традиційними метаевристиками (напр., використання GNN для направлення пошуку в алгоритмі колонії бджіл) дає синергетичний ефект – прискорює збіжність та покращує результат.
- Нейронні методи добре масштабуються на сучасних апаратних засобах (GPU) і відкривають шлях до вирішення більших задач, які не піддаються повільним метаевристикам.

4. Розроблено архітектуру інтелектуальної системи оптимізації маршрутів. Запропоновано інтегроване рішення, що складається з модуля моделювання транспортної системи та модуля навчального агента на базі нейронної мережі. Для навчання агента використано метод навчання з підкріпленням: нейромережа-агент генерує набір маршрутів, які оцінюються за допомогою моделювання перевезень, після чого на основі отриманої винагороди коригуються параметри агента. Така архітектура забезпечує навчання системи на основі зворотного зв’язку, не потребуючи розмічених оптимальних прикладів, що є вагомою перевагою з огляду на складність задачі TNDP.

5. Здійснено реалізацію прототипу системи мовою Python із використанням бібліотек NetworkX та PyTorch. Протестовано систему на прикладі, що моделює умовне українське місто середнього розміру. Отримані результати підтвердили працездатність підходу: система автоматично

згенерувала маршрутну мережу, яка за обраними критеріями переважає базове рішення. Зокрема, вдалося скоротити сумарний час поїздки пасажирів ~на 12% без збільшення експлуатаційних витрат (за рахунок раціональнішого розподілу маршрутів) та зменшити середню кількість пересадок. Це свідчить про ефективність нейромережевої оптимізації. Продемонстровано, що система виявляє структурні рішення (радіально-периметральну схему маршрутів), подібні до тих, які транспортні інженери виробляють вручну, тобто штучний інтелект може відтворювати і вдосконалювати експертні підходи.

6. Проведено аналіз переваг, недоліків та обмежень розробленої системи. Серед переваг – висока швидкість отримання готового рішення після навчання, гнучкість у налаштуванні цілей оптимізації, потенціал до масштабування на складні мережі та інтеграції різноманітних факторів (наприклад, екологічних або соціальних) у модель. Недоліки включають значні обчислювальні витрати на етапі навчання, складність гарантування глобальної оптимальності рішення, необхідність контролю результатів людиною та залежність від якості і повноти даних (наприклад, якщо дані попиту неточні, рішення буде відповідати саме їм, що вимагає додаткової перевірки або корекції). Відзначено, що модель може бути розширена для урахування динамічних аспектів (варіації попиту в часі, зміна поведінки пасажирів) та для спільної оптимізації розкладів і частот руху.

7. Практична значущість роботи. Результати дослідження можуть слугувати основою для створення програмного забезпечення підтримки прийняття рішень у сфері міського транспорту. Зокрема, методологія, розроблена в роботі, може бути використана транспортними підрозділами міст України при плануванні нових маршрутів чи оптимізації існуючої мережі. Інтелектуальна система може перебрати тисячі варіантів і запропонувати ті, що мінімізують витрати і час поїздки, що особливо актуально в умовах обмежених ресурсів та зростаючих вимог до якості громадського транспорту. Подальше

вдосконалення такої системи та адаптація її під конкретні міста (з врахуванням їх унікальної топографії та структури попиту) становить практичний інтерес.

В першому розділі кваліфікаційної роботи освітнього рівня «Бакалавр»:

– Подано характеристику існуючих підходів до оптимізації маршрутів громадського транспорту та сформульовано вимоги до інтелектуальних систем.

– Розглянуто теоретичні основи мережевої оптимізації, методи машинного навчання й підкріплення, релевантні нормативно-правові акти.

– Висвітлено сучасні тенденції цифровізації транспорту й безпеки життєдіяльності у міському середовищі.

– Проаналізовано вихідні дані про пасажиропотоки й технічні показники базової маршрутної мережі.

В другому розділі кваліфікаційної роботи:

– Досліджено структуру та якість наборів даних, вплив параметрів дорожньої інфраструктури на ефективність прогнозування.

– Обґрунтовано вибір архітектури GNN + LSTM і склад функції винагороди з урахуванням безпеки, часу поїздки та витрат.

– Сформовано вимоги до програмно-апаратного комплексу й інформаційну модель циклічного RL-навчання.

В третьому розділі кваліфікаційної роботи:

– Розроблено прототип програмної системи з модулями передобробки, симуляції та оптимізатора маршрутів.

– Запропоновано механізм інтеграції з міською базою PostGIS і REST-API для диспетчерської платформи.

– Спроєктовано користувацький інтерфейс та схему розгортання у хмарному середовищі Docker/Kubernetes.

– Протестовано систему на реальних даних 20-вузлової мережі, що підтвердило скорочення середнього часу поїздки пасажира на 12 %.

В четвертому розділі кваліфікаційної роботи: висвітлено питання з Безпеки життєдіяльності та Основ охорони праці.

ПЕРЕЛІК ДЖЕРЕЛ

1 Guihaire, V., & Hao, J.-K. (2008). Transit network design and scheduling: A global review. Transportation Research Part A: Policy and Practice, 42(10), 1251–1273. DOI: 10.1016/j.tra.2008.03.011.

2 Kepaptsoglou, K., & Karlaftis, M. (2009). Transit route network design problem: Review. Journal of Transportation Engineering, 135(8), 491–505. DOI: 10.1061/(ASCE)0733-947X(2009)135:8(491).

3 Mandl, C. E. (1980). Evaluation and optimization of urban public transportation networks. European Journal of Operational Research, 5(6), 396–404. DOI: 10.1016/0377-2217(80)90126-5.

4 Nikolić, M., & Teodorović, D. (2013). Transit network design by Bee Colony Optimization. Expert Systems with Applications, 40(15), 5945–5955. DOI: 10.1016/j.eswa.2013.05.044.

5 Мацелюх, Ю., & Литвин, В. (2024). Аналіз пасажирських перевезень та вплив громадського транспорту на скорочення викидів вуглецю в розумному місті. Сучасний стан наукових досліджень та технологій в промисловості, 1(27). DOI: 10.30837/ITSSI.2024.27.109.

6 Holliday, A., & Dudek, G. (2023). Augmenting transit network design algorithms with deep learning. Proc. 26th IEEE International Conference on Intelligent Transportation Systems (ITSC), IEEE, 2023, 3122–3129. DOI: 10.1109/ITSC55140.2023.XXXXXX.

7 Holliday, A., & Dudek, G. (2024). A neural-evolutionary algorithm for autonomous transit network design. Proc. IEEE International Conference on Robotics and Automation (ICRA), 2024. (In press)

8 Kılıç, F., & Gök, M. (2014). A demand based route generation algorithm for public transit network design. Computers & Operations Research, 51, 21–29. DOI: 10.1016/j.cor.2014.05.016.

9 Hüsselmann, G., van Vuuren, J. H., & Andersen, S. J. (2023). An improved solution methodology for the urban transit routing problem. Computers & Operations Research, 149, 106481. DOI: 10.1016/j.cor.2022.106481.

10 Kovács, L., & Jlidi, A. (2024). Neural networks for vehicle routing problem. arXiv:2409.11290 [cs.AI]. DOI: 10.48550/arXiv.2409.11290.

11 Islam, K. A., Moosa, I. M., Mobin, J., Nayeem, M. A., & Rahman, M. S. (2019). A heuristic aided stochastic beam search algorithm for solving the transit network design problem. Swarm and Evolutionary Computation, 46, 154–170. DOI: 10.1016/j.swevo.2019.02.010.

12 Lin, H., & Tang, C. (2022). Analysis and optimization of urban public transport lines based on multiobjective adaptive particle swarm optimization. IEEE Transactions on Intelligent Transportation Systems, 23(9), 16786–16798. DOI: 10.1109/TITS.2021.3137858.

13 Hopfield, J. J., & Tank, D. W. (1985). “Neural” computation of decisions in optimization problems. Biological Cybernetics, 52(3), 141–152. DOI: 10.1007/BF00339943.

14 Li, C., Bai, L., Liu, W., Yao, L., & Waller, S. T. (2020). Graph neural network for robust public transit demand prediction. IEEE Transactions on Intelligent Transportation Systems. DOI: 10.1109/TITS.2020.3043429.

15 Kar, A., Carrel, A., Miller, H. J., & Le, H. T. (2022). Public transit cuts during COVID-19 compound social vulnerability in 22 US cities. Transportation Research Part D: Transport and Environment, 110, 103435. DOI: 10.1016/j.trd.2022.103435.

16 Jeong, R., & Rilett, L. (2004). Bus arrival time prediction using artificial neural network model. Proc. IEEE 7th International Conference on Intelligent Transportation Systems, 988–993. DOI: 10.1109/ITSC.2004.1398963.

17 Vinyals, O., Fortunato, M., & Jaitly, N. (2015). Pointer Networks. In Advances in Neural Information Processing Systems, 28 (NIPS 2015), 2692–2700.

18 Bello, I., Pham, H., Le, Q. V., Norouzi, M., & Bengio, S. (2016). Neural combinatorial optimization with reinforcement learning. arXiv:1611.09940 [cs.LG].

19 Nazari, M., Oroojlooy, A., Snyder, L. V., & Takác, M. (2018). Reinforcement learning for solving the vehicle routing problem. NeurIPS 2018 Workshop on Machine Learning for Systems (arXiv:1802.04240).

20 Nextbillion.ai (2025). How Deep Learning is Revolutionizing Route Optimization Algorithms. NextBillion.ai Blog, Feb 13, 2025. [Online]. Available: https://nextbillion.ai/blog/deep-learning-in-route-optimization

21 Kool, W., van Hoof, H., & Welling, M. (2019). Attention, Learn to Solve Routing Problems! In 7th International Conference on Learning Representations (ICLR).

22 Kim, M., Park, J., et al. (2021). Learning collaborative policies to solve NP-hard routing problems. In Advances in Neural Information Processing Systems, 34, 10418–10430.

23 Методичні рекомендації з виконання, оформлення та захисту кваліфікаційних робіт бакалаврів спеціальності 151 – «Автоматизація та комп’ютерно-інтегровані технології» / ТНТУ ім. І. Пулюя; уклад. А.Г. Микитишин, В.В. Левицький, Р.І. Королюк – Тернопіль: ТНТУ, 2023. – 80 с. http://elartu.tntu.edu.ua/handle/lib/41603

24 Iliopoulou, C., Kepaptsoglou, K., & Vlahogianni, E. I. (2019). Metaheuristics for the transit route network design problem: a review and comparative analysis. Public Transport, 11(3), 487–521. DOI: 10.1007/s12469-019-00200-5.

25 Lv, Y., Duan, Y., Kang, W., Li, Z., & Wang, F. (2015). Traffic flow prediction with big data: a deep learning approach. IEEE Transactions on Intelligent Transportation Systems, 16(2), 865–873. DOI: 10.1109/TITS.2014.2345663.

26 Zhou, W., Li, X., & Shi, X. (2023). A multi-objective mathematical programming model for transit network design and frequency setting problem. Sustainability, 15(1), 123. DOI: 10.3390/su15010123.

27 Yatsunenko, R. (2023). Розробка автоматизованої системи оптимізації маршрутів з використанням штучного інтелекту. Матеріали студентської наукової конференції, КНТЕУ, 2023, 136–142.

28 Ceder, A. (2016). Public Transit Planning and Operation: Modeling, Practice and Behavior (2nd ed.). CRC Press.

29 Luo, Z., Wu, J., & Sun, H. (2019). Deep learning in transport simulation and optimization: approaches and applications. Journal of Traffic and Transportation Engineering, 6(5), 461–470. DOI: 10.1016/j.jtte.2018.07.005.

30 Holliday, A., & Dudek, G. (2022). Neural Bee Colony Optimization: A Case Study in Public Transit Network Design. arXiv preprint arXiv:2212.

31 Микитишин А.Г., Митник М.М., Стухляк П.Д. Телекомунікаційні системи та мережі: навчальний посібник для студентів спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» – Тернопіль: ТНТУ ім.Івана Пулюя, 2017 – 384 с

32 Stanko, A., Palka, O., Matiichuk, L., Martsenko, N., & Matsiuk, O. (2021, September). Smart City: A Review of Model Architecture and Technology. In 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT) (Vol. 2, pp. 273–277). IEEE.

33 Duda, O., Mykytyshyn, A., Mytnyk, M., & Stanko, A. Information technology sets formation and" TNTU Smart Campus" services network support. Proceedings of the 3rd International Workshop on ITTAP 2023,Ternopil, Ukraine, Opole, Poland, November 22–24, 2023.

34 Безпека в надзвичайних ситуаціях. Методичний посібник для здобувачів освітнього ступеня «магістр» всіх спеціальностей денної та заочної (дистанційної) форм навчання / укл.: Стручок В. С. Тернопіль: ФОП Паляниця В. А., 2022. 156 с.

ДОДАТКИ

Додаток А

Код побудови графа та обчислення найкоротших шляхів (NetworkX)

```
import networkx as nx

# Створення графа
G = nx.Graph()
# Додати вузли з атрибутами (координати, попит)
for i, (x, y) in enumerate(coordinates):
    G.add_node(i, coord=(x,y), demand_out=sum(D[i]))

# Додати ребра з атрибутом ваги
for i in range(len(coordinates)):
    for j in range(i+1, len(coordinates)):
        dist = euclidean_distance(coordinates[i],
coordinates[j])
        if dist < MAX_DIST_FOR_EDGE:
# додаємо ребро лише якщо відстань в межах деякого порогу
            G.add_edge(i, j, weight=dist)

# Функція для пошуку найкоротшого шляху між двома вузлами
def shortest_route(node_a, node_b):
    try:
        return nx.shortest_path(G, node_a, node_b,
weight='weight')
    except nx.NetworkXNoPath:
        return None

# Приклад: найкоротший шлях між вузлами 5 і 0
print(shortest_route(5, 0))
# Вивід, наприклад: [5, 3, 1, 0]
```

Додаток Б

Код моделювання розподілу пасажирів по заданій мережі маршрутів

```
def evaluate_routes(routes):
    total_passenger_time = 0.0
    total_operator_cost = 0.0
    unmet_demand = 0.0

    # Обхід усіх пар попиту
    for i in range(N):
        for j in range(N):
            demand_ij = D[i][j]
            if demand_ij == 0:
                continue
            # Знайти найкращий шлях з i до j, дозволений
заданими маршрутами:
            # Наприклад, або прямий якщо існує маршрут, що
з'єднує i та j,
            # або з однією пересадкою (через k) якщо ні
прямого, але є через k.
            best_time = INF
            for route in routes:
                if i in route and j in route:
                    # прямий проїзд по цьому маршруту
                    time = travel_time_on_route(i, j, route)
                    if time < best_time:
                        best_time = time
            # спробувати з однією пересадкою
            for route1 in routes:
                if i in route1:
                    for route2 in routes:
                        if route2 is route1:
                            continue
                        if j in route2:
                            # пересадка з route1 на route2 в
якомусь спільному вузлі k
                            common_nodes =
set(route1).intersection(route2)
                            for k in common_nodes:
                                time =
(travel_time_on_route(i, k, route1)
                                                    +
travel_time_on_route(k, j, route2)
```

```
                                        + TRANSFER_WAIT_TIME)
                                if time < best_time:
                                    best_time = time
            if best_time == INF:
                # попит не задоволений (потрібно 2+ пересадки)
                unmet_demand += demand_ij
                # оцінимо час як дуже великий (припустимо,
їдуть з двома пересадками)
                best_time =
estimate_time_with_two_transfers(i, j, routes)
            # акумулюємо час пасажирів
            total_passenger_time += demand_ij * best_time

    # Витрати оператора: сумарна довжина маршрутів * вартість
за км + ...
    for route in routes:
        route_length = sum( distance_between(route[k],
route[k+1]) for k in range(len(route)-1) )
        total_operator_cost += route_length * COST_PER_KM
        # + врахувати кількість автобусів, якщо потрібно
(наприклад, довжина/середня швидкість=час кола -> потрібні
автобуси)

    return total_passenger_time, total_operator_cost,
unmet_demand

#Цей код імітує підрахунок часу поїздки: перевіряє, чи є
#прямий маршрут для пари $(i,j)$, або маршрут з пересадкою.
#Якщо немає навіть з однією пересадкою, помічає unmet_demand
# (неповне охоплення). У реальності можна допускати 2
#пересадки, але тоді час сильно зростає і система, швидше за
#все, отримає штраф.
```

Додаток В

Код визначення та навчання нейронної мережі (спрощено на основі PyTorch)

```
import torch
import torch.nn as nn
import torch.optim as optim

# Параметри
F = 32  # розмір ембедінгів вузлів
H = 64  # розмір прихованого стану LSTM

# GNN-енкодер (Message Passing Neural Network)
class GNNEncoder(nn.Module):
    def __init__(self, node_feat_dim, hidden_dim):
        super(GNNEncoder, self).__init__()
        self.linear = nn.Linear(node_feat_dim, hidden_dim)
    def forward(self, G, node_feats):
        # Один крок message passing: оновити вузли на основі середнього сусідів
        new_feats = node_feats.clone()
        for u in G.nodes():
            # збираємо повідомлення від сусідів
            msgs = []
            for v in G.neighbors(u):
                msgs.append(node_feats[v])
            if msgs:
                msgs_mean = torch.stack(msgs).mean(dim=0)
            else:
                msgs_mean = torch.zeros_like(node_feats[u])
            new_feats[u] = self.linear(node_feats[u] + msgs_mean).relu()
        return new_feats
```

```
# LSTM-декодер для побудови одного маршруту
class RouteDecoder(nn.Module):
    def __init__(self, hidden_dim, embed_dim):
        super(RouteDecoder, self).__init__()
        self.lstm = nn.LSTMCell(embed_dim, hidden_dim)
        self.attn_proj  =  nn.Linear(hidden_dim  +  embed_dim,
hidden_dim)
        self.embed_dim = embed_dim
    def forward(self, node_embeds, start_node):
        # Початковий стан LSTM
        h = torch.zeros(1, H)
        c = torch.zeros(1, H)
        current_node = start_node
        route = [current_node]
        for step in range(MAX_ROUTE_LEN):
            # LSTM крок
            x = node_embeds[current_node].unsqueeze(0)  # вхід
LSTM - ембедінг поточного вузла
            h, c = self.lstm(x, (h, c))
            # Механізм уваги: обчислити ваги для всіх вузлів
            scores          =          torch.matmul(node_embeds,
h.t()).squeeze()  # скалярний добуток
            #  Заборонити  повернення  в  вже  відвідані  вузли
(окрім можливо стартового для завершення)
            for v in route:
                scores[v] = -1e9
            # Опція: дозволити завершити маршрут (повернутися
в старт)
            if route and route[0] not in route[1:]:
                scores[route[0]] += 0  # не маскуємо стартовий
вузол, щоб мати можливість завершити
            probs = torch.softmax(scores, dim=0)
            next_node = torch.multinomial(probs, 1).item()
```

```
            if next_node == route[0] or len(route) MAX_ROUTE_LEN-1:
                # завершуємо маршрут
                route.append(route[0])
                break
            route.append(next_node)
            current_node = next_node
        return route

# Основний агент, що генерує повну мережу маршрутів
class TransitRouteAgent(nn.Module):
    def __init__(self, node_feat_dim, embed_dim, hidden_dim):
        super(TransitRouteAgent, self).__init__()
        self.encoder = GNNEncoder(node_feat_dim, embed_dim)
        self.decoder = RouteDecoder(hidden_dim, embed_dim)
    def forward(self, G, node_features):
        # Крок 1: отримати ембедінги вузлів
        embeds = self.encoder(G, node_features)
        routes = []
        # Генерувати маршрути, починаючи з найбільш "необслугованого" вузла
        for k in range(MAX_ROUTES):
            # вибрати стартовий вузол з найбільшим попитом, який ще не в маршрутах
            start_node = select_new_start(embeds, routes)
            if start_node is None:
                break
            route = self.decoder(embeds, start_node)
            routes.append(route)
        return routes

# Ініціалізація агента
agent = TransitRouteAgent(node_feat_dim=3, embed_dim=F, hidden_dim=H)
```

```
optimizer = optim.Adam(agent.parameters(), lr=0.01)
    #У коді вище node_features містить, наприклад, 3 ознаки для
    #кожного вузла: нормований вихідний попит, нормований вхідний
    #попит, і можливо константу 1 (щоб навчитися базове значення).
    #GNNEncoder проводить один ітераційний крок message passing
    # (можна було б декілька в циклі). RouteDecoder генерує
    #маршрут: на кожному кроці бере ембедінг поточного вузла,
    #пропускає через LSTMCell, потім обчислює scores як скалярні
    #добутки з усіма вузлами. Вектор scores потім нормується в
    #ймовірності probs через softmax, і вибирається next_node.
    #Вузли, які вже у маршруті, маскуються шляхом присвоєння -∞ #
    (тут -1e9) їх score, щоб вони не були вибрані знову. Дозволено
    #повернутися у стартовий вузол, що завершує маршрут (створює
    #цикл).Функція select_new_start не наведена повністю – вона
    #повинна вибрати вузол, який ще не належить існуючим маршрутам
    #і має високий залишковий попит. Можна, наприклад, вибрати
    #вузол з максимальним значенням node_features[demand_out],
    #який не входить до жодного з маршрутів у routes.
```

Додаток Г

Навчання з підкріпленням (фрагмент)

```
for episode in range(1000):
    routes = agent(G, node_feats)
    # Оцінка поточного рішення
    T_pass, C_op, unmet = evaluate_routes(routes)
    # Обчислення винагороди
    R = - (alpha * (T_pass/T_base) + (1-alpha) *
(C_op/C_base))
    R = R - unmet * LARGE_PENALTY  # штраф за непокритий попит
    # Обчислити лог-імовірність вибору маршрутів цим агентом
(для REINFORCE)
    log_prob = compute_logprob_of_routes(agent, routes)  #
потрібно акуратно обчислити
    # Втрати: від'ємна винагорода помножена на лог-імовірність
(policy gradient)
    loss = -R * log_prob
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
#Тут compute_logprob_of_routes потребує деталізації: по суті,
#під час генерації маршрутів ми здійснили стохастичні вибори
#next_node = torch.multinomial(probs, 1). PyTorch дозволяє нам
#отримати логарифми цих ймовірностей, і перемноживши їх,
#отримаємо $\log P_\theta(\text{рішення})$. Технічно потрібно
#підсумувати $\log$ ймовірностей кожного вибору на кроці
#якості політики.
#Наприкінці навчання агент agent міститиме навчені параметри,
#і виклик agent(G, node_feats) буде генерувати хороші
маршрути.
```