Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Центр післядипломної освіти
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів розробки мультиагентної системи навчання світлофорного регулювання для оптимізації міських транспортних потоків
(тема)

Виконав:
здобувач другого року навчання,
групи СШІздм-24-1
Олександр КОЛІЙ
(власне ім'я, прізвище)
Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)
Освітня програма Системи штучного інтелекту
(повна назва освітньої програми)
Керівник проф. Марина КУДРЯВЦЕВА
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Лариса ЧАЛА
(підпис) (власне ім'я, прізвище)

2026 р.

Харківський національний університет радіоелектроніки

Факультет Центр післядипломної освіти

Кафедра Штучного інтелекту

Рівень вищої освіти другий (магістерський)

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

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

Освітня програма Системи штучного інтелекту
(повна назва)

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

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Колію Олександру Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів розробки мультиагентної системи навчання світлофорного регулювання для оптимізації міських транспортних потоків

затверджена наказом університету від 6 квітня 20 26 р. № 48Стз

2. Термін подання здобувачем роботи до екзаменаційної комісії 26 травня 20 26 р.

3. Вихідні дані до роботи Науково-технічні публікації, дані Інтернет-джерел та відомих наукових проєктів, Python documentation

4. Перелік питань, що потрібно опрацювати в роботі

1) Аналіз предметної галузі та постановка задачі

2) Теоретичне дослідження методів штучного інтелекту для розв'язання задачі світлофорного регулювання

3) Розробка методології та архітектури мультиагентної системи навчання світлофорного регулювання

4) Програмна реалізація, експериментальне налаштування та оцінка ефективності

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / термін виконання етапів роботи	Примітка
1.	Отримання завдання на кваліфікаційну роботу	06.04.2026	виконано
2.	Аналіз предметної галузі та постановка задачі	20.02.2026	виконано
3.	Теоретичне дослідження методів штучного	10.03.2026	виконано
	інтелекту для розв’язання задачі світлофорного		
	регулювання		
4.	Розробка методології та архітектури	01.04.2026	виконано
	мультиагентної системи навчання світлофорного		
	регулювання		
5.	Програмна реалізація, експериментальне	25.04.2026	виконано
	налаштування та оцінка ефективності		
6.	Оформлення пояснювальної записки	15.05.2026	виконано
7.	Попередній захист	25.05.2026	виконано
8.	Захист перед ЕК	26.05.2026	

Дата видачі завдання 6 квітня 20 26 р.

Здобувач _____
(підпис)

Керівник роботи _____ проф. Марина КУДРЯВЦЕВА
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 245 с., 21 рис., 25 табл., 7 дод., 18 джерел.

АДАПТИВНЕ КЕРУВАННЯ, МУЛЬТИАГЕНТНІ СИСТЕМИ, ПІДКРІПЛЮВАЛЬНЕ НАВЧАННЯ, СВІТЛОФОРНЕ РЕГУЛЮВАННЯ, ТРАНСПОРТНІ ПОТОКИ, ШТУЧНИЙ ІНТЕЛЕКТ

Об’єкт дослідження – процес світлофорного керування міськими транспортними потоками в умовах динамічного транспортного попиту.

Предмет дослідження – методи побудови, навчання та оцінювання мультиагентної системи керування світлофорними фазами в симуляційному середовищі.

Мета роботи – дослідження методів розробки мультиагентної системи навчання світлофорного регулювання для оптимізації міських транспортних потоків.

Методи дослідження – аналіз предметної області світлофорного регулювання, методи підкріплювального та мультиагентного навчання, моделювання транспортних потоків, алгоритми DQN і PPO, експериментальне порівняння моделей за показниками затримок, черг, часу очікування, втрат часу та кількості завершених поїздок.

Наукова новизна роботи – обґрунтовано підхід до подання світлофорного регулювання міської транспортної мережі як мультиагентної задачі навчання з урахуванням локальних станів агентів, допустимих дій, функції винагороди та мережевих наслідків прийнятих рішень.

Галузь застосування – інтелектуальні транспортні системи, адаптивне світлофорне керування, моделювання міських транспортних потоків, системи підтримки прийняття рішень у транспортній інфраструктурі та дослідження методів штучного інтелекту для керування складними динамічними системами.

ABSTRACT

Master’s thesis contains: 245 pp., 21 fig., 25 tabl., 7 ann., 18 references.

ADAPTIVE CONTROL, ARTIFICIAL INTELLIGENCE, MULTI-AGENT SYSTEMS, REINFORCEMENT LEARNING, TRAFFIC FLOWS, TRAFFIC SIGNAL CONTROL.

The object of the study is the process of traffic signal control of urban traffic flows under conditions of dynamic traffic demand.

The subject of the study is the methods for developing, training, and evaluating a multi-agent system for controlling traffic signal phases in a simulation environment.

The aim of the work is to study methods for developing a multi-agent learning system for traffic signal control in order to optimize urban traffic flows.

The research methods include the analysis of the subject area of traffic signal control, reinforcement learning and multi-agent learning methods, traffic flow modeling, DQN and PPO algorithms, and experimental comparison of models based on indicators such as delays, queues, waiting time, time loss, and the number of completed trips.

The scientific novelty of the work lies in the substantiation of an approach to representing traffic signal control in an urban transport network as a multi-agent learning problem, taking into account the local states of agents, admissible actions, the reward function, and the network-level consequences of the decisions made.

The field of application includes intelligent transport systems, adaptive traffic signal control, urban traffic flow modeling, decision-support systems in transport infrastructure, and research on artificial intelligence methods for controlling complex dynamic systems.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів 9
Вступ 11
1 Аналіз предметної галузі та постановка задачі 13
1.1 Аналіз предметної галузі світлофорного регулювання міських транспортних потоків 13
1.2 Проблематика оптимізації світлофорного керування в умовах динамічного транспортного попиту 18
1.3 Огляд традиційних методів світлофорного регулювання 24
1.4 Огляд інтелектуальних методів оптимізації транспортних потоків .. 30
1.5 Аналіз мультиагентного підходу до керування світлофорними об’єктами 36
1.6 Аналіз програмних засобів і симуляційних середовищ для моделювання транспортних потоків 41
1.7 Формалізація задачі дослідження 47
1.8 Постановка задачі та завдання роботи 52
1.9 Висновки до розділу 55
2 Теоретичне дослідження методів штучного інтелекту для розв’язання задачі світлофорного регулювання 56
2.1 Загальна характеристика задачі навчання керування в транспортних системах 56
2.2 Загальна характеристика задачі навчання керування в транспортних системах 61
2.3 Математична постановка задачі як марковського процесу прийняття рішень 67
2.4 Методи Q-навчання та Deep Q-Network для керування світлофорами 72
2.5 Методи політик і алгоритм Proximal Policy Optimization 78
2.6 Actor-Critic підходи в задачах адаптивного керування рухом 84

2.7 Мультиагентне підкріплювальне навчання 88
2.8 Формування простору станів для агента світлофорного регулювання 96
2.9 Формування простору дій агента 102
2.10 Проєктування функції винагороди 107
2.11 Метрики оцінювання ефективності світлофорного керування 112
2.12 Обмеження застосування методів підкріплювального навчання в реальних транспортних системах 116
3 Розробка методології та архітектури мультиагентної системи навчання світлофорного регулювання 120
3.1 Обмеження застосування методів підкріплювального навчання в реальних транспортних системах 120
3.2 Архітектура мультиагентної системи 125
3.3 Модель транспортної мережі та її елементи 133
3.4 Представлення світлофорного об’єкта як інтелектуального агента 139
3.5 Організація взаємодії агентів у транспортній мережі 144
3.6 Формування вхідних даних і стану середовища 149
3.7 Проєктування простору керуючих дій 154
3.8 Розробка функції винагороди для оптимізації транспортного потоку 158
3.9 Вибір і порівняння підходів до навчання мультиагентної системи на основі DQN та PPO 162
3.10 Архітектура нейромережевих моделей DQN та PPO для світлофорного агента 167
3.11 Методологія навчання та підбору параметрів для DQN і PPO 171
3.12 Сценарії експериментального дослідження та схема порівняння DQN і PPO 179
3.13 Висновки до розділу 184
4 Програмна реалізація, експериментальне налаштування та оцінка ефективності 185

4.1 Програмна реалізація мультиагентної системи світлофорного регулювання 185
4.2 Підготовка SUMO-сценаріїв, генерація транспортного попиту та baseline-контролери 191
4.3 Реалізація навчання моделей DQN і PPO в мультиагентному середовищі 194
4.4 Автоматичний підбір параметрів DQN і PPO та організація експериментальних запусків 199
4.5 Збір результатів, аналіз метрик і порівняння ефективності DQN та PPO 204
4.6 Перевірка стійкості найкращих моделей DQN і PPO до зміни інтенсивності транспортного потоку 211
4.7 Висновки до розділу 215
Висновки 216
Перелік джерел посилання 218
Додаток А Фрагменти програмної реалізації нейромережевого агента 220
Додаток Б Конфігураційні параметри SUMO-сценаріїв 228
Додаток В Структури вхідних даних експериментального середовища 229
Додаток Г Оцінювання моделей і збереження метрик 231
Додаток Д Програмна реалізація launcher-а 232
Додаток Е Результати підсумкових матеріалів auto-tuning 238
Додаток Ж Відомість кваліфікаційної роботи 245

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

CLIP – Clipped surrogate objective – обмежена, або кліпована, сурогатна цільова функція;

CSV – Comma-Separated Values – значення, розділені комами;

DQN – Deep Q-Network – глибока Q-мережа;

GAE – Generalized Advantage Estimation – узагальнене оцінювання переваги;

GPS – Global Positioning System – глобальна система позиціонування;

GRU – Gated Recurrent Unit – рекурентний блок із вентилями;

GUI – Graphical User Interface – графічний інтерфейс користувача;

ID – Identifier – ідентифікатор;

JSON – JavaScript Object Notation – об’єктна нотація JavaScript;

MARL – Multi-Agent Reinforcement Learning – мультиагентне підкріплювальне навчання;

MATSim – Multi-Agent Transport Simulation – мультиагентне транспортне моделювання;

MDP – Markov Decision Process – марковський процес прийняття рішень;

OPAC – Optimized Policies for Adaptive Control – оптимізовані політики адаптивного керування;

PPO – Proximal Policy Optimization – проксимальна оптимізація політики;

ReLU – Rectified Linear Unit – випрямлена лінійна функція активації;

RHODES – Real-time Hierarchical Optimized Distributed Effective System – ієрархічна оптимізована розподілена система керування в реальному часі;

RL – Reinforcement Learning – підкріплювальне навчання;

SCATS – Sydney Coordinated Adaptive Traffic System – Сіднейська координована адаптивна система керування рухом;

SCOOT – Split Cycle Offset Optimization Technique – метод оптимізації розподілу фаз, циклу та зсуву світлофорного регулювання;

SUMO – Simulation of Urban MObility – моделювання міської мобільності;

TraCI – Traffic Control Interface – інтерфейс керування дорожнім рухом;

XML – Extensible Markup Language – розширювана мова розмітки.

ВСТУП

Світлофорне регулювання є одним із ключових механізмів керування міськими транспортними потоками, оскільки саме на регульованих перехрестях виникає потреба у розподілі обмеженої пропускної здатності між взаємно конфліктними напрямками руху. У міських умовах транспортний попит має нерівномірний і динамічний характер: інтенсивність потоків змінюється залежно від часу доби, локальних затримок, маршрутної поведінки водіїв, стану суміжних ділянок мережі та випадкових збурень. Через це фіксовані плани світлофорного керування не завжди забезпечують раціональне використання зеленого часу, а локальне налаштування одного перехрестя може спричиняти перенесення перевантаження на сусідні вузли транспортної мережі.

Застосування методів штучного інтелекту, зокрема підкріплювального та мультиагентного навчання, дає змогу розглядати світлофорні об’єкти як інтелектуальних агентів, що приймають рішення на основі поточного стану транспортного середовища. Такий підхід є актуальним, оскільки він дозволяє перейти від статичного налаштування фаз до адаптивного керування, у якому враховуються черги, затримки, пропуск транспортних засобів, частота перемикань і вплив рішень на суміжні перехрестя. Водночас використання навчальних алгоритмів у транспортних системах потребує обережного врахування інженерних обмежень: безпеки фаз, мінімальних тривалостей сигналів, часткової спостережуваності та стабільності поведінки агентів [1], [7].

Об’єктом дослідження є процес світлофорного керування міськими транспортними потоками в умовах динамічного транспортного попиту.

Предметом дослідження є методи побудови, навчання та експериментального оцінювання мультиагентної системи керування світлофорними фазами у симуляційному середовищі SUMO.

Метою даного дослідження є дослідження методів розробки мультиагентної системи навчання світлофорного регулювання для оптимізації міських транспортних потоків. Досягнення цієї мети передбачає аналіз предметної області світлофорного керування, розгляд традиційних та інтелектуальних підходів до оптимізації транспортних потоків, формалізацію задачі як процесу прийняття рішень, проєктування простору станів, дій і функції винагороди, а також реалізацію експериментального контуру для навчання та порівняння алгоритмів DQN і PPO в середовищі SUMO [8], [16].

Наукова новизна дослідження полягає в обґрунтуванні та розробленні підходу до подання світлофорного регулювання міської транспортної мережі як мультиагентної задачі навчання, у якій кожне регульоване перехрестя розглядається як окремий агент із локальним станом, допустимим простором дій і навчальним сигналом, що відображає транспортні наслідки прийнятих рішень. На відміну від підходів, орієнтованих лише на ізольоване перехрестя або фіксований світлофорний план, у роботі враховується необхідність узгодження локальних рішень агентів із мережевими ефектами, зокрема накопиченням черг, зміною затримок і впливом на суміжні вузли.

Практичне значення роботи полягає у створенні програмно-експериментального контуру, що поєднує SUMO-моделювання, генерацію транспортного попиту, навчання моделей DQN і PPO, автоматизований підбір параметрів, збирання `tripinfo`-метрик і побудову графіків для порівняльного аналізу. Це дає змогу оцінювати якість світлофорного керування не лише за внутрішніми показниками навчання, а й за фактичними транспортними метриками: середнім часом поїздки, часом очікування, втратами часу, кількістю завершених поїздок і поведінкою системи за різних масштабів попиту. Таким чином, дослідження поєднує теоретичну постановку задачі мультиагентного навчання з практичною перевіркою алгоритмів у відтворюваному симуляційному середовищі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

Інтелектуальне регулювання транспортних потоків є ключовим напрямом сучасних транспортних систем, що формується на перетині теорії транспортних потоків, теорії керування, теорії масового обслуговування та методів штучного інтелекту. Предметна область охоплює закономірності формування, розподілу та управління транспортними потоками у вулично-дорожній мережі з метою підвищення ефективності руху, зменшення затримок, мінімізації викидів і підвищення безпеки дорожнього руху.

1.1 Аналіз предметної галузі світлофорного регулювання міських транспортних потоків

Світлофорне регулювання є одним із базових інструментів керування міськими транспортними потоками, оскільки саме на регульованих перехрестях відбувається просторово-часовий розподіл взаємно конфліктних напрямків руху. Для міської мережі перехрестя не є лише геометричним місцем перетину доріг. Воно функціонує як вузол прийняття керуючих рішень, у якому обмежений дорожній простір послідовно надається різним групам учасників руху: транспортним потокам основного й другорядного напрямків, поворотним потокам, пішоходам, громадському транспорту та спеціальним транспортним засобам. Саме тому якість світлофорного керування безпосередньо впливає не тільки на локальну затримку біля окремого перехрестя, а й на стабільність руху в суміжних ділянках вулично-дорожньої мережі.

У контексті дослідження міських транспортних потоків регульоване перехрестя доцільно розглядати як критичну точку мережі, де виникає потреба в узгодженні попиту на проїзд із реальною пропускною здатністю підходів. На відміну від лінійної ділянки дороги, де транспортні засоби переважно рухаються в одному або кількох паралельних напрямках,

перехрестя поєднує потоки з різними траєкторіями, пріоритетами та часовими вимогами. Прямий рух, ліві й праві повороти, пішохідні переходи та можливий рух громадського транспорту створюють конфліктні точки, які не можуть бути одночасно відкритими для всіх учасників. Світлофорний цикл у цьому разі виконує функцію механізму впорядкування: він розділяє конфліктні маневри в часі, зменшує ризик аварійних ситуацій і дає змогу підтримувати керований режим руху. Водночас таке впорядкування має ціну у вигляді затримок, накопичення черг і втрат пропускної здатності під час перехідних інтервалів. Через це завдання світлофорного регулювання не зводиться до простого почергового дозволу руху, а передбачає пошук балансу між безпекою, пропускною здатністю, рівномірністю обслуговування напрямків і стійкістю всієї транспортної мережі [1].

Транспортний потік у межах підходу до перехрестя характеризується інтенсивністю прибуття транспортних засобів, складом потоку, швидкістю руху, щільністю, часткою поворотних маневрів і нерівномірністю прибуття в часі. Для світлофорного керування особливо важливою є не лише середня інтенсивність, а й характер її коливань. Навіть якщо середнє навантаження на підхід є прийнятним, короткочасні пікові прибуття можуть спричинити утворення черг, які не встигають розвантажитися протягом одного циклу. У таких умовах фіксовані параметри світлофорного плану можуть виявитися недостатньо гнучкими, оскільки вони орієнтовані на усереднений стан, а не на поточну динаміку потоку.

Підхід до перехрестя є ділянкою, на якій транспортні засоби переходять від вільного або частково обмеженого руху до режиму очікування, розгону та проходження стоп-лінії. Його стан визначається не тільки кількістю смуг і дозволеними напрямками руху, а й тим, як саме транспортний попит розподіляється між цими смугами. Наприклад, окрема смуга для лівого повороту може зменшувати блокування прямого руху, але за надмірного попиту на поворотний маневр вона сама стає джерелом затримок і може впливати на сусідні смуги. Якщо ж поворотні потоки не

виділені просторово, затримка одного транспортного засобу здатна поширюватися на транспортні засоби з іншими траєкторіями. Для інтелектуальної системи керування це означає, що стан перехрестя не можна описувати лише загальною кількістю автомобілів на вході. Потрібно враховувати структуру попиту за напрямками, довжину черг на окремих смугах, наявність блокування, залишкову чергу після завершення зеленого сигналу та взаємний вплив суміжних підходів.

Світлофорний цикл складається з послідовності фаз, кожна з яких надає право руху певній групі сумісних потоків. Фаза має тривалість дозволеного руху, а між фазами використовуються перехідні інтервали, необхідні для безпечного завершення маневрів і підготовки до відкриття наступної групи напрямків. Практичне значення фазової структури полягає в тому, що вона визначає, які саме потоки можуть обслуговуватися одночасно, а які змушені очікувати. Надто проста фазова схема може створювати небезпечні або неефективні конфлікти між маневрами, тоді як надмірне дроблення циклу збільшує втрати часу на перемикання та зменшує частку ефективного зеленого сигналу. У працях із теорії та практики налаштування світлофорів підкреслюється, що вибір фаз, тривалості циклу й розподілу зеленого часу має спиратися на транспортний попит, геометрію перехрестя та операційні цілі керування [2].

Черги біля регульованого перехрестя виникають тоді, коли інтенсивність прибуття транспортних засобів протягом заборонного сигналу або недостатньо ефективної зеленої фази перевищує можливість їхнього обслуговування. Черга є важливим індикатором стану транспортної системи, оскільки вона показує накопичений наслідок попередніх керуючих рішень, а не лише миттєве навантаження. Якщо після завершення зеленої фази на підході залишається значна кількість транспортних засобів, це свідчить про недостатнє обслуговування відповідного напрямку або про перевищення попиту над пропускною здатністю. У разі повторення такої ситуації протягом кількох циклів формується залишкова черга, яка

поступово збільшує затримку, знижує середню швидкість руху й може призводити до блокування попередніх перехресть. Особливо небезпечним є поширення черги за межі виділеної смуги або підходу, оскільки тоді локальна проблема перетворюється на мережевий ефект: транспортні засоби, які не мають наміру рухатися через перевантажений напрямок, також починають втрачати час через фізичне перекриття смуг.

Затримка є одним із найважливіших показників оцінювання світлофорного регулювання, але її зміст ширший за простий час очікування на червоний сигнал. Вона охоплює втрати часу через зниження швидкості перед стоп-лінією, очікування в черзі, повільний розгін після відкриття руху та можливе повторне очікування, якщо транспортний засіб не встигає пройти перехрестя за один цикл. Для міського руху саме затримка часто є тим показником, який найвиразніше відображає якість керування з позиції користувача транспортної системи. Водночас мінімізація затримки лише на одному підході не завжди є коректною ціллю, оскільки вона може призвести до погіршення умов для інших напрямків. Тому світлофорне регулювання потребує оцінювання не ізольованого виграшу окремого потоку, а розподілу затримок між усіма учасниками руху.

Пропускна здатність регульованого перехрестя залежить від геометрії підходів, кількості смуг, дозволених маневрів, характеристик транспортного потоку та параметрів світлофорного циклу. У міських умовах вона не є сталою величиною, оскільки фактична можливість пропуску транспортних засобів змінюється під впливом пішохідних фаз, поворотних конфліктів, зупинок громадського транспорту, порушень правил паркування, погодних умов і нерівномірності прибуття автомобілів. Методики аналізу пропускної здатності й рівня обслуговування регульованих перехресть зазвичай розглядають взаємозв’язок між попитом, насиченим потоком, зеленим часом і затримкою, що дає змогу оцінити, наскільки ефективно перехрестя використовує доступний дорожній простір [3]. Для задачі побудови інтелектуальної системи це має

принципове значення: алгоритм керування повинен працювати не з абстрактним перехрестям, а з обмеженою системою, у якій кожне продовження зеленої фази для одного напрямку фактично зменшує ресурс обслуговування інших напрямків.

Світлофорне керування може бути реалізоване за різними принципами. Найпростішим є жорстке керування з наперед заданими тривалостями фаз і циклу. Його перевага полягає в передбачуваності та простоті налаштування, однак такий підхід погано реагує на короткочасні зміни попиту, аварійні ситуації або нерегулярні пікові навантаження. Координоване керування групою перехресть дає змогу формувати «зелені хвилі» для магістральних напрямків, але воно потребує узгодження між сусідніми вузлами та може знижувати гнучкість локального реагування. Адаптивні підходи використовують дані про поточний стан руху, змінюючи параметри керування відповідно до фактичного навантаження. Саме в цьому напрямі виникає зв’язок із методами штучного інтелекту: перехрестя може розглядатися як агент, що спостерігає локальний стан середовища, обирає керуючу дію та отримує наслідок у вигляді зміни черг, затримок і пропуску транспортних засобів.

Для подальшої побудови мультиагентної системи важливо визначити, які характеристики транспортного потоку мають бути представлені в моделі стану. До таких характеристик належать довжина черги на кожному підході, кількість транспортних засобів, що прибувають до стоп-лінії, час очікування, фаза, яка діє в поточний момент, залишковий час фази, наявність перевантаження, а також інформація про суміжні перехрестя. Останній аспект є особливо важливим для міської мережі, оскільки дія одного світлофора впливає на майбутній стан сусіднього вузла. Якщо один агент занадто довго утримує зелений сигнал для певного напрямку, він може передати великий транспортний пакет на наступне перехрестя, яке не готове його прийняти. У результаті локально обґрунтоване рішення здатне погіршити мережеву ситуацію.

Предметна галузь світлофорного регулювання також пов’язана з обмеженнями, які не можна ігнорувати під час побудови інтелектуальної системи. Алгоритм керування не має права порушувати вимоги безпеки, мінімальні тривалості сигналів, перехідні інтервали, правила обслуговування пішоходів і логіку сумісності фаз. Крім того, у реальному середовищі дані з детекторів можуть бути неповними, запізнілими або зашумленими. Це означає, що система навчання повинна враховувати не тільки оптимізаційний критерій, а й надійність спостережень, обмеженість керуючих дій і можливість виникнення нестандартних ситуацій. Без такого врахування інтелектуальний підхід може демонструвати прийнятні результати в моделюванні, але втрачати практичну придатність при перенесенні на реальну транспортну мережу.

1.2 Проблематика оптимізації світлофорного керування в умовах динамічного транспортного попиту

Оптимізація світлофорного керування в міській транспортній мережі ускладнюється тим, що транспортний попит не є сталим у часі. Якщо в попередньому підрозділі світлофорне регулювання розглядалося як механізм просторово-часового розподілу конфліктних потоків на перехресті, то в цьому підрозділі доцільно зосередитися на проблемі мінливості самого навантаження. Для міського середовища характерні добові, тижневі та ситуативні коливання інтенсивності руху, через які один і той самий світлофорний план може бути прийнятним в одних умовах і неефективним в інших.

Динамічний транспортний попит формується під впливом багатьох взаємопов’язаних чинників. У ранкові години переважають потоки в напрямку ділових, освітніх і промислових зон, тоді як у вечірній період навантаження часто зміщується у протилежному напрямку. Упродовж дня структура руху змінюється через службові поїздки, доставку товарів,

переміщення громадського транспорту, активність пішоходів і нерівномірний розподіл поїздок приватними автомобілями. Окремі відхилення виникають унаслідок дорожньо-транспортних пригод, ремонтних робіт, несприятливих погодних умов, масових заходів або тимчасових обмежень руху. Такі ситуації не лише збільшують кількість транспортних засобів на окремих підходах, а й змінюють напрямкову структуру потоку: частина водіїв обирає альтернативні маршрути, що переносить перевантаження на сусідні вулиці та перехрестя. У результаті транспортна мережа функціонує як взаємозалежна система, у якій локальне збурення може швидко перетворитися на ширший мережевий ефект.

Особливо складною для світлофорного керування є нерівномірність прибуття транспортних засобів. Навіть за відносно стабільної середньої інтенсивності потік може надходити групами, наприклад після відкриття зеленого сигналу на попередньому перехресті. Якщо світлофорний об’єкт не враховує таку хвильову структуру прибуття, частина ефективного зеленого часу може витрачатися не в момент найбільшого попиту, а тоді, коли відповідний підхід ще не має достатньої кількості транспортних засобів. Це призводить до нераціонального використання пропускної здатності, оскільки один напрямок отримує ресурс обслуговування без фактичної потреби, тоді як інші напрямки накопичують черги.

Фіксовані плани світлофорного регулювання традиційно будуються на основі попередньо зібраних даних про інтенсивність руху, структуру потоків і характерні періоди доби. Такий підхід має практичну цінність, оскільки він забезпечує передбачуваність роботи світлофорного об’єкта, спрощує налаштування та дає змогу врахувати базові вимоги безпеки. Проте його обмеження проявляються саме в умовах змінного попиту. План, розрахований для середнього навантаження, не завжди здатний адекватно реагувати на короткочасні піки, випадкові затримки або різку зміну напрямку основного потоку. Наприклад, якщо тривалість зеленої фази для другорядного напрямку залишається незмінною незалежно від фактичної

кількості транспортних засобів, то в періоди низького попиту частина циклу використовується неефективно. Натомість у періоди високого попиту той самий інтервал може бути недостатнім, що спричиняє залишкові черги й повторне очікування протягом кількох циклів. У методичних підходах до налаштування світлофорів підкреслюється, що параметри циклу, фаз і зеленого часу мають відповідати реальному транспортному попиту, однак на практиці цей попит часто змінюється швидше, ніж оновлюються фіксовані плани [1].

Наслідком такої невідповідності стає накопичення затримок. Затримка в умовах динамічного попиту має не лише локальний, а й часовий характер: рішення, яке було прийняте в одному циклі, впливає на стан наступних циклів. Якщо транспортні засоби не встигають залишити підхід під час дозволеного сигналу, залишкова черга стає початковою умовою для наступного інтервалу регулювання. За повторення цієї ситуації система поступово переходить від короткочасного перевантаження до стійкого заторового стану.

Проблема ускладнюється тим, що транспортні потоки на різних підходах перехрестя не є незалежними. Збільшення зеленого часу для одного напрямку може зменшити затримку саме на цьому підході, але водночас збільшити очікування для інших потоків, зокрема для пішоходів, громадського транспорту або лівоповоротних маневрів. Через це оптимізація не може зводитися до простого реагування на найбільшу чергу. Якщо система щоразу надає пріоритет найзавантаженішому напрямку, слабші або нерегулярні потоки можуть отримувати недостатнє обслуговування. У транспортній практиці це створює проблему справедливого розподілу часу між учасниками руху: потрібно не лише мінімізувати сумарну затримку, а й уникати надмірного погіршення умов для окремих груп користувачів. Для міських перехресть це особливо суттєво, оскільки транспортна ефективність має поєднуватися з безпекою,

доступністю пішохідних переходів і стабільністю руху громадського транспорту.

Окреме значення має вплив громадського транспорту. Автобуси, тролейбуси або трамваї перевозять значну кількість пасажирів, тому їхня затримка не завжди може оцінюватися так само, як затримка одного легкового автомобіля. Водночас надання пріоритету громадському транспорту без урахування поточного стану суміжних потоків може порушити баланс роботи перехрестя. Якщо зелений сигнал продовжується для пропуску одного маршрутного транспортного засобу, це може бути виправданим з позиції пасажирських втрат часу, але за високого навантаження на інших підходах таке рішення здатне посилити черги й погіршити загальну стійкість регулювання.

Пішохідні потоки також роблять задачу світлофорного керування менш одновимірною. У районах із високою щільністю забудови, біля станцій громадського транспорту, навчальних закладів, торговельних центрів або адміністративних установ пішохідний попит може суттєво змінюватися протягом коротких проміжків часу. На відміну від транспортних засобів, пішоходи мають інші вимоги до безпеки, часу очікування та тривалості переходу. Скорочення пішохідної фази заради збільшення пропуску автомобілів не завжди є допустимим, навіть якщо воно формально зменшує транспортну затримку. Крім того, надмірне очікування пішоходів може провокувати порушення сигналів, що безпосередньо впливає на безпеку руху. Тому в умовах динамічного попиту світлофорне керування має враховувати не лише автомобільні черги, а й багатокомпонентну структуру міського руху, де різні учасники мають різні часові та безпекові обмеження.

Ще однією проблемою є обмежена передбачуваність транспортної ситуації. Частина змін має регулярний характер і може бути врахована за допомогою типових добових планів, однак значна кількість збурень виникає випадково. Аварія на суміжній ділянці, тимчасове перекриття смуги,

затримка громадського транспорту на зупинці або різке погіршення погодних умов змінюють фактичну пропускну здатність дороги та поведінку водіїв. У таких випадках історичні середні значення втрачають достатню інформативність. Система керування повинна реагувати на поточний стан, а не лише на заздалегідь заданий розклад. Саме тому в сучасних підходах до організації світлофорного регулювання зростає роль детекторів руху, камер, сенсорних даних і алгоритмів, здатних інтерпретувати ситуацію в реальному або близькому до реального часу [3].

Однак наявність даних сама по собі не розв’язує задачу оптимізації. Дані можуть бути неповними, зашумленими, запізнілими або обмеженими лише певною зоною спостереження. Наприклад, детектор може фіксувати наявність транспортного засобу біля стоп-лінії, але не завжди дає повне уявлення про довжину черги, швидкість її зростання або причини блокування смуг. Камери можуть забезпечувати ширший огляд, однак їхня ефективність залежить від освітлення, погодних умов, якості розпізнавання та коректності інтерпретації транспортних об’єктів. Для інтелектуального керування це означає, що алгоритм має працювати не з ідеальним описом середовища, а з наближеним образом транспортної ситуації. Неправильна оцінка стану може призвести до помилкового вибору фази, надмірного продовження зеленого сигналу або недостатнього обслуговування перевантаженого напрямку.

Суттєва складність полягає також у виборі критерію оптимізації. Для світлофорного керування можуть розглядатися різні цілі: зменшення середньої затримки, скорочення довжини черг, збільшення пропуску транспортних засобів, зниження кількості зупинок, підтримання пріоритету громадського транспорту, запобігання блокуванню суміжних перехресть або вирівнювання умов руху між напрямками. Ці цілі не завжди узгоджуються між собою. Наприклад, мінімізація сумарної затримки може призвести до того, що малонавантажений напрямок очікуватиме непропорційно довго, тоді як обмеження максимальної черги може

потребувати частішого перемикання фаз і збільшувати втрати часу на перехідні інтервали. У класичних методах розрахунку світлофорних параметрів ця суперечність частково враховується через вибір тривалості циклу, розподілу зеленого часу та оцінку затримок [2]. Проте в динамічному середовищі оптимальний розподіл ресурсу змінюється разом зі станом мережі, тому статичний критерій не завжди відображає фактичну якість керування.

У мережі з кількох перехресть проблема набуває ще більшої складності. Рішення окремого світлофорного об’єкта впливає на прибуття транспортних засобів до сусідніх вузлів. Якщо перехрестя пропускає великий транспортний пакет у напрямку, де наступний вузол уже перевантажений, локально ефективне рішення може створити затор нижче за потоком. Навпаки, надмірне стримування потоку на одному перехресті може зменшити перевантаження на наступному, але спричинити накопичення черги на попередній ділянці. Це створює потребу в координації, однак жорстка координація не завжди достатньо гнучка для непередбачуваних змін. Для міської мережі характерна ситуація, коли локальні та глобальні цілі керування частково конфліктують: перехрестя має реагувати на власний стан, але його дії повинні бути узгоджені з ширшою транспортною ситуацією.

Саме ця особливість робить задачу оптимізації світлофорного регулювання природно придатною для розгляду в межах мультиагентного підходу. Кожне перехрестя може інтерпретуватися як агент, що спостерігає локальні параметри руху, приймає рішення щодо фаз або тривалості сигналів і впливає на стан сусідніх агентів. Проте така постановка не усуває складності, а лише переводить її в іншу площину. Потрібно визначити, яку інформацію агент повинен отримувати, як він має враховувати дії сусідів, яким чином уникати конкуренції між перехрестями та як забезпечити дотримання обмежень безпеки. Крім того, агентне рішення має бути достатньо стабільним: світлофорне керування не може змінюватися

хаотично лише через короткочасні флуктуації потоку, оскільки надмірна реактивність здатна збільшити кількість перемикань і погіршити передбачуваність руху.

1.3 Огляд традиційних методів світлофорного регулювання

Традиційні методи світлофорного регулювання сформували практичну основу керування міськими перехрестями задовго до активного використання методів штучного інтелекту. Їхнє значення полягає в тому, що вони забезпечують зрозумілу, нормативно контрольовану й технічно надійну організацію руху в умовах обмеженої пропускної здатності перехресть. У контексті цієї роботи такі методи доцільно розглядати не як застарілу альтернативу інтелектуальним системам, а як вихідну базу, від якої відштовхується подальший розвиток адаптивного та мультиагентного керування.

Найпоширенішим класичним підходом є світлофорне регулювання з фіксованим циклом. Його сутність полягає у використанні наперед заданої послідовності фаз і сталої тривалості кожної з них у межах певного періоду доби. Такий режим зазвичай формується на основі попередніх обстежень транспортного потоку, статистики інтенсивності руху, напрямкового розподілу автомобілів, пішохідного попиту та вимог безпеки. Практична цінність фіксованого циклу полягає в простоті впровадження, передбачуваності для учасників руху та зручності технічного обслуговування. Для перехресть зі стабільним добовим профілем навантаження та незначними випадковими коливаннями такий підхід може забезпечувати прийнятну якість регулювання без складної інформаційної інфраструктури. У методичних матеріалах із налаштування світлофорів підкреслюється, що вибір тривалості циклу, фазової структури й розподілу зеленого часу має спиратися на співвідношення попиту, пропускної здатності та затримок [1]. Водночас саме орієнтація на заздалегідь оцінений

попит обмежує гнучкість цього методу. Якщо фактична ситуація суттєво відрізняється від тієї, для якої було розраховано план, світлофор продовжує працювати за попередньою логікою, навіть коли частина зеленого часу витрачається без реальної потреби або, навпаки, перевантажений напрямок отримує недостатній ресурс обслуговування.

Розрахунок тривалості фаз у традиційних методах спрямований на розподіл обмеженого часу циклу між конфліктними напрямками руху. На практиці цей розподіл повинен ураховувати інтенсивність транспортних потоків, кількість смуг, насичений потік, пішохідні інтервали, перехідні сигнали та допустимий рівень затримки. Класичні підходи дають змогу впорядкувати цей процес і перетворити налаштування світлофора з інтуїтивного рішення на інженерну процедуру. Зокрема, підхід Вебстера став одним із відомих прикладів формалізованого визначення параметрів сигналізації для ізольованого перехрестя [2].

Однак інженерна зручність таких методів не означає їхньої універсальності. Розрахунок, виконаний для середніх або пікових значень інтенсивності, часто не відображає короткочасних змін, які виникають у реальному міському русі. Якщо на одному підході протягом кількох циклів формується залишкова черга, фіксований план не завжди здатний швидко перерозподілити зелений час. Якщо ж попит на другорядному напрямку тимчасово зменшується, світлофор усе одно може надавати йому запланований інтервал, знижуючи ефективність обслуговування інших потоків. Особливо помітним це стає на перехрестях, де транспортні потоки мають нерівномірну структуру: вранці переважає один напрямок, удень активізуються локальні поїздки, а ввечері змінюється співвідношення прямого та поворотного руху. Через це у багатьох містах фіксоване регулювання реалізується не одним універсальним планом, а набором добових планів для ранкового піку, міжпікового періоду, вечірнього піку та нічного часу. Такий підхід підвищує практичну придатність системи, але не

усуває залежності від історичних даних і не забезпечує повноцінної реакції на випадкові збурення.

Окрему групу становлять методи координованого світлофорного регулювання, у яких кілька перехресть розглядаються не ізольовано, а як пов’язана послідовність вузлів уздовж магістралі або транспортного коридору. Найвідомішим прикладом є організація руху за принципом «зеленої хвилі». У цьому випадку моменти початку зелених сигналів на сусідніх перехрестях зміщуються так, щоб група транспортних засобів могла проходити декілька регульованих вузлів із мінімальною кількістю зупинок. Такий підхід має значну практичну цінність для напрямків із вираженим основним потоком, зокрема на міських магістралях, під’їздах до центрів ділової активності або транспортних коридорах із високою часткою транзитного руху.

Ефективність «зеленої хвилі» залежить від узгодження тривалості циклу, швидкості руху, відстані між перехрестями, структури фаз і величини часових зміщень. Якщо ці параметри підібрані коректно, координоване регулювання зменшує кількість зупинок, знижує середню затримку на магістральному напрямку та робить рух більш плавним. Проте така координація завжди пов’язана з певним пріоритетом одних потоків над іншими. Для бокових вулиць або поворотних напрямків це може означати збільшення часу очікування, особливо якщо основний коридор отримує значну частку зеленого часу. Крім того, «зелена хвиля» добре працює тоді, коли транспортні засоби рухаються приблизно з розрахунковою швидкістю і прибувають до наступного перехрестя передбачуваними групами. У щільній міській мережі ця умова часто порушується через зупинки громадського транспорту, паркування біля проїзної частини, нерівномірне прискорення автомобілів, пішохідні переходи, змішування потоків і випадкові затримки. У результаті часові зміщення, які були ефективними для одного режиму руху, можуть стати менш придатними за іншого навантаження.

Координоване регулювання також має мережеве обмеження. Якщо оптимізація виконується переважно для одного напрямку, система може погіршувати умови руху на суміжних вулицях, де накопичуються черги від другорядних підходів. У великих міських районах транспортний попит рідко має лише одну домінантну вісь. Потоки перерозподіляються між маршрутами, змінюються через дорожні роботи або аварійні ситуації, а також залежать від поведінки водіїв, які реагують на затори вибором альтернативних шляхів. Тому жорстка координація потребує регулярного перегляду планів і не завжди забезпечує стійкість у ситуаціях, де структура попиту швидко змінюється.

Проміжним кроком між фіксованим і повністю адаптивним керуванням є транспортно-залежне регулювання на основі детекторів. У таких системах світлофор не обов’язково змінює всю логіку роботи, але може продовжувати або скорочувати окремі фази залежно від наявності транспортних засобів на підходах. Наприклад, якщо на другорядному напрямку немає автомобілів, відповідна фаза може бути пропущена або зменшена до мінімально допустимої тривалості. Якщо ж детектори фіксують наявність черги, зелений сигнал може бути продовжений у межах заданих обмежень. Такий підхід дає змогу уникнути частини втрат, характерних для жорсткого циклу, але його можливості визначаються якістю детекторів, місцем їхнього встановлення та закладеними правилами реагування.

Транспортно-залежне керування добре підходить для ізольованих перехресть або вузлів із нерегулярним попитом на другорядних підходах. Його перевага полягає в тому, що система використовує поточну інформацію про наявність транспортних засобів, не потребуючи складного прогнозування стану всієї мережі. Проте локальна реакція не завжди є достатньою. Якщо світлофор продовжує фазу лише тому, що детектор фіксує транспортний засіб біля стоп-лінії, він може не враховувати накопичення черги на сусідньому перехресті або наслідки передачі великої

групи автомобілів нижче за потоком. Крім того, детектори часто надають обмежений опис ситуації: вони можуть показувати факт присутності автомобіля, але не завжди коректно відображають довжину черги, швидкість її зростання, структуру маневрів або причини затримки. Через це транспортно-залежне регулювання підвищує гнучкість порівняно з фіксованим планом, але не розв’язує повністю задачу оптимізації в умовах складної міської мережі.

Більш розвиненим напрямом традиційного керування є адаптивні системи світлофорного регулювання. Їхня відмінність полягає в тому, що параметри роботи світлофорів змінюються на основі поточних даних про транспортний потік, а не лише відповідно до заздалегідь заданого розкладу. Такі системи можуть коригувати тривалості фаз, розподіл зеленого часу, довжину циклу та часові зміщення між перехрестями. У науковій і практичній літературі до відомих прикладів належать SCOOT, SCATS, OPAC, RHODES та інші системи адаптивного керування, які відрізняються алгоритмічною логікою, рівнем централізації та способом використання детекторних даних [4].

Система SCOOT ґрунтується на постійному збиранні інформації про транспортні потоки та поступовій зміні параметрів сигналізації для групи пов’язаних перехресть. Її логіка полягає не в різкому перепланування роботи світлофорів, а в регулярному коригуванні розподілу зеленого часу, циклу та зміщень з урахуванням поточної ситуації. Така обережна адаптація має практичний сенс: надто різкі зміни сигналів можуть порушити стабільність руху, тоді як малі, але часті корекції дають змогу реагувати на зміну попиту без надмірної непередбачуваності. SCOOT особливо придатна для мереж, де важливо підтримувати узгоджену роботу декількох перехресть, оскільки вона орієнтована не лише на локальний стан окремого вузла, а й на координацію в межах міської ділянки.

Система SCATS також належить до адаптивних підходів, але має власну логіку організації керування. У загальному вигляді вона

використовує дані з детекторів для оцінювання завантаженості підходів і вибору або коригування параметрів роботи світлофорів у межах підсистеми. Практична цінність таких систем полягає в тому, що вони поєднують інженерно контрольовану структуру сигналізації з можливістю реагувати на поточний транспортний стан. На відміну від повністю статичних планів, адаптивна система не прив’язана лише до середніх значень попиту, а може змінювати режим роботи протягом дня. Водночас вона залишається залежною від якості первинних даних, коректності налаштування зон керування, працездатності обладнання та правил, закладених розробниками системи. Якщо детектори встановлені невдало або не відображають критичні ділянки накопичення черг, адаптивний алгоритм отримує неповну картину середовища і може приймати рішення, які формально відповідають виміряним даним, але не усувають фактичну причину затримок.

Порівняно з фіксованими планами адаптивні системи мають очевидну перевагу в умовах змінного попиту. Вони здатні частково компенсувати нерівномірність прибуття транспортних засобів, реагувати на відхилення від типового добового профілю та підтримувати координацію між перехрестями без повного ручного переналаштування планів. Проте їх не можна розглядати як остаточне розв’язання проблеми міського світлофорного керування. Значна частина таких систем побудована на евристичних правилах, інженерних припущеннях і поступовій оптимізації окремих параметрів. Це забезпечує надійність і контрольованість, але обмежує здатність системи навчатися на довгострокових наслідках власних дій, враховувати складні взаємодії між перехрестями та адаптуватися до сценаріїв, які не були достатньо представлені під час початкового налаштування.

Для теми цієї роботи особливо важливим є те, що традиційні методи здебільшого розглядають світлофорне керування як задачу добору параметрів у межах уже визначеної структури. Фіксований цикл визначає сталі тривалості фаз, координоване керування підбирає зміщення між

перехрестями, транспортно-залежне регулювання реагує на локальну присутність транспортних засобів, а адаптивні системи коригують параметри на основі детекторних даних. Усі ці підходи мають практичну цінність, оскільки вони забезпечили реальну експлуатацію світлофорних мереж і сформували набір вимог до безпеки, стабільності та керованості. Проте складна міська мережа створює ситуації, у яких локально коректне рішення не обов’язково є оптимальним для системи загалом. Продовження зеленої фази на одному перехресті може зменшити чергу на цьому підході, але водночас передати надлишковий потік до сусіднього вузла. Навпаки, стримування потоку може бути невигідним локально, але корисним для запобігання блокуванню нижче за маршрутом. Такі взаємозалежності важко повністю описати наперед заданими правилами.

1.4 Огляд інтелектуальних методів оптимізації транспортних потоків

Перехід від традиційних алгоритмів світлофорного регулювання до інтелектуальних методів зумовлений не лише ускладненням міського руху, а й зміною самого підходу до керування. Якщо класичні методи переважно спираються на заздалегідь визначені правила, розраховані параметри циклу або локальні реакції на детекторні дані, то методи штучного інтелекту дають змогу розглядати транспортну мережу як динамічне середовище, у якому керуючі рішення можуть уточнюватися на основі даних, прогнозів і наслідків попередніх дій.

У задачах оптимізації транспортних потоків штучний інтелект виконує кілька взаємопов’язаних функцій. По-перше, він використовується для аналізу поточного стану руху: визначення довжини черг, оцінювання завантаженості підходів, виявлення нерівномірності прибуття транспортних засобів і розпізнавання ситуацій, у яких фіксований план регулювання стає неефективним. По-друге, інтелектуальні методи застосовуються для прогнозування короткострокового транспортного попиту, що особливо

важливо для міських ділянок із хвильовим прибуттям автомобілів від сусідніх перехресть. По-третє, такі методи можуть підтримувати вибір керуючої дії: продовження або завершення фази, перерозподіл зеленого часу, зміна пріоритету напрямків чи координація кількох світлофорних об’єктів. Саме ця здатність поєднувати спостереження, прогноз і керування відрізняє інтелектуальні підходи від суто інженерних схем, у яких поведінка системи майже повністю задається наперед.

Машинне навчання є однією з базових груп методів, що застосовуються в інтелектуальних транспортних системах. Його практична цінність полягає в можливості виявляти залежності між параметрами транспортного потоку та якістю керування без повного ручного опису всіх причинно-наслідкових зв’язків. Для світлофорного регулювання це має особливе значення, оскільки одна й та сама інтенсивність руху може мати різні наслідки залежно від фазової структури, залишкових черг, розподілу поворотних маневрів і стану суміжних перехресть.

У межах машинного навчання можуть розглядатися як моделі прогнозування, так і моделі класифікації транспортних ситуацій. Наприклад, на основі історичних даних, отриманих із детекторів, камер або імітаційного моделювання, можна прогнозувати очікуване навантаження на підхід у наступні інтервали часу. Така інформація дає змогу не лише реагувати на вже сформовану чергу, а й запобігати її надмірному зростанню. Водночас машинне навчання не усуває потреби в коректному визначенні ознак стану. Якщо модель отримує лише загальну кількість транспортних засобів на підході, але не враховує напрямкову структуру, блокування смуг або вплив сусіднього вузла, її рішення можуть бути формально обґрунтованими з погляду вхідних даних, але недостатніми для реальної оптимізації. Через це підготовка даних, вибір релевантних ознак і перевірка стійкості моделі залишаються критичними умовами використання машинного навчання в транспортному керуванні.

Нейронні мережі розширюють можливості машинного навчання завдяки здатності обробляти складні нелінійні залежності. У транспортній сфері вони можуть використовуватися для прогнозування інтенсивності руху, оцінювання швидкості, виявлення заторових станів, аналізу відеопотоків і формування представлення стану для подальшого прийняття рішень. Для міського світлофорного регулювання важливо, що нейронна модель може враховувати не один ізольований показник, а комбінацію ознак: довжину черги, час очікування, поточну фазу, зміну попиту в часі, інтенсивність прибуття з попереднього перехрестя та інші параметри, які разом утворюють картину транспортної ситуації.

Однак нейронні мережі мають і суттєві обмеження. Вони потребують достатньої кількості репрезентативних даних, чутливі до якості навчальної вибірки та не завжди забезпечують прозоре пояснення отриманого рішення. У задачі світлофорного керування це створює додаткову проблему, оскільки система повинна залишатися не лише ефективною, а й контрольованою з погляду безпеки. Якщо модель рекомендує змінити фазу або продовжити зелений сигнал, потрібно розуміти, чи не призведе це до порушення мінімальних інтервалів, надмірного очікування пішоходів або блокування суміжного перехрестя. Тому нейронні мережі доцільно розглядати не як самодостатню заміну правилам безпеки, а як інструмент для глибшого аналізу стану та підтримки адаптивного прийняття рішень.

Окреме місце серед інтелектуальних підходів займає нечітка логіка. Її перевага полягає в тому, що вона дає змогу працювати з неповними, нечіткими або лінгвістично описаними даними, які часто виникають у транспортному керуванні. У практиці регулювання поняття на кшталт «довга черга», «високе навантаження», «коротке очікування» або «помірний потік» не завжди мають жорстку межу, але можуть бути зрозумілими для експертного аналізу. Нечіткі регулятори формалізують такі оцінки у вигляді правил, що пов’язують стан підходів із керуючими діями. Наприклад, якщо на одному напрямку черга є значною, а на конфліктному напрямку попит

низький, система може збільшити тривалість поточної фази в допустимих межах.

Практична привабливість нечіткої логіки полягає в її інтерпретованості. На відміну від багатьох нейромережевих моделей, нечіткий регулятор дозволяє простежити, яке правило спрацювало і чому було обрано певну дію. Це важливо для транспортних систем, де рішення повинні відповідати інженерним обмеженням і бути зрозумілими для налаштування. Водночас нечітка логіка значною мірою залежить від якості експертно сформованих правил і функцій належності. Якщо правила не враховують складні мережеві взаємодії або були налаштовані лише для типових сценаріїв, система може втрачати ефективність у нестандартних умовах. У літературі також підкреслюється, що нечіткі підходи часто комбінуються з іншими методами, зокрема з генетичними алгоритмами, щоб автоматизувати добір параметрів і зменшити залежність від ручного налаштування [5].

Еволюційні алгоритми застосовуються в транспортному керуванні як засоби пошуку прийнятних або близьких до оптимальних параметрів у складному просторі рішень. До таких параметрів можуть належати тривалості фаз, порядок їх перемикання, часові зміщення між сусідніми перехрестями, ваги критеріїв оптимізації або параметри нечітких правил. Їхня перевага полягає в тому, що вони не потребують простого аналітичного опису цільової функції й можуть працювати з багатокритеріальними задачами, де потрібно одночасно враховувати затримку, довжину черг, кількість зупинок і пропускну здатність.

Разом із тим еволюційні алгоритми мають обмеження, які стримують їх безпосереднє використання в режимі реального часу. Вони зазвичай потребують багаторазового оцінювання різних варіантів рішень, що може бути обчислювально витратним для великої міської мережі. Через це такі алгоритми часто доцільніше застосовувати для попереднього налаштування параметрів, оптимізації сценаріїв в імітаційному середовищі або поєднання

з іншими методами, які вже виконують оперативне керування. Для теми цієї роботи важливо, що еволюційні підходи демонструють можливість автоматизованого пошуку керуючих стратегій, але не повністю розв’язують проблему навчання на послідовних наслідках дій у динамічному транспортному середовищі.

Графові моделі природно відповідають структурі міської транспортної мережі. Перехрестя можна розглядати як вузли графа, а дорожні ділянки між ними – як ребра, через які передаються транспортні потоки. Такий опис дає змогу перейти від аналізу окремого світлофора до аналізу взаємозалежної системи, де рішення одного вузла впливає на майбутній стан інших вузлів. У задачах прогнозування транспортного попиту графові та графові нейронні моделі використовуються для врахування просторово-часових залежностей: транспортне навантаження на одному перехресті не є ізольованим, оскільки воно залежить від попередніх сигналів, маршрутного вибору водіїв і пропускної здатності сусідніх ділянок [6].

Застосування графових моделей особливо важливе для мультиагентного світлофорного керування. Якщо кожне перехрестя інтерпретується як агент, то граф задає структуру його взаємодії з іншими агентами. Локальний агент може отримувати інформацію не про всю міську мережу, а лише про суміжні вузли або найближчий транспортний коридор. Такий підхід зменшує складність задачі, але водночас створює проблему часткової спостережуваності: агент приймає рішення на основі обмеженої інформації, тоді як наслідки його дій можуть поширюватися далі за межі локального оточення. Отже, графове представлення не тільки описує топологію мережі, а й допомагає визначити, яку інформацію доцільно передавати між агентами для узгодженого керування.

Прогнозування транспортного попиту є ще одним важливим напрямом застосування штучного інтелекту. Для світлофорного регулювання прогноз має короткостроковий характер: система повинна

оцінити, які потоки прибудуть до перехрестя в найближчі інтервали, чи зростатиме черга на певному підході, чи очікується надходження транспортного пакета від попереднього вузла. На відміну від довгострокового транспортного планування, тут має значення не загальний добовий попит, а найближча динаміка, яка може вплинути на вибір наступної фази. Прогнозування дає змогу зробити керування менш реактивним: система не лише відповідає на вже сформований стан, а й готується до очікуваної зміни навантаження.

Найбільш безпосередньо з темою цієї роботи пов’язане підкріплювальне навчання. Його сутність полягає в тому, що агент навчається вибирати дії через взаємодію із середовищем і отримання оцінки наслідків цих дій. У контексті світлофорного регулювання агентом може бути окреме перехрестя, дія відповідає вибору або зміні фази, а стан середовища описується параметрами транспортного потоку. Якість дії може оцінюватися через зміну затримок, довжин черг, кількості зупинок або пропуску транспортних засобів. Принципова відмінність такого підходу від класичних правил полягає в тому, що стратегія керування не задається повністю наперед, а формується в процесі навчання на основі досвіду взаємодії з транспортним середовищем.

Для світлофорного керування підкріплювальне навчання є привабливим тому, що воно враховує послідовний характер рішень. Зміна фази в поточний момент впливає не тільки на миттєву чергу, а й на стан наступних циклів, прибуття транспортних засобів до сусідніх перехресть і можливість виникнення блокування. Класичні алгоритми часто оптимізують параметри в межах короткого інтервалу або реагують на поточне перевантаження, тоді як підкріплювальне навчання дає змогу оцінювати дію через її відкладені наслідки. Саме це робить його придатним для задач, у яких локально вигідне рішення може бути небажаним з погляду мережевої стійкості. У дослідженнях інтелектуальних транспортних систем підкреслюється, що глибоке підкріплювальне навчання активно

застосовується до задач керування сигналами, маршрутизації, автономного руху та інших складних транспортних процесів [7].

У мультиагентній постановці підкріплювальне навчання набуває ще більшого значення. Кожне перехрестя може навчатися власної політики керування, але його рішення впливають на стани інших агентів. Це наближує модель до реальної міської мережі, де неможливо повністю централізувати всі рішення без різкого зростання складності. Водночас мультиагентний підхід створює нові проблеми: потрібно забезпечити координацію агентів, уникнути конфлікту локальних цілей, визначити обсяг обміну інформацією та зберегти стабільність навчання. Якщо агенти діють лише в інтересах власного перехрестя, вони можуть зменшувати локальну затримку, але погіршувати стан сусідніх вузлів. Якщо ж система надмірно орієнтується на глобальний критерій, окремі напрямки можуть отримувати неприйнятно довге очікування. Через це підкріплювальне навчання в транспортній мережі потребує ретельного формування винагороди, обмежень безпеки й механізмів координації.

1.5 Аналіз мультиагентного підходу до керування світлофорними об’єктами

Міська транспортна мережа за своєю природою не є сукупністю незалежних перехресть, які можна оптимізувати окремо без урахування взаємного впливу. Кожний світлофорний об’єкт працює в певному локальному середовищі, однак наслідки його рішень виходять за межі одного вузла. Зміна фази, продовження зеленого сигналу або затримка пропуску окремого напрямку впливають на прибуття транспортних засобів до сусідніх перехресть, формуючи просторово-часову залежність між елементами мережі.

У контексті розглянутих раніше проблем динамічного попиту та обмежень традиційного керування мультиагентний підхід дає змогу

описати транспортну мережу ближче до її фактичної структури. Якщо ізольоване перехрестя розглядати як окрему задачу оптимізації, то основна увага зосереджується на виборі фаз і тривалостей сигналів для локальних підходів. Такий опис може бути достатнім для простих ділянок або для перехресть із незначним впливом на суміжні вузли. Проте в щільній міській мережі транспортні засоби переходять від одного світлофорного об'єкта до іншого, а черга, що виникла на одному підході, може фізично перекрити рух на попередній ділянці. Через це оптимізація одного світлофора без урахування сусідів здатна створювати прихований негативний ефект: локальне зменшення затримки на одному напрямку супроводжується зростанням навантаження нижче за потоком. Мультиагентне подання дозволяє перенести центр аналізу з окремого світлофорного циклу на взаємодію між керуючими вузлами, де кожне перехрестя має власний стан, власні дії та власний вплив на загальну транспортну ситуацію.

Агентом у такій системі доцільно вважати регульоване перехрестя або світлофорний контролер, який сприймає стан локального транспортного середовища та приймає рішення щодо керування сигналами. До локального стану можуть належати довжини черг, поточна фаза, інтенсивність прибуття транспортних засобів, час очікування на підходах і наявність залишкової черги. Однак для міської мережі важливо, що агент не існує ізольовано. Його рішення змінює не лише власні показники, а й умови, у яких через короткий проміжок часу працюватимуть інші агенти.

Доцільність такого підходу особливо помітна на прикладі транспортного коридору з кількома послідовними регульованими перехрестями. Якщо перший світлофор тривалий час утримує зелений сигнал для магістрального напрямку, він може швидко розвантажити власний підхід. З локального погляду це рішення виглядає обґрунтованим, оскільки черга перед стоп-лінією зменшується, а пропуск транспортних засобів зростає. Проте якщо наступне перехрестя в цей момент обслуговує конфліктний напрямок або вже має значну залишкову чергу, переданий

транспортний потік не буде прийнятий мережею без додаткових втрат. У результаті виникає накопичення транспортних засобів між перехрестями, можливе блокування смуг і зниження ефективності попереднього рішення. Саме така ситуація показує обмеженість локальної оптимізації: світлофор може покращити власний показник, але погіршити мережевий стан. Мультиагентна модель, навпаки, робить цей взаємний вплив частиною самої постановки задачі.

У мультиагентній системі кожне перехрестя має певний ступінь автономності. Це означає, що агент може приймати рішення на основі доступних йому спостережень, не очікуючи повного централізованого розрахунку для всієї мережі. Така властивість є важливою для транспортного керування, оскільки стан міського руху змінюється швидко, а повна інформація про всі вузли часто є недоступною або надходить із затримкою. Автономність, однак, не повинна перетворюватися на незалежність у вузькому сенсі. Агент має враховувати, що його дія є частиною спільного процесу, де рішення сусідів можуть посилювати або послаблювати очікуваний ефект.

Структурно міську транспортну мережу зручно інтерпретувати як граф, у якому перехрестя відповідають вузлам, а дорожні ділянки між – зв’язкам передавання транспортних потоків. Такий опис узгоджується з підходами до аналізу просторово-часових залежностей у транспортних системах, де навантаження на одному вузлі розглядається не як ізольована величина, а як результат взаємодії з попередніми й наступними ділянками мережі [6]. Для мультиагентного керування це має методичне значення: граф визначає не тільки фізичну топологію доріг, а й інформаційні межі взаємодії між агентами. Перехрестя може обмінюватися даними з безпосередніми сусідами, з вузлами транспортного коридору або з координаційним рівнем, який узагальнює мережевий стан. При цьому надмірне розширення інформаційного обміну може ускладнювати систему, збільшувати вимоги до даних і знижувати швидкість прийняття рішень.

Тому практично важливим є не максимальне збирання інформації, а визначення такого обсягу взаємодії, який справді покращує узгодженість керування.

Порівняно з централізованим підходом мультиагентна модель краще масштабується для великих міських мереж. Централізована система теоретично може враховувати стан усіх перехресть, однак зі збільшенням кількості вузлів різко зростає складність пошуку рішень, обсяг даних і чутливість до відмов окремих компонентів. Якщо центральний модуль отримує неповні або запізнілі дані, якість керування може погіршуватися одразу для значної частини мережі. У мультиагентній постановці частина рішень переноситься на локальний рівень, що робить систему гнучкішою та потенційно стійкішою до часткових інформаційних втрат.

Водночас мультиагентний підхід не усуває проблему координації, а робить її центральним елементом дослідження. Якщо кожний агент оптимізує лише власну затримку або власну довжину черги, між перехрестями може виникнути конкуренція за пропускну здатність мережі. Наприклад, агент, який намагається швидко очистити свій підхід, може передати надлишковий потік до сусіднього вузла, не врахувавши його поточну завантаженість. Інший агент, навпаки, може надмірно стримувати потік, щоб уникнути локального перевантаження, але цим збільшувати чергу на попередній ділянці. Такі конфлікти свідчать, що для ефективної мультиагентної системи недостатньо просто призначити кожному перехрестю окремий алгоритм. Потрібно визначити механізми узгодження дій: обмін інформацією про очікуване прибуття транспортних засобів, урахування стану сусідніх черг, спільні критерії винагороди або правила обмеження дій, які можуть погіршити мережеву стійкість.

Особливе значення в такій системі має часткова спостережуваність. Жодне окреме перехрестя зазвичай не має повного уявлення про стан усієї міської мережі. Воно може спостерігати власні підходи, отримувати дані від найближчих детекторів або повідомлення від сусідніх агентів, але не завжди

знає причини зміни потоку на віддалених ділянках. Через це агент приймає рішення в умовах обмеженої інформації, а наслідки його дій можуть проявлятися із затримкою. Для світлофорного регулювання це принципово: невдале рішення не завжди помітне одразу, оскільки його ефект може виникнути на наступному перехресті через певний час.

Підкріплювальне навчання природно поєднується з мультиагентним поданням, оскільки дозволяє розглядати керування як послідовний процес взаємодії агента із середовищем. У такій постановці світлофорний агент не просто виконує наперед задане правило, а поступово формує політику дій на основі наслідків попередніх рішень. Для окремого перехрестя це означає можливість навчитися обирати фазу або змінювати її тривалість залежно від поточного стану підходів. Для мережі загалом важливішим є те, що навчання може враховувати відкладені наслідки: дія, яка тимчасово збільшує очікування на одному підході, може бути виправданою, якщо вона запобігає блокуванню сусіднього вузла або стабілізує рух у коридорі. Саме здатність оцінювати рішення через подальший розвиток транспортної ситуації робить підкріплювальне навчання змістовно придатним для складних інтелектуальних транспортних систем [7].

Разом із тим використання мультиагентного навчання потребує обережності. Транспортна мережа є середовищем із високими вимогами до безпеки, тому агент не може вільно випробовувати будь-які дії, які формально можуть зменшити затримку. Мінімальні тривалості сигналів, перехідні інтервали, сумісність фаз, пішохідні переходи та вимоги до пріоритету окремих видів транспорту мають залишатися жорсткими обмеженнями. Крім того, навчання кількох агентів одночасно може бути нестабільним, оскільки середовище для кожного агента змінюється не лише через транспортний попит, а й через адаптацію інших агентів. Це відрізняє мультиагентну задачу від керування одним світлофором: агент навчається не в статичному середовищі, а в системі, де сусідні вузли також змінюють свою поведінку.

Практичний інтерес становлять різні рівні координації агентів. Найпростіший варіант передбачає незалежне навчання, коли кожне перехрестя використовує власні спостереження та власний критерій якості. Такий підхід є відносно простим, але він слабко враховує мережеві наслідки. Інший варіант полягає в обміні локальними станами між сусідніми агентами: перехрестя може отримувати інформацію про чергу або очікуване прибуття з попереднього вузла й коригувати рішення з урахуванням цієї інформації. Більш складною є схема, у якій агенти навчаються з урахуванням спільної або частково спільної винагороди, що відображає не лише локальні, а й мережеві показники. Кожний із цих підходів має компроміси. Надто локальна модель є простою, але може створювати конфлікти між вузлами. Надто глобальна модель краще враховує загальну ефективність, але може ускладнювати навчання й робити рішення менш чутливими до специфіки окремого перехрестя.

З позиції предметної області важливо також розрізняти координацію та централізацію. Координація не обов’язково означає, що всі рішення повинні прийматися одним керуючим центром. Вона може реалізовуватися через локальний обмін повідомленнями, спільні обмеження, погоджені критерії або ієрархічну структуру, у якій загальний рівень задає цілі, а локальні агенти адаптують дії до фактичного стану підходів. Такий підхід узгоджується з логікою адаптивного світлофорного керування, де поточні дані про рух використовуються для коригування параметрів сигналізації, але розширює її за рахунок агентної взаємодії та навчання на наслідках прийнятих рішень [4].

1.6 Аналіз програмних засобів і симуляційних середовищ для моделювання транспортних потоків

Дослідження методів навчання світлофорного регулювання потребує середовища, у якому можна відтворювати транспортний попит, фазову

структуру перехресть, утворення черг і наслідки керуючих рішень без втручання в реальну дорожню інфраструктуру. Для задачі, що розглядається в роботі, це має принципове значення: алгоритм світлофорного керування не може перевірятися безпосередньо на міській мережі на ранніх етапах розробки, оскільки помилкові дії можуть збільшити затримки, створити небезпечні конфлікти або погіршити умови руху на суміжних перехрестях. Тому симуляційне середовище виступає проміжною ланкою між теоретичною моделлю мультиагентного керування та її можливою практичною апробацією.

На відміну від статичного розрахунку параметрів світлофорного циклу, імітаційне моделювання дає змогу аналізувати транспортний потік як процес, що розгортається в часі. Це особливо важливо для підкріплювального навчання, де агент не лише обирає дію, а й отримує відкладений наслідок цієї дії у вигляді зміни черг, затримок, кількості зупинок або пропуску транспортних засобів. Якщо в попередніх підрозділах було показано, що міська мережа характеризується динамічним попитом, частковою спостережуваністю та взаємним впливом перехресть, то вибір симулятора має забезпечити можливість відтворити саме ці властивості. Недостатньо мати лише графічне зображення дороги або спрощений генератор автомобілів. Для дослідження потрібне середовище, у якому можна задавати структуру мережі, розподіл потоків за напрямками, логіку світлофорних фаз, детектори, маршрутний вибір, часові сценарії попиту та показники якості керування. Крім того, симулятор повинен дозволяти багаторазово запускати однакові або контрольовано змінені сценарії, оскільки порівняння алгоритмів є коректним лише тоді, коли вони перевіряються в зіставних умовах.

Програмні засоби моделювання транспортних потоків можна умовно розглядати за кількома ознаками: рівнем деталізації руху, доступністю для дослідника, підтримкою зовнішнього керування, можливістю масштабування та зручністю інтеграції з алгоритмами штучного інтелекту.

Мікроскопічні симулятори описують рух окремих транспортних засобів, тому вони краще підходять для аналізу черг, зупинок, проходження стоп-лінії та впливу світлофорних фаз. Мезоскопічні й макроскопічні підходи зручніші для великих мереж, однак вони часто узагальнюють поведінку потоку настільки, що частина локальних ефектів світлофорного керування втрачається. Для цієї роботи пріоритет має саме мікроскопічне або гібридне моделювання, оскільки агент світлофорного керування повинен реагувати на стан окремих підходів, смуг і черг.

Серед відомих комерційних інструментів значне поширення має PTV Vissim. Це середовище орієнтоване на мікроскопічне й мультимодальне моделювання транспортних систем, підтримує аналіз взаємодії різних учасників руху, сценарне оцінювання та візуалізацію результатів [9]. Його перевагою є розвинений інструментарій для інженерного моделювання, зручна графічна оболонка, можливість калібрування поведінкових параметрів і використання в професійних транспортних проєктах. Для задачі світлофорного регулювання це означає, що Vissim може бути корисним там, де потрібна висока деталізація дорожньої інфраструктури, наочне представлення сценаріїв та підготовка матеріалів для практичного планування. Водночас для дослідження з комп’ютерних наук важливими є не лише точність моделювання й візуальна якість, а й доступність середовища, можливість автоматизованого запуску великої кількості експериментів, інтеграція з Python та відтворюваність програмного контуру. Комерційна ліцензійна модель і залежність від спеціалізованих інструментів налаштування можуть ускладнювати використання Vissim як основної платформи для відкритого експериментального дослідження алгоритмів навчання.

Подібну роль у професійному транспортному моделюванні відіграє Aimsun. Це середовище використовується для аналізу транспортних мереж різного масштабу й підтримує поєднання різних рівнів деталізації моделювання. Його доцільно розглядати як інструмент для складних

міських або регіональних сценаріїв, де важливими є маршрутний вибір, взаємодія локальних ділянок із ширшою мережею та порівняння альтернатив організації руху. Проте для побудови мультиагентної системи навчання світлофорного регулювання ключовим критерієм є не максимальна кількість інженерних функцій, а можливість безпосередньо зв’язати симулятор із програмним агентом, який на кожному кроці отримує стан середовища й повертає керуючу дію.

Окрему групу становлять агентно-орієнтовані транспортні платформи, зокрема MATSim. Це відкрите середовище розроблене для моделювання транспортної поведінки великої кількості синтетичних агентів, які виконують добові плани переміщень, обирають маршрути й адаптують свою поведінку в процесі ітераційного моделювання [10]. Для стратегічного аналізу мобільності MATSim має суттєві переваги, оскільки дозволяє описувати не лише рух транспортних засобів у мережі, а й поведінкову логіку учасників транспортної системи. У контексті цієї дипломної роботи така агентна природа є змістовно близькою до теми мультиагентних систем. Однак основна задача дослідження пов’язана не з довгостроковим плануванням пересувань населення, а з оперативним навчанням світлофорного керування на рівні перехресть. Через це MATSim є важливим для розуміння агентного підходу в транспортному моделюванні, але не є найзручнішим базовим середовищем для покрокового керування фазами світлофорів у навчальному циклі.

Для подальшої реалізації більш придатним є SUMO, тобто Simulation of Urban MObility. Це відкрите мікроскопічне симуляційне середовище, призначене для моделювання руху в міських транспортних мережах, підтримки різних типів транспортних засобів, маршрутів, детекторів, світлофорних програм і сценаріїв попиту. У наукових публікаціях SUMO розглядається як інструмент, придатний для досліджень у сфері транспортного моделювання, оптимізації та поєднання симуляції із зовнішніми алгоритмічними компонентами [8]. Для цієї роботи особливо

важливо, що SUMO є відкритим програмним середовищем, тому його можна використовувати без залежності від комерційної ліцензії, налаштовувати сценарії в текстових конфігураційних файлах і відтворювати експерименти на різних етапах розробки.

Перевага SUMO полягає не лише у відкритості, а й у відповідності структурі задачі світлофорного керування. У середовищі можна задавати дорожню мережу, вузли, смуги, маршрути, потоки транспортних засобів, світлофорні фази та детектори. Це дозволяє сформувати експериментальний сценарій, у якому кожне регульоване перехрестя може розглядатися як окремий агент. Стан агента може бути побудований на основі даних про черги, кількість транспортних засобів на підходах, час очікування або поточну фазу, а дія агента може відповідати вибору наступної фази чи зміні тривалості зеленого сигналу в межах заданих обмежень. Такий зв’язок між транспортною постановкою й програмною реалізацією є суттєвим: симулятор не повинен нав’язувати лише фіксовану логіку керування, а має допускати зовнішнє прийняття рішень.

Ключовою властивістю SUMO для дослідження алгоритмів навчання є підтримка взаємодії з Python через TraCI. Завдяки цьому симуляція може працювати в покроковому режимі: зовнішня програма отримує поточні показники транспортного середовища, обчислює дію агента, передає її до симулятора й переходить до наступного кроку. Для підкріплювального навчання така схема є природною, оскільки вона відповідає циклу взаємодії «стан – дія – наслідок – оцінка». На практичному рівні це дає змогу реалізувати навчальний контур без створення власного транспортного симулятора з нуля. Алгоритмічна частина може бути написана мовою Python із використанням бібліотек машинного навчання, тоді як SUMO відповідає за відтворення транспортної динаміки. У результаті межа між моделлю керування та середовищем стає достатньо чіткою: агент не керує абстрактними змінними, а взаємодіє з формалізованою транспортною мережею, де його рішення мають спостережувані наслідки.

Водночас використання SUMO не усуває всіх методичних обмежень. Будь-яке симуляційне середовище є наближенням до реального руху, тому результати навчання залежать від того, наскільки коректно задано мережу, потоки, маршрути, поведінкові параметри водіїв і світлофорні обмеження. Якщо сценарій надмірно спрощений, алгоритм може навчитися ефективно діяти лише в штучно зручних умовах. Якщо ж сценарій одразу є занадто складним, інтерпретація результатів і пошук причин помилок у поведінці агентів стають ускладненими. Через це доцільним є поступовий підхід: спочатку перевіряти алгоритм на простішій мережі з обмеженою кількістю перехресть, а потім переходити до складніших конфігурацій із кількома взаємозалежними вузлами, змінним попитом і різними режимами фазового керування.

Окремої уваги потребує питання відтворюваності експериментів. Для наукового дослідження важливо, щоб однакові вхідні умови давали можливість повторно отримати зіставні результати, а зміна параметрів була контрольованою. SUMO добре узгоджується з цією вимогою, оскільки сценарії можуть зберігатися у вигляді конфігураційних файлів, а запуск симуляцій може автоматизуватися програмно. Це дає змогу порівнювати фіксоване керування, адаптивні правила та навчений агентний підхід у межах однієї мережі й однакових потоків. Для дипломної роботи така властивість є важливою не лише технічно, а й методично: висновки про ефективність мультиагентного керування повинні спиратися на контрольоване порівняння, а не на окремий візуально переконливий сценарій.

Порівняння розглянутих середовищ свідчить, що кожне з них має власну сферу доцільного використання. PTV Vissim і Aimsun є сильними професійними платформами для інженерного аналізу, підготовки транспортних проєктів і детального сценарного моделювання. MATSim краще відповідає задачам великомасштабного агентного моделювання мобільності та поведінки учасників транспортної системи. SUMO,

натомість, поєднує достатню деталізацію мікроскопічного моделювання, відкритість, підтримку світлофорного керування, можливість автоматизації та зручну інтеграцію з Python. Саме ця комбінація властивостей є найбільш релевантною для дослідження методів розробки мультиагентної системи навчання світлофорного регулювання.

1.7 Формалізація задачі дослідження

Проведений аналіз предметної галузі показує, що задача оптимізації світлофорного регулювання не може бути коректно описана лише через налаштування тривалості окремих фаз на ізольованому перехресті. Міська транспортна мережа є динамічною системою, у якій транспортні потоки переміщуються між суміжними вузлами, черги накопичуються в часі, а наслідки одного керуючого рішення можуть проявлятися на іншій ділянці мережі. Тому на завершальному етапі аналітичного розділу доцільно перейти від опису проблеми до формалізованого подання основних елементів задачі, які надалі можуть бути використані для побудови математичної моделі, алгоритму навчання та експериментального середовища.

У межах цього дослідження транспортна мережа розглядається як сукупність дорожніх ділянок, смуг руху, регульованих перехресть і напрямків переміщення транспортних засобів між ними. Такий опис узгоджується з графовим баченням міської інфраструктури, де перехрестя виконують роль вузлів прийняття керуючих рішень, а дорожні сегменти між ними виступають каналами передавання транспортного потоку. Для задачі світлофорного регулювання важливими є не всі елементи дорожньої мережі, а насамперед ті, що впливають на утворення черг і пропуск транспортних засобів через стоп-лінію. До них належать вхідні підходи до перехресть, смуги з різними дозволеними маневрами, вихідні ділянки, можливі конфліктні напрямки, структура фаз і часові обмеження сигналів. У такій

постановці транспортна мережа не є лише геометричною картою. Вона задає просторові межі взаємодії агентів, визначає, які перехрестя є сусідніми, які потоки можуть впливати один на одного та де локальна черга може перерости в мережеве блокування.

Множина регульованих перехресть у задачі відповідає тим вузлам мережі, на яких світлофорний контролер може змінювати режим обслуговування транспортних напрямків. Кожне таке перехрестя має власну фазову структуру, набір підходів, поточний стан черг і локальні обмеження. Нерегульовані вузли, виїзди з мережі або допоміжні дорожні елементи можуть бути присутніми в симуляційному сценарії, однак вони не розглядаються як активні суб’єкти керування. Це розмежування потрібне для того, щоб надалі чітко визначити, де система приймає рішення, а де транспортна динаміка лише змінюється під впливом уже прийнятих рішень.

Агентом у формалізованій постановці доцільно вважати світлофорний контролер окремого регульованого перехрестя. Такий агент сприймає доступну інформацію про стан локального транспортного середовища, обирає керуючу дію та впливає на подальший рух транспортних засобів через зміну або збереження світлофорної фази. Змістовно це відповідає логіці мультиагентного керування, розглянутій у попередньому підрозділі: кожне перехрестя має певну автономність, але його рішення не є ізольованим від інших учасників системи. Якщо агент збільшує пропуск одного напрямку, він не лише зменшує власну чергу, а й формує потік, який через певний час досягне сусіднього перехрестя. Через це агент у цій задачі повинен розглядатися не як незалежний оптимізатор власного підходу, а як елемент взаємозалежної керуючої системи. Саме така інтерпретація створює основу для подальшого використання методів мультиагентного підкріплювального навчання, де рішення окремого агента оцінюється з урахуванням локальних і мережевих наслідків [7].

Стан агента має відображати ті характеристики транспортної ситуації, які є суттєвими для вибору світлофорної дії. До таких характеристик

належать поточна фаза світлофора, кількість транспортних засобів на вхідних смугах, довжина черг, зайнятість смуг, середня швидкість руху, час очікування транспортних засобів, а також ознаки залишкової черги після попереднього циклу. У програмному контурі, пов’язаному з використанням SUMO та середовища для навчання агентів, ці показники можуть бути отримані через детектори, статистику смуг або внутрішні структури симулятора. Водночас у першому розділі важливо не деталізувати програмний спосіб зчитування кожної змінної, а визначити смисловий склад стану. Агент повинен мати достатньо інформації для розрізнення ситуацій, у яких одна й та сама фаза є ефективною або, навпаки, створює додаткові втрати. Наприклад, зелений сигнал для магістрального напрямку є доцільним за наявності значної черги й вільної вихідної ділянки, але може бути менш виправданим, якщо нижче за потоком уже сформовано блокування.

Окремо слід враховувати, що стан у реальній або симуляційній транспортній системі не завжди є повним. Агент може спостерігати власні підходи, але не мати повної інформації про віддалені ділянки мережі, причини зміни транспортного попиту або майбутні прибуття транспортних засобів. Тому задача повинна формалізуватися з урахуванням часткової спостережуваності. Це означає, що рішення агента ґрунтується на доступному образі ситуації, а не на ідеальному знанні всього транспортного середовища. Для міського світлофорного керування така особливість є принциповою: навіть у симуляторі можна отримати багато змінних, однак практично значуща модель має бути сумісною з такими даними, які потенційно можуть бути зібрані детекторами, камерами або іншими сенсорними засобами.

Множина можливих дій агента визначається фазовою структурою відповідного світлофорного об’єкта. У загальному випадку дія може означати вибір наступної дозволеної фази, збереження поточної фази на черговому кроці керування або перехід до наступної фази за встановленою

послідовністю. Для подальшої реалізації важливо, що дія не повинна порушувати логіку сумісності транспортних напрямків. Агент не створює нові фази довільно, а працює з наперед визначеним набором безпечних фаз, у яких відкритими є тільки такі потоки, що не мають неприпустимих конфліктів. Такий підхід узгоджується з практикою світлофорного регулювання, де фазова структура, перехідні інтервали та мінімальні тривалості сигналів задаються з урахуванням безпеки руху [1].

Дії агентів мають також часові обмеження. Світлофорний сигнал не може перемикатися надто часто, оскільки це збільшує втрати на перехідні інтервали, погіршує передбачуваність для водіїв і може знижувати фактичну пропускну здатність перехрестя. З іншого боку, надмірно тривале утримання однієї фази може призвести до непропорційного очікування на конфліктних напрямках. Тому простір дій повинен бути не лише дискретним набором фаз, а й керованою множиною допустимих рішень з урахуванням мінімального та максимального часу обслуговування, жовтого сигналу, захисних інтервалів і правил переходу між фазами. У цьому полягає відмінність транспортної задачі від абстрактної задачі вибору дії: алгоритм навчання може пропонувати рішення, але остаточна дія має залишатися в межах інженерно допустимого режиму світлофорного об’єкта.

Цільові показники задачі повинні відображати якість руху не тільки на одному перехресті, а й у межах усієї досліджуваної мережі. До основних показників належать середня затримка транспортних засобів, сумарна довжина черг, кількість зупинок, час очікування, пропуск транспортних засобів через регульовані перехрестя та ознаки виникнення блокування. Для мультиагентної системи особливо важливим є співвідношення локальних і глобальних критеріїв. Якщо агент оцінюється лише за власною чергою, він може приймати рішення, які швидко очищують його підхід, але передають перевантаження до сусіднього вузла. Якщо ж оцінювання є тільки глобальним, окремий агент може втратити чіткий зв’язок між власною дією та результатом. Тому подальша модель має передбачати можливість

поєднання локальних показників із мережевими, зокрема через урахування стану сусідніх перехресть або спільних характеристик транспортного потоку.

Для цієї роботи доцільно розглядати оптимізацію як пошук такої стратегії керування, за якої світлофорні агенти зменшують накопичення черг і затримок, не порушуючи обмежень безпеки та фазової логіки. У цьому формулюванні оптимальність не означає абсолютне усунення заторів, оскільки транспортний попит може перевищувати пропускну здатність мережі незалежно від якості алгоритму. Більш коректним є прагнення до раціонального розподілу обмеженого зеленого часу між конфліктними напрямками, стабілізації черг, запобігання їх поширенню на суміжні ділянки та підвищення узгодженості роботи кількох світлофорних об’єктів. Така постановка є реалістичнішою для міських умов, де система керування не створює додаткову дорожню місткість, а лише впливає на те, як доступна пропускна здатність використовується в часі.

Обмеження задачі можна поділити на транспортні, інформаційні та алгоритмічні. Транспортні обмеження пов’язані з геометрією мережі, кількістю смуг, дозволеними маневрами, пропускною здатністю підходів, пішохідними фазами, мінімальними й максимальними тривалостями сигналів та неможливістю одночасного обслуговування конфліктних потоків. Інформаційні обмеження зумовлені неповнотою спостережень, затримками отримання даних, можливою похибкою сенсорів і тим, що агент зазвичай бачить лише локальну частину мережі. Алгоритмічні обмеження виникають через необхідність приймати рішення в покроковому режимі, зберігати стабільність навчання та уникати хаотичних перемикань. Наявність цих обмежень означає, що подальша система не може оцінюватися лише за підсумковим зменшенням середньої затримки. Вона також має демонструвати коректну поведінку в межах допустимих керуючих дій.

Симуляційне середовище в цій постановці виконує роль контрольованого транспортного середовища, у якому агенти взаємодіють із мережею без ризику для реальної інфраструктури. З урахуванням попереднього аналізу доцільним базовим інструментом для такої постановки є SUMO, оскільки воно підтримує мікроскопічне моделювання руху, світлофорні програми, маршрути, детектори та зовнішнє керування через програмний інтерфейс [8]. Для формалізації задачі це означає, що транспортна мережа, попит, фазові програми й показники якості можуть бути задані у відтворюваному вигляді, а агенти можуть отримувати стан середовища та передавати керуючі дії на кожному кроці симуляції. Наявність у досліджуваному програмному проєкті сценаріїв SUMO та компонентів для опису станів, дій, винагород і метрик підтверджує практичну придатність такого подання для подальших розділів, однак у межах першого розділу ці елементи розглядаються саме як основа постановки, а не як детальний опис реалізації.

1.8 Постановка задачі та завдання роботи

Проведений аналіз предметної галузі показав, що задача оптимізації світлофорного регулювання міських транспортних потоків не може бути зведена лише до добору сталої тривалості фаз або локального зменшення черги на одному підході. Регульоване перехрестя функціонує як частина взаємопов’язаної транспортної мережі, де кожне керуюче рішення впливає на затримки, пропуск транспортних засобів, накопичення черг і стан суміжних вузлів. Через це постановка задачі роботи має поєднувати транспортно-інженерні обмеження світлофорного керування з методами штучного інтелекту, здатними враховувати динамічний характер міського попиту [1].

Метою кваліфікаційної роботи є дослідження методів розробки мультиагентної системи навчання світлофорного регулювання для

оптимізації міських транспортних потоків. У межах цієї мети світлофорні об’єкти розглядаються не як незалежні регулятори, а як сукупність агентів, що діють у спільному транспортному середовищі. Кожен агент повинен отримувати інформацію про поточний стан відповідного перехрестя, обирати допустиму керуючу дію та впливати на подальший розподіл потоків у мережі. Така постановка відповідає логіці мультиагентного підкріплювального навчання, у якому якість рішення оцінюється не тільки за миттєвим результатом, а й за наслідками для подальшого стану середовища [7]. Об’єктом дослідження є процес світлофорного керування міськими транспортними потоками, а предметом дослідження є методи побудови, навчання та оцінювання мультиагентної системи керування світлофорними фазами в симуляційному середовищі.

З урахуванням сформульованої мети дослідницька задача полягає в тому, щоб визначити, яким чином методи підкріплювального навчання можуть бути використані для адаптивного керування групою світлофорних об’єктів у міській транспортній мережі. Практична задача полягає в побудові відтворюваного програмно-експериментального контуру, у якому можна моделювати транспортні сценарії, навчати агентів, порівнювати різні підходи до керування та оцінювати їхній вплив на затримки, черги й пропуск транспортних засобів. Такий поділ є важливим, оскільки робота має не лише описати теоретичну придатність мультиагентного підходу, а й перевірити його в умовах контрольованого моделювання.

Для досягнення поставленої мети необхідно послідовно розв’язати кілька взаємопов’язаних завдань. Насамперед потрібно проаналізувати предметну область світлофорного регулювання, визначити основні характеристики міських транспортних потоків, роль фазової структури, причини виникнення черг і затримок, а також обмеження, пов’язані з безпекою та пропускною здатністю перехресть. Далі необхідно дослідити традиційні, адаптивні та інтелектуальні методи керування світлофорними об’єктами, щоб визначити, які їхні властивості можуть бути використані як

основа для побудови навчальної системи, а які обмеження потребують застосування більш гнучких алгоритмів. Окремим завданням є аналіз методів підкріплювального навчання та мультиагентного керування, оскільки саме вони дають змогу представити світлофорний контролер як агента, що приймає рішення в умовах часткової спостережуваності, змінного попиту та відкладених наслідків власних дій. На основі цього аналізу потрібно розробити архітектуру мультиагентної системи, визначити склад спостережень, множину допустимих дій, принцип формування винагороди, спосіб урахування локальних і мережевих показників, а також обмеження, які не дозволяють агентам порушувати логіку безпечного світлофорного регулювання.

Наступним змістовним етапом є реалізація моделі в симуляційному середовищі. З огляду на попередній аналіз, доцільним інструментом для цього є SUMO, оскільки воно підтримує мікроскопічне моделювання руху, світлофорні програми, маршрути, детектори та зовнішнє керування через програмний інтерфейс [8]. Використання наявного програмного проєкту, що містить компоненти для опису транспортних сценаріїв, станів, дій, винагород і метрик, дозволяє перейти від загальної постановки задачі до конкретної експериментальної перевірки. При цьому реалізація не повинна обмежуватися запуском одного сценарію. Вона має забезпечувати можливість відтворюваного навчання агентів, запуску тестових симуляцій, збирання показників якості та порівняння результатів із базовими або традиційними режимами керування.

Експериментальна частина роботи має бути спрямована на оцінювання того, чи здатна запропонована мультиагентна система покращувати параметри руху порівняно з простішими підходами до світлофорного керування. Для цього необхідно визначити набір критеріїв оцінювання, зокрема середню затримку транспортних засобів, довжину черг, час очікування, кількість зупинок, пропуск транспортних засобів і стійкість роботи мережі за різних сценаріїв попиту. Важливо враховувати,

що позитивний результат не повинен трактуватися лише як зменшення одного числового показника. Для транспортної системи суттєвими є також відсутність надмірно частих перемикань, збереження допустимої фазової логіки, рівномірність обслуговування напрямків і здатність системи не переносити локальне перевантаження на сусідні перехрестя.

1.9 Висновки до розділу

У першому розділі проведено аналіз предметної галузі світлофорного регулювання міських транспортних потоків, розглянуто причини виникнення черг і затримок, особливості динамічного транспортного попиту, обмеження традиційних методів керування та можливості застосування інтелектуальних підходів. Показано, що фіксовані, координовані та транспортно-залежні методи мають практичну цінність, однак недостатньо ефективно реагують на змінний попит, залишкові черги та взаємний вплив суміжних перехресть.

Обґрунтовано доцільність використання мультиагентного підходу, у якому кожне регульоване перехрестя розглядається як агент, що спостерігає локальний стан, приймає керуюче рішення та впливає на роботу сусідніх вузлів. Визначено, що перспективним методом для такої задачі є підкріплювальне навчання, оскільки воно дає змогу враховувати не лише поточний стан транспортного потоку, а й відкладені наслідки керуючих дій.

Також обґрунтовано вибір SUMO як базового симуляційного середовища, придатного для мікроскопічного моделювання руху, опису світлофорних фаз, збирання транспортних показників і взаємодії з алгоритмами навчання через Python. У тексті розрахунки кількісного ефекту не наведені, тому конкретні числові значення у висновку відсутні. Основними показниками для подальшого оцінювання системи визначено середню затримку, довжину черг, час очікування, кількість зупинок, пропуск транспортних засобів і стійкість роботи мережі.

2 ТЕОРЕТИЧНЕ ДОСЛІДЖЕННЯ МЕТОДІВ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ РОЗВ'ЯЗАННЯ ЗАДАЧІ СВІТЛОФОРНОГО РЕГУЛЮВАННЯ

2.1 Загальна характеристика задачі навчання керування в транспортних системах

У межах другого розділу задача світлофорного регулювання розглядається не лише як інженерне налаштування фаз, а як процес навчання керуючої політики. Такий перехід є методично важливим, оскільки попередній аналіз показав залежність якості регулювання від поточного стану транспортного середовища, залишкових черг, взаємодії суміжних перехресть і відкладених наслідків прийнятих рішень. Світлофорний контролер у цій постановці діє як агент, який отримує спостереження, обирає допустиму дію, впливає на транспортний потік і надалі оцінює результат через зміну показників руху.

Загальна логіка навчання керування в транспортній системі відповідає задачам послідовного прийняття рішень. На кожному кроці часу система перебуває в певному стані, що відображає завантаженість підходів, активну фазу, довжини черг, час очікування транспортних засобів і, за потреби, інформацію про сусідні перехрестя. Після цього агент обирає керуючу дію: зберегти поточну фазу, перейти до іншої допустимої фази або змінити параметр обслуговування в межах заданих обмежень. Дія змінює умови руху, оскільки одні транспортні потоки отримують право проїзду, а інші продовжують очікувати. У результаті середовище переходить до нового стану, а агент отримує числову оцінку якості рішення, наприклад через зменшення або збільшення черг, затримок, кількості зупинок чи ознак блокування. Саме повторюваність такого циклу створює підґрунтя для використання підкріплювального навчання, де політика формується на основі досвіду взаємодії агента із середовищем [11].

Формально задачу можна подати як марковський процес прийняття рішень, у якому поточний стан містить достатню інформацію для вибору подальшої дії. Для світлофорного керування це припущення має практичні обмеження, оскільки реальний агент не завжди спостерігає всю транспортну мережу. Проте в симуляційному середовищі можна визначити контрольований набір спостережень, достатній для навчання та порівняння методів.

$$M = \langle S, A, P, R, \gamma \rangle, \tag{2.1}$$

де M – модель задачі послідовного прийняття рішень;

S – множина можливих станів транспортного середовища;

A – множина допустимих керуючих дій агента;

P – функція переходу між станами після виконання дії;

R – функція винагороди або штрафу;

γ – коефіцієнт дисконтування майбутніх результатів.

У транспортній інтерпретації множина S не повинна зводитися до загальної кількості автомобілів на перехресті. Для задачі світлофорного регулювання важливим є просторовий розподіл транспортних засобів за підходами та смугами, оскільки однакова кількість автомобілів може відповідати різним керуючим ситуаціям. Наприклад, десять транспортних засобів, рівномірно розподілених між підходами, і десять транспортних засобів, накопичених на одному перевантаженому напрямку, потребують різної реакції світлофорного агента. Доцільно також враховувати поточну фазу, тривалість її дії, залишкову чергу після попереднього зеленого сигналу та показники наближення транспортних засобів до стоп-лінії. Якщо задача розглядається в мультиагентній постановці, стан окремого агента може доповнюватися агрегованими характеристиками сусідніх вузлів, але такий обмін має бути обмеженим і змістовно виправданим. Надмірне розширення стану збільшує розмірність задачі, ускладнює навчання та може

призвести до залежності політики від даних, які в реальному середовищі важко отримати без затримок або похибок.

Множина A визначається не довільними математичними діями, а фазовою структурою світлофорного об’єкта. Це принципова відмінність транспортної задачі від абстрактного прикладу навчання агента. Світлофорний агент не може обирати будь-яку комбінацію сигналів, оскільки частина напрямків має конфліктні траєкторії, а перемикання фаз потребує перехідних інтервалів. Тому дія повинна бути допустимою з позиції безпеки руху, мінімальної тривалості сигналів і логіки сумісності потоків. У найпростішому випадку дія відповідає вибору однієї з наперед визначених фаз. У складнішому варіанті дія може означати продовження поточної фази на один крок керування або дозвіл переходу до наступної фази за заданою послідовністю (рисунок 2.1).

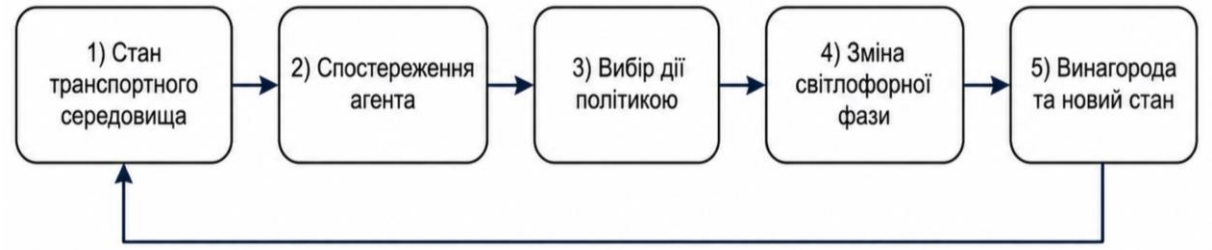


Рисунок 2.1 – Схема циклу взаємодії світлофорного агента з транспортним середовищем

Керуюча політика визначає, яку дію агент обирає в конкретному стані. У детермінованому випадку політика прямо зіставляє стану одну дію, а в стохастичному випадку задає ймовірності вибору різних дій. Для світлофорного керування стохастична політика є корисною на етапі навчання, оскільки дозволяє агенту досліджувати різні режими роботи й не закріплюватися передчасно на локально зручній, але неефективній стратегії. Водночас під час експлуатації або тестування бажаною є стабільніша

поведінка, у якій агент не перемикає фази хаотично й не створює непередбачуваних умов для транспортного потоку.

Політику можна подати так:

$$\pi(a \mid s) = P(A_t = a \mid S_t = s), \tag{2.2}$$

де $\pi(a \mid s)$ – політика вибору дії;

s – поточний стан середовища;

a – керуюча дія;

A_t – дія, обрана на кроці t;

S_t – стан середовища на кроці t;

P – імовірність вибору відповідної дії.

Особливість транспортної задачі полягає в тому, що якість дії не завжди проявляється одразу. Якщо агент відкриває фазу для перевантаженого напрямку, миттєво може зменшитися локальна черга, але через певний час великий транспортний пакет може досягти наступного перехрестя й спричинити перевантаження там. Навпаки, короткочасне утримання другорядного напрямку в режимі очікування може бути виправданим, якщо воно запобігає блокуванню магістрального потоку. Через це у функції оцінювання потрібно враховувати не лише поточну винагороду, а й майбутні наслідки рішень. Для цього використовується дисконтована сума винагород:

$$G_t = \sum_{k=0}^{\infty} \gamma^k \, r_{t+k+1}, \tag{2.3}$$

де G_t – сумарна очікувана оцінка результату після кроку t;

γ – коефіцієнт дисконтування;

r_{t+k+1} – винагорода, отримана після наступних керуючих кроків;

k – індекс майбутнього кроку.

Функція винагороди є одним із найбільш відповідальних елементів постановки задачі. Якщо винагорода формується тільки як штраф за довжину локальної черги, агент може навчитися швидко очищувати власний підхід, але не враховувати наслідки для сусідніх перехресть. Якщо ж винагорода описує лише глобальний стан мережі, окремому агенту складніше встановити причинний зв’язок між власною дією та отриманим результатом. Для задачі мультиагентного світлофорного керування доцільним є поєднання локальних і мережевих складових: локальна частина відображає черги, затримки та час очікування на конкретному перехресті, а мережевий компонент враховує пропуск транспортних засобів, поширення черг і можливе блокування суміжних ділянок. У літературі з глибокого підкріплювального навчання для інтелектуальних транспортних систем підкреслюється, що вибір станів, дій і винагороди безпосередньо впливає на здатність агента навчатися стабільній та узгодженій поведінці [7].

У таблиці 2.1 наведено відповідність компонентів задачі навчання елементам світлофорного керування.

Таблиця 2.1 – Відповідність компонентів задачі навчання елементам світлофорного керування

Компонент задачі навчання	Інтерпретація для світлофорного регулювання
Агент	Світлофорний контролер окремого регульованого перехрестя
Середовище	Транспортна мережа або її симуляційна модель
Стан	Черги, затримки, активна фаза, час очікування, локальні та сусідні показники руху
Дія	Вибір фази, продовження фази або перехід до наступної допустимої фази
Винагорода	Оцінка зміни затримок, черг, пропуску транспортних засобів і стійкості руху
Політика	Правило або модель, що визначає вибір дії за поточним станом
Обмеження	Безпечна фазова структура, мінімальні інтервали, сумісність потоків, часткова спостережуваність

Практична реалізація такої постановки потребує контрольованого середовища, у якому можна багаторазово повторювати сценарії, змінювати попит і безпечно перевіряти поведінку агентів. Реальна транспортна мережа не підходить для первинного навчання, оскільки помилки політики можуть спричинити затримки, небезпечні ситуації або нестабільність руху. Тому симуляційне моделювання виконує роль проміжного рівня між теоретичною постановкою та практичною перевіркою.

У межах цієї роботи таким середовищем доцільно розглядати SUMO, оскільки воно підтримує мікроскопічне моделювання транспортних засобів, світлофорні програми, маршрути, детектори та зовнішнє керування через програмний інтерфейс [8]. Це дає змогу організувати цикл навчання, у якому агент отримує спостереження зі симуляції, передає керуючу дію, після чого середовище оновлює стан транспортного потоку та повертає показники для розрахунку винагороди.

Водночас подання світлофорного регулювання як задачі навчання не усуває інженерних обмежень предметної області. Агент може оптимізувати вибір дій лише в межах допустимого простору, сформованого фазовою схемою, мінімальними й максимальними тривалостями сигналів, перехідними інтервалами та правилами безпеки. Це означає, що навчальна система не повинна розглядатися як повністю автономний генератор сигналів. Її коректніше трактувати як механізм вибору найдоцільнішої дії серед заздалегідь дозволених варіантів.

Такий підхід зменшує ризик некоректної поведінки, полегшує інтерпретацію результатів і забезпечує зв’язок між методами штучного інтелекту та реальною логікою світлофорного керування.

2.2 Загальна характеристика задачі навчання керування в транспортних системах

З урахуванням попередньої формалізації задачі світлофорного регулювання як процесу послідовного прийняття рішень доцільно

детальніше розглянути базові поняття підкріплювального навчання, на яких надалі ґрунтується опис DQN, PPO та мультиагентної моделі керування. У цій постановці навчання не зводиться до прямого розрахунку сталої тривалості фаз. Воно передбачає, що світлофорний контролер поступово формує правило поведінки на основі багаторазової взаємодії з транспортним середовищем, отримуючи числову оцінку наслідків власних дій [11].

Ключовим елементом такої постановки є агент. У задачі світлофорного регулювання агентом виступає світлофорний контролер окремого регульованого перехрестя або логічний модуль, який приймає рішення щодо вибору фази. Його поведінка не є довільною: агент діє в межах заданої фазової структури, не може відкривати конфліктні напрямки руху та повинен ураховувати мінімальні тривалості сигналів і перехідні інтервали. Середовищем для агента є транспортна мережа або її симуляційна модель, у якій транспортні засоби прибувають до перехресть, утворюють черги, проходять стоп-лінії та впливають на стан суміжних ділянок. На відміну від статичного інженерного розрахунку, у підкріплювальному навчанні середовище змінюється після кожної дії агента, тому якість рішення оцінюється не ізольовано, а через подальшу динаміку транспортного потоку.

Стан середовища описує інформацію, доступну агенту в момент прийняття рішення. Для світлофорного керування стан може містити довжини черг на підходах, поточну фазу, тривалість її дії, час очікування транспортних засобів, кількість автомобілів поблизу стоп-лінії, показники завантаженості смуг і, за потреби, агреговані характеристики сусідніх перехресть. Важливо, що стан не повинен бути надмірно широким. Якщо до нього включити всі змінні симулятора, модель може стати складною для навчання та менш придатною для перенесення в реальні умови. Тому стан має відображати ті характеристики, які справді впливають на вибір керуючої дії.

Дія агента відповідає керуючому впливу на світлофорний об’єкт. У дискретній постановці дія може означати вибір однієї з допустимих фаз, продовження поточної фази або перехід до наступної фази за наперед визначеною послідовністю. Такий підхід узгоджується з практикою світлофорного регулювання, оскільки безпека руху задає жорсткі обмеження на простір можливих рішень [1]. Для методів DQN і PPO це має різне значення. DQN зручний для дискретного простору дій, де кожній фазі або команді відповідає окреме значення якості. PPO може працювати як із дискретними, так і з параметризованими діями, що робить його придатним для складніших варіантів керування, наприклад для вибору фази разом із тривалістю її продовження.

Взаємодію агента із середовищем можна подати як послідовність переходів:

$$S_t \overset{A_t}{\rightarrow} S_{t+1}, \quad r_{t+1} = R(S_t, A_t, S_{t+1}), \tag{2.4}$$

де S_t – стан транспортного середовища на кроці t;

A_t – дія світлофорного агента;

S_{t+1} – новий стан після виконання дії;

r_{t+1} – винагорода, отримана після переходу;

R – функція оцінювання результату дії.

Винагорода є числовим показником, який визначає, наскільки корисною була дія агента. У транспортній задачі вона зазвичай формується не як позитивна оцінка самого перемикання фази, а як відображення зміни якості руху. Наприклад, агент може отримувати штраф за збільшення сумарної черги, часу очікування або кількості зупинок. Водночас занадто проста функція винагороди може спричинити небажану поведінку. Якщо оцінювати лише локальну чергу, агент може надмірно обслуговувати один напрямок і погіршувати стан сусідніх вузлів. Якщо використовувати тільки глобальний показник мережі, окремому агенту складніше встановити

зв’язок між власною дією та отриманим результатом. Тому для мультиагентного світлофорного керування доцільно поєднувати локальні та мережеві складові винагороди [7].

Один із можливих узагальнених варіантів функції винагороди для світлофорного агента можна записати так:

$$r_t = -(\alpha Q_t + \beta W_t + \delta C_t), \tag{2.5}$$

де r_t – винагорода агента на кроці t;

Q_t – показник сумарної або середньої довжини черги;

W_t – показник часу очікування транспортних засобів;

C_t – штраф за небажані умови, наприклад блокування або надмірно часте перемикання;

α, β, δ – вагові коефіцієнти відповідних складових.

Політика визначає спосіб вибору дії за поточним станом. У світлофорному керуванні політика фактично відповідає правилу, за яким контролер вирішує, яку фазу залишити активною або яку фазу обрати наступною. На етапі навчання політика повинна поєднувати використання вже знайдених ефективних рішень із дослідженням нових варіантів поведінки. Це особливо важливо для транспортних систем, де короткостроково приваблива дія може мати відкладені негативні наслідки. Наприклад, швидке очищення одного підходу може передати великий транспортний потік до сусіднього перехрестя, яке ще не готове його прийняти. Тому політика має оцінювати дію не лише за миттєвим ефектом, а й за її внеском у подальший стан мережі (рисунок 2.2).

Епізодом у задачі навчання є завершена послідовність взаємодій агента із середовищем. У симуляційному моделюванні епізод може відповідати одному запуску транспортного сценарію протягом заданого інтервалу часу, наприклад моделюванню ранкового пікового періоду або умовного фрагмента міського руху. Після завершення епізоду система може

обчислити накопичену винагороду, середню затримку, довжину черг та інші показники якості. Епізодична структура важлива тому, що агент навчається не за одним рішенням, а за послідовністю рішень, у якій кожна дія впливає на майбутні умови. Саме ця властивість відрізняє підкріплювальне навчання від звичайної класифікації або регресії.

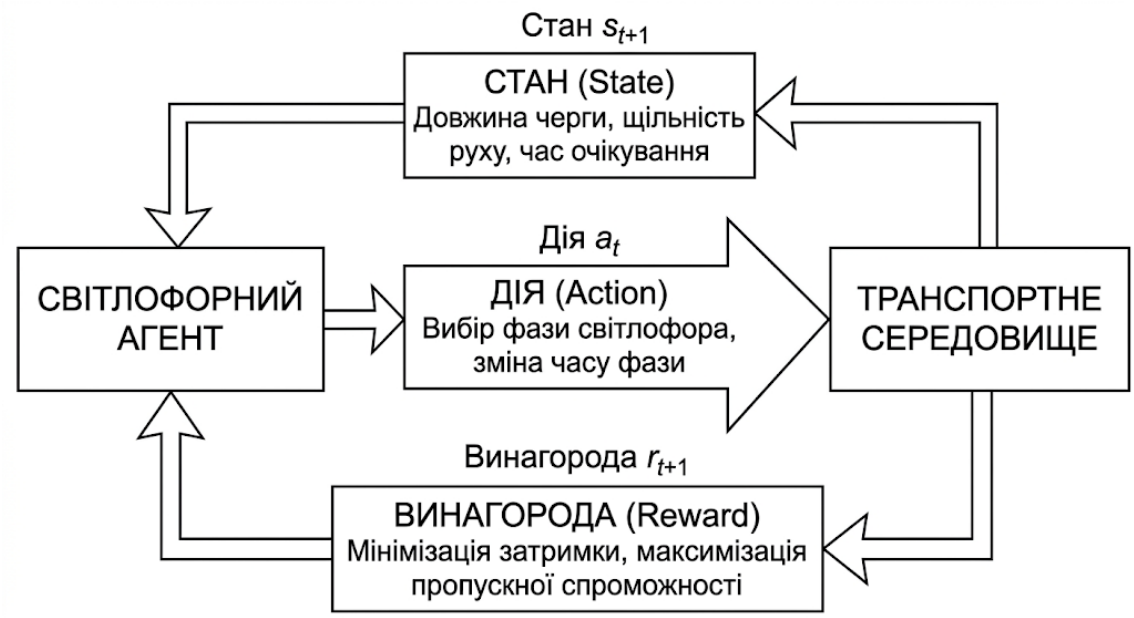


Рисунок 2.2 – Узагальнена схема підкріплювального навчання світлофорного контролера

Накопичена винагорода характеризує загальну корисність поведінки агента впродовж певного горизонту часу. Для транспортної системи це означає, що оцінюється не тільки миттєве скорочення черги після окремого перемикання, а й здатність політики підтримувати прийнятний стан руху протягом усього епізоду. З урахуванням коефіцієнта дисконтування майбутні винагороди можуть мати меншу вагу, ніж ближчі за часом результати. Це дає змогу регулювати баланс між короткостроковим реагуванням і довгостроковою стабільністю керування ().

$$J(\pi) = \mathbb{E}_{\pi}[\sum_{t=0}^{T} \gamma^t r_t], \tag{2.6}$$

де $J(\pi)$ – очікувана якість політики π;

$\mathbb{E}_\pi$ – математичне сподівання за траєкторіями, сформованими політикою;

T – тривалість епізоду;

γ – коефіцієнт дисконтування;

r_t – винагорода на кроці t.

У таблиці 2.2 наведено базові поняття підкріплювального навчання в задачі світлофорного регулювання. У межах подальшого дослідження ці поняття виконують роль спільної методичної основи для різних алгоритмів навчання. DQN використовує оцінювання корисності дій у заданих станах і тому природно відповідає задачі вибору однієї з дискретних світлофорних фаз. PPO розглядає безпосереднє вдосконалення політики, що є корисним для стабілізації навчання та обмеження надмірно різких змін поведінки агента.

Таблиця 2.2 – Базові поняття підкріплювального навчання в задачі світлофорного регулювання

Поняття	Зміст у підкріплювальному навчанні	Інтерпретація для світлофорного керування
Агент	Суб’єкт, що приймає рішення	Світлофорний контролер перехрестя
Середовище	Система, з якою взаємодіє агент	Транспортна мережа або симуляція SUMO
Стан	Опис ситуації на поточному кроці	Черги, затримки, активна фаза, завантаженість підходів
Дія	Керуюче рішення агента	Вибір, продовження або зміна світлофорної фази
Винагорода	Числова оцінка наслідку дії	Штраф або виграш за зміну черг, затримок і пропуску
Політика	Правило вибору дії	Стратегія керування фазами за поточним станом
Епізод	Завершена послідовність взаємодій	Один запуск транспортного сценарію
Накопичена винагорода	Узагальнена оцінка поведінки	Якість керування протягом усього сценарію

У мультиагентній постановці наведені поняття зберігаються, але кожен агент взаємодіє не лише із транспортним середовищем, а й опосередковано з іншими агентами через зміну потоків між перехрестями. Саме тому стан, дія та винагорода мають визначатися так, щоб відображати як локальну ефективність окремого контролера, так і мережеві наслідки його рішень.

2.3 Математична постановка задачі як марковського процесу прийняття рішень

Формалізація задачі світлофорного регулювання як марковського процесу прийняття рішень є необхідним методичним етапом перед описом конкретних алгоритмів навчання. У попередніх підрозділах було визначено загальну логіку взаємодії світлофорного агента з транспортним середовищем, однак для застосування методів підкріплювального навчання цього недостатньо. Потрібно задати математичну структуру, у межах якої стан, дія, перехід, винагорода та довгострокова оцінка поведінки мають однозначну інтерпретацію [11].

У класичній постановці марковський процес прийняття рішень описується впорядкованою сукупністю компонентів, які визначають усі елементи взаємодії агента із середовищем. Для задачі світлофорного керування така сукупність має не абстрактний, а прикладний зміст: множина станів відображає поточну транспортну ситуацію на підходах до перехрестя, множина дій задається допустимими світлофорними фазами, функція переходів описує зміну черг і потоків після керуючого впливу, функція винагороди пов’язує наслідки дії з критеріями якості руху, а коефіцієнт дисконтування визначає значущість майбутніх наслідків. Узагальнено модель було подано в формулі (2.1). Марковська властивість означає, що для вибору наступної дії достатньо поточного стану, а не всієї попередньої історії руху. У реальній транспортній мережі це припущення

виконується лише наближено, оскільки на поточну ситуацію впливають попередні фази, нерівномірність прибуття транспортних засобів, поведінка водіїв і рішення сусідніх світлофорів. Проте для побудови навчальної моделі воно є продуктивним спрощенням. Якщо стан містить довжини черг, активну фазу, час її дії, показники очікування та агреговані характеристики суміжних підходів, то агент отримує достатньо інформації для прийняття керуючого рішення без явного збереження всієї історії симуляції. Саме такий підхід дає змогу перейти від описового аналізу транспортної ситуації до алгоритмічної моделі навчання.

Для окремого перехрестя стан доцільно подавати як вектор ознак, сформований на кожному кроці керування. Його склад залежить від доступних датчиків або даних симуляційного середовища, але в межах досліджуваної задачі принципово важливо відобразити не тільки кількість транспортних засобів, а й просторово-часову структуру навантаження. Наприклад, однакова сумарна кількість автомобілів може відповідати різним ситуаціям: рівномірному розподілу між підходами, критичному накопиченню на одному напрямку або утворенню черги, що поширюється до попереднього перехрестя. Тому стан агента можна подати так:

$$s_t^i = \left[q_t^i, w_t^i, p_t^i, \tau_t^i, n_t^i\right], \tag{2.7}$$

де s_t^i – стан і-го світлофорного агента на кроці t;

q_t^i – вектор довжин черг на підходах до перехрестя;

w_t^i – показники часу очікування транспортних засобів;

p_t^i – поточна світлофорна фаза;

τ_t^i – тривалість дії поточної фази;

n_t^i – агреговані характеристики сусідніх перехресть або суміжних ділянок мережі.

Множина дій у цій задачі обмежується вимогами безпеки та фазовою структурою перехрестя. Агент не повинен формувати довільні комбінації

сигналів, оскільки відкриття конфліктних напрямків може створити небезпечну ситуацію. Через це простір дій визначається не всіма теоретично можливими керуючими впливами, а тільки тими, що відповідають заздалегідь заданим допустимим фазам, мінімальним тривалостям зеленого сигналу та перехідним інтервалам [1]. У дискретній постановці дія може означати вибір однієї з фаз, продовження поточної фази або перехід до наступної фази за встановленою послідовністю. Формально множину дій і-го агента можна записати як:

$$A^i = \{a_1^i, a_2^i, \dots, a_m^i\}, \tag{2.8}$$

де A^i – множина допустимих дій і-го агента;

$a_1^i, a_2^i, \dots, a_m^i$ – окремі керуючі дії, що відповідають дозволеним фазам або режимам зміни фази;

m – кількість допустимих дій для відповідного світлофорного об’єкта.

Функція переходів відображає, як транспортне середовище змінюється після виконання дії. У задачі світлофорного керування вона залежить від поточного стану підходів, обраної фази, інтенсивності прибуття транспортних засобів, пропускної здатності смуг, поведінки водіїв і стану сусідніх вузлів. Аналітично задати таку функцію для міської мережі складно, оскільки рух має стохастичний і просторово розподілений характер. Тому в практичній частині дослідження функція переходів може реалізовуватися через симуляційне середовище, зокрема SUMO, яке оновлює положення транспортних засобів, черги та показники руху після кожного керуючого кроку [8]. У формальному вигляді перехід можна подати так:

$$P(\, s_{t+1} \mid s_t, a_t\,) = Pr(\, S_{t+1} = s_{t+1} \mid S_t = s_t, A_t = a_t\,), \tag{2.9}$$

де $P(s_{t+1} \mid s_t, a_t)$ – імовірність переходу до нового стану після виконання дії;

s_t – поточний стан;

a_t – дія агента;

S_{t+1} – наступний стан середовища;

Pr – імовірнісна оцінка відповідної події.

Функція винагороди має безпосередньо пов'язувати математичну модель із метою оптимізації міських транспортних потоків. Якщо винагорода задається занадто вузько, наприклад лише через довжину черги на одному підході, агент може сформувати локально вигідну, але мережево неузгоджену поведінку. Якщо ж винагорода повністю глобальна, окремому світлофорному агенту складніше визначити, яка саме його дія спричинила поліпшення або погіршення результату. Тому для мультиагентної системи доцільно використовувати комбіновану винагороду, яка поєднує локальний штраф за черги та очікування з додатковою складовою, що відображає вплив на суміжні ділянки. Один із можливих варіантів запису має такий вигляд:

$$r_t^i = -\left(\alpha Q_t^i + \beta W_t^i + \delta B_t^i\right) + \eta F_t^i, \tag{2.10}$$

де r_t^i – винагорода і-го агента на кроці t;

Q_t^i – показник черги на підходах до перехрестя;

W_t^i – сумарний або середній час очікування;

B_t^i – штраф за блокування, надмірне накопичення або небажане перемикання;

F_t^i – показник пропуску транспортних засобів;

$\alpha, \beta, \delta, \eta$ – вагові коефіцієнти складових винагороди.

Запропонована форма винагороди не є єдино можливою, але вона відображає логіку дослідження: агент не повинен просто мінімізувати одну

локальну величину, а має враховувати взаємозв'язок між затримкою, чергою, пропуском і стійкістю руху. Вагові коефіцієнти в такій функції виконують роль механізму налаштування пріоритетів. Наприклад, збільшення ваги W_t^i посилює увагу до часу очікування, тоді як збільшення δ робить небажаними режими, що призводять до блокування або нестабільного перемикання фаз. У подальшій експериментальній частині такі коефіцієнти можуть уточнюватися відповідно до сценарію моделювання, однак сама структура винагороди вже задає методичну основу для навчання агентів (рисунок 2.3).

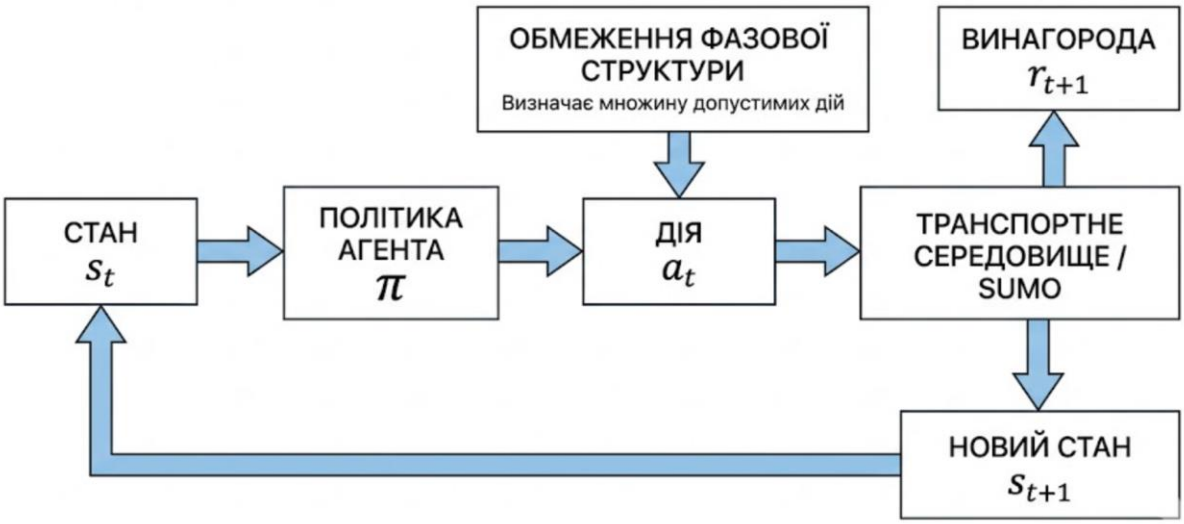


Рисунок 2.3 – Формальна схема MDP-моделі світлофорного регулювання

Коефіцієнт дисконтування γ визначає, наскільки сильно агент враховує майбутні наслідки поточних рішень. Для світлофорного регулювання це має особливе значення, оскільки дія може мати відкладений ефект. Зміна фази зараз впливає не тільки на поточну чергу, а й на транспортний пакет, який через певний час прибуде до сусіднього перехрестя. Якщо γ занадто малий, агент орієнтуватиметься переважно на короткострокове зменшення локального штрафу. Якщо γ наближений до одиниці, модель приділятиме більше уваги довгостроковій стабільності, але

навчання може стати складнішим через більшу залежність від майбутніх станів. У задачах інтелектуального транспортного керування такий компроміс є принциповим, оскільки локально швидке рішення не завжди є корисним для мережі загалом [7].

У мультиагентній постановці кожне регульоване перехрестя можна розглядати як окремого агента $i \in \{1, 2, \ldots, N\}$, де N – кількість світлофорних об'єктів у досліджуваній мережі. Тоді глобальний стан системи формується як сукупність локальних станів, а спільна дія складається з рішень усіх агентів. Така модель краще відповідає природі міської транспортної мережі, ніж повністю ізольоване керування, оскільки потік, випущений одним перехрестям, стає вхідним навантаженням для іншого. Водночас повна централізація може вимагати надмірного обсягу даних і ускладнювати навчання. Тому доцільним є проміжний підхід, за якого агент приймає рішення на основі локального стану, але може враховувати обмежений набір сусідніх показників.

2.4 Методи Q-навчання та Deep Q-Network для керування світлофорами

З урахуванням MDP-постановки задачі світлофорного регулювання наступним методичним кроком є вибір алгоритму, здатного навчати агента приймати дискретні керуючі рішення. Для світлофорного контролера така дискретність є природною: дія може відповідати збереженню поточної фази, переходу до наступної допустимої фази або вибору однієї з наперед визначених фазових комбінацій. Тому методи, що оцінюють якість пар «стан – дія», мають безпосереднє прикладне значення для цієї задачі.

Q-навчання належить до методів підкріплювального навчання, у яких агент не потребує явної аналітичної моделі переходів середовища. Це важливо для транспортної системи, оскільки точний опис зміни черг після перемикання фази залежить від випадкового прибуття транспортних

засобів, поведінки водіїв, пропускної здатності смуг і стану сусідніх перехресть. Замість побудови повної функції переходів агент поступово уточнює оцінку Q(s, a), яка показує очікувану довгострокову корисність виконання дії a у стані s. У класичному табличному варіанті для кожної можливої пари стану й дії зберігається окреме значення. Після виконання дії, отримання винагороди та переходу до нового стану ця оцінка коригується за правилом часової різниці [12]:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_{a'} Q_t(s_{t+1}, a') - Q_t(s_t, a_t) \right], \quad (2.11)$$

де $Q_t(s_t, a_t)$ – поточна оцінка якості виконання дії a_t у стані s_t;

α – швидкість навчання;

r_{t+1} – винагорода після виконання дії;

γ – коефіцієнт дисконтування;

$\max_{a'} Q_t(s_{t+1}, a')$ – найкраща очікувана оцінка дії в наступному стані;

a' – можлива дія в стані s_{t+1}.

Зміст цієї формули для світлофорного керування полягає в тому, що агент порівнює попередню оцінку дії з новою оцінкою, яка враховує фактичний результат і найкращу можливу подальшу дію. Якщо перемикання фази призвело до зменшення черги або часу очікування без погіршення стану суміжних підходів, відповідна Q-оцінка збільшується. Якщо ж дія спричинила накопичення черги, часті зупинки або небажане блокування, її оцінка зменшується через штрафну складову винагороди. Такий механізм робить Q-навчання придатним для задач, у яких якість рішення проявляється не тільки миттєво, а й через подальшу зміну транспортної ситуації.У найпростішій реалізації стан світлофорного агента можна дискретизувати, наприклад подати довжини черг як рівні «мала», «середня», «велика», а діями вважати «залишити фазу» та «перемкнути

фазу». Проте така схема швидко втрачає практичну зручність. Якщо враховувати кілька підходів, активну фазу, час її дії, показники очікування та сусідні перехрестя, кількість можливих станів різко зростає. У результаті табличне Q-навчання потребує великого обсягу пам’яті, повільно накопичує досвід для рідкісних ситуацій і погано узагальнює поведінку на схожі, але не ідентичні транспортні стани. Саме це обмеження зумовлює перехід до апроксимації Q-функції нейронною мережею.

Deep Q-Network є розвитком Q-навчання, у якому значення Q(s, a) не зберігаються в таблиці, а обчислюються параметризованою нейронною мережею. На вхід такої мережі подається вектор стану світлофорного агента, наприклад довжини черг, час очікування, поточна фаза, тривалість фази та агреговані ознаки сусідніх ділянок. На виході формується набір Q-значень, по одному для кожної допустимої дії. Це особливо зручно для дискретного світлофорного керування: агент не повинен окремо запускати модель для кожної дії, а може одним прямим проходом отримати оцінки всіх варіантів і вибрати той, що має найбільше значення [13].

$$Q(s_t, a_t; \theta) \approx Q^*(s_t, a_t), \tag{2.12}$$

де $Q(s_t, a_t; \theta)$ – апроксимована нейронною мережею оцінка дії a_t у стані s_t;

θ – параметри нейронної мережі;

$Q^*(s_t, a_t)$ – оптимальна функція цінності дії.

Навчання DQN ґрунтується на мінімізації розбіжності між поточним прогнозом мережі та цільовим Q-значенням. Ціль формується за логікою Q-навчання: вона включає отриману винагороду та найкращу оцінку дії в наступному стані. У задачі світлофорного керування це означає, що мережа навчається пов’язувати поточний транспортний стан із наслідками конкретного керуючого рішення. Якщо дія «залишити поточну фазу» в умовах зростання черги на конфліктному напрямку систематично призводить до збільшення штрафу, модель поступово зменшує її Q-оцінку

для подібних станів. Якщо дія «перемкнути фазу» після досягнення мінімального зеленого часу дає кращий баланс черг, її оцінка зростає. Рисунок 2.4 ілюструє схему використання DQN для вибору світлофорної дії.

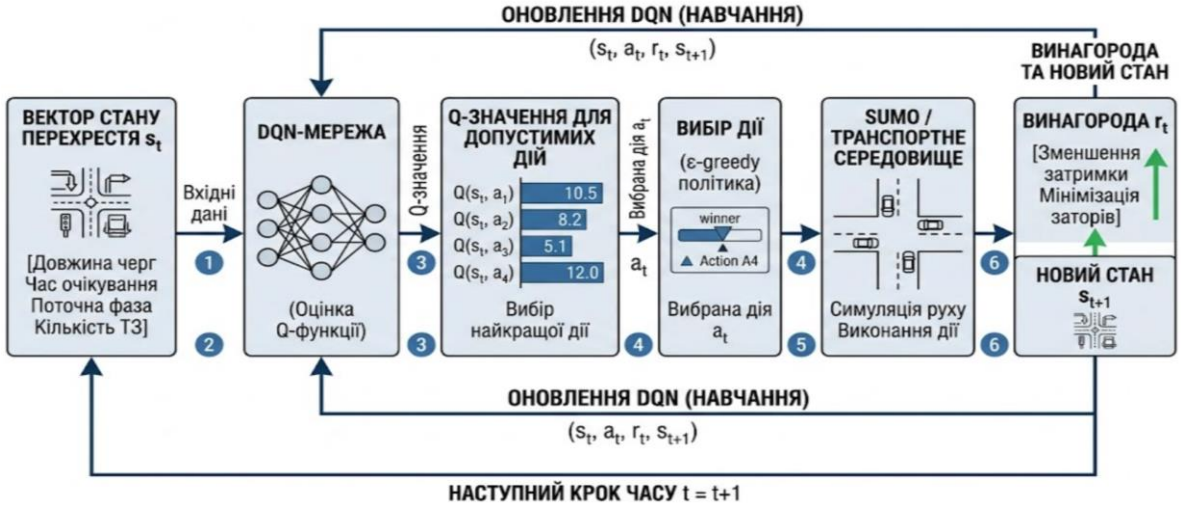


Рисунок 2.4 – Схема використання DQN для вибору світлофорної дії

Формально цільове значення можна подати так:

$$y_t = r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a'; \theta^-), \tag{2.13}$$

де y_t – цільове Q-значення для оновлення мережі;

r_{t+1} – винагорода після виконання дії;

γ – коефіцієнт дисконтування;

$Q(s_{t+1}, a'; \theta^-)$ – оцінка дії в наступному стані, отримана за допомогою цільової мережі;

θ^- – параметри цільової мережі.

Для стабілізації навчання в DQN зазвичай використовуються два механізми: буфер відтворення досвіду та цільова мережа. Буфер зберігає переходи виду (s_t, a_t, r_{t+1}, s_{t+1}), після чого навчання виконується не лише на останньому спостереженні, а на випадкових міні-вибірках із накопиченого

досвіду. У транспортній задачі це зменшує залежність оновлень від послідовності конкретного симуляційного епізоду, де протягом певного часу можуть переважати однотипні стани, наприклад ранкове перевантаження одного напрямку. Цільова мережа, параметри якої оновлюються повільніше, запобігає ситуації, коли модель одночасно змінює і прогноз, і ціль для власного навчання. Без цих механізмів DQN може демонструвати нестабільність, особливо коли винагорода залежить від кількох взаємопов'язаних показників: черг, затримок, кількості зупинок і пропуску транспортних засобів [13]. Вибір дії після обчислення Q-значень може виконуватися жадібно, тобто через дію з найбільшою оцінкою. Однак під час навчання така стратегія може передчасно закріпити неефективну поведінку, якщо агент випадково отримав високі оцінки для обмеженої кількості станів. Тому на етапі навчання часто застосовується ε-жадібна стратегія, яка поєднує використання вже знайдених рішень із дослідженням альтернативних дій:

$$a_t = \begin{cases} \text{random}(A), & \text{з імовірністю } \varepsilon, \\ \underset{a \in A}{\text{argmax}} Q(s_t, a; \theta), & \text{з імовірністю } 1 - \varepsilon, \end{cases} \tag{2.14}$$

де a_t – дія світлофорного агента на кроці t;

A – множина допустимих дій;

ε – імовірність випадкового вибору дії;

$Q(s_t, a; \theta)$ – оцінка дії a у стані s_t, обчислена DQN-мережею.

У контексті цієї кваліфікаційної роботи DQN доцільно розглядати не як повністю автономний замінник інженерної логіки світлофорного об'єкта, а як механізм вибору найдоцільнішої дії з уже дозволеної множини. Це узгоджується з попередньо сформульованими обмеженнями: агент не створює небезпечні фази, не порушує мінімальні тривалості сигналів і не ігнорує перехідні інтервали. Нейронна мережа лише оцінює, яка з доступних дій у поточному стані має більшу очікувану корисність. Такий

підхід зберігає зв’язок між методами штучного інтелекту та реальною логікою транспортного керування.

Порівняльний аналіз особливостей застосування класичного Q-навчання та методу DQN у задачі світлофорного керування наведено в таблиці 2.3.

Таблиця 2.3 – Порівняння Q-навчання та DQN у задачі світлофорного керування

Ознака	Q-навчання	Deep Q-Network
Подання Q-функції	Таблиця значень для пар «стан – дія»	Нейронна мережа, що апроксимує Q-значення
Тип дій	Дискретні дії	Дискретні дії
Робота зі складними станами	Обмежена через швидке зростання таблиці	Краща завдяки узагальненню за ознаками стану
Вимоги до даних	Менші для малих задач	Вищі, потрібні багаторазові симуляційні епізоди
Стабільність навчання	Вища в малих дискретних просторах	Потребує буфера досвіду та цільової мережі
Придатність для світлофорного агента	Корисне як базова методична модель	Доцільне для реалістичніших транспортних станів

Разом із тим DQN має суттєві обмеження. Метод орієнтований на дискретний простір дій, тому він зручний для вибору фази або рішення «продовжити / перемкнути», але менш природний для неперервного керування тривалістю зеленого сигналу. Крім того, якість навчання залежить від складу стану, масштабу ознак, форми винагороди та різноманітності симуляційних сценаріїв. Якщо навчальні епізоди охоплюють лише один режим попиту, агент може сформувати політику, яка добре працює для цього сценарію, але гірше переноситься на інші умови. Для мультиагентної системи додається ще одна складність: дії одного світлофора змінюють умови навчання для інших агентів, тому середовище стає нестаціонарним. Це не заперечує доцільності DQN, але вимагає

обережного формування експериментального контуру та подальшого порівняння з іншими методами.

2.5 Методи політик і алгоритм Proximal Policy Optimization

У попередньому підрозділі DQN було розглянуто як метод, що навчає світлофорного агента через оцінювання корисності окремих дискретних дій. Така логіка добре узгоджується з вибором однієї з наперед заданих фаз, однак вона не вичерпує можливостей підкріплювального навчання для світлофорного регулювання. Інший підхід полягає в безпосередньому навчанні політики, тобто правила, яке визначає ймовірність вибору керуючої дії за поточним станом транспортного середовища.

Методи політик відрізняються від Q-підходів тим, що вони не обов’язково будують явну оцінку всіх можливих дій у кожному стані. Замість цього параметризована модель $\pi\theta(a|s)$ безпосередньо задає поведінку агента: для дискретного простору вона формує розподіл імовірностей між допустимими фазами, а для параметризованого або неперервного простору може визначати числовий параметр керування, наприклад тривалість продовження зеленого сигналу в заданих межах. Для світлофорного регулювання це має методичне значення. Якщо DQN природно відповідає задачі «обрати одну дію з фіксованої множини», то методи політик дають змогу гнучкіше описувати керування, у якому дія може поєднувати вибір фази та її часовий параметр. При цьому інженерні обмеження не зникають: політика не повинна створювати небезпечні комбінації сигналів, а лише обирає допустимий варіант у межах фазової структури, мінімального зеленого часу та перехідних інтервалів [11].

Метою навчання політики є максимізація очікуваної сумарної винагороди. У транспортній задачі така винагорода може враховувати зменшення черг, часу очікування, кількості зупинок, ознак блокування та

покращення пропуску транспортних засобів. Формально цільову функцію для параметризованої політики можна записати так:

$$J(\theta) = \mathbb{E}_{\pi_\theta}[\sum_{t=0}^{T} \gamma^t r_t], \quad (2.15)$$

де $J(\theta)$ – очікувана якість політики з параметрами θ;

π_θ – параметризована політика агента;

$\mathbb{E}_{\pi_\theta}$ – математичне сподівання за траєкторіями, сформованими цією політикою;

γ – коефіцієнт дисконтування;

r_t – винагорода на кроці t;

T – тривалість епізоду моделювання.

Безпосередня оптимізація політики має перевагу в задачах, де бажано уникнути жорсткого зіставлення кожного стану з одним найкращим варіантом дії. У світлофорному керуванні це важливо через стохастичність транспортного потоку. Два схожі стани можуть відрізнятися прихованими чинниками: швидкістю наближення транспортних засобів, нерівномірністю прибуття на сусідніх підходах, залишковими чергами після попередньої фази або впливом суміжного перехрестя.

Стохастична політика дає змогу агенту на етапі навчання досліджувати кілька допустимих рішень і поступово зміщувати ймовірність у бік тих дій, які дають кращий довгостроковий результат. Водночас така гнучкість створює ризик нестабільних оновлень: якщо параметри політики змінюються занадто різко, агент може швидко втратити вже знайдену корисну поведінку. Для світлофорної системи це особливо небажано, оскільки нестабільна політика може спричиняти часті перемикання, накопичення черг або нерівномірне обслуговування напрямків.

Алгоритм Proximal Policy Optimization належить до методів градієнта політики та був запропонований як практичний спосіб стабілізувати оновлення політики без надмірно складної процедури оптимізації [14]. Його

основна ідея полягає в тому, що нова політика не повинна суттєво відхилятися від попередньої за один крок навчання. Такий принцип добре відповідає задачі світлофорного регулювання: агент може поступово вдосконалювати поведінку, але не повинен після кількох епізодів різко змінювати логіку керування настільки, щоб раніше стабільний режим став хаотичним.

Ключовим елементом PPO є відношення ймовірностей вибору тієї самої дії старою та новою політикою. Якщо політика після оновлення починає надавати дії значно більшу або значно меншу ймовірність, це означає, що зміна поведінки може бути надто різкою. Для контролю такого процесу використовується співвідношення:

$$\rho_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, \tag{2.16}$$

де $\rho_t(\theta)$ – відношення ймовірностей вибору дії після та до оновлення політики;

$\pi_\theta(a_t \mid s_t)$ – імовірність вибору дії a_t у стані s_t новою політикою;

$\pi_{\theta_{\text{old}}}(a_t \mid s_t)$ – імовірність цієї самої дії старою політикою;

θ – поточні параметри політики;

θ_{old} – параметри політики до оновлення.

Для визначення того, чи була дія кориснішою або гіршою за очікувану, у PPO використовується оцінка переваги A_t. У прикладній інтерпретації для світлофорного агента вона показує, наскільки конкретне рішення, наприклад продовження поточної фази або перемикання на іншу допустиму фазу, виявилося кращим порівняно із середньою очікуваною поведінкою в такому стані.

Якщо дія зменшила чергу без надмірного погіршення сусідніх напрямків, її перевага може бути додатною. Якщо дія збільшила затримки

або спричинила небажане накопичення транспортних засобів, перевага набуває від'ємного змісту.

Стабілізація PPO досягається через обмежену, або clipped, цільову функцію. Вона не дозволяє алгоритму отримувати надмірний виграш від занадто великої зміни ймовірності дії, навіть якщо поточна оцінка переваги виглядає привабливою. Формально така функція має вигляд:

$$L^{CLIP}(\theta) = \mathbb{E}_t[\min(\rho_t(\theta)A_t, \text{clip}(\rho_t(\theta), 1-\varepsilon, 1+\varepsilon)A_t)], \qquad (2.17)$$

де $L^{CLIP}(\theta)$ – обмежена цільова функція PPO;

A_t – оцінка переваги дії на кроці t;

ε – параметр допустимого відхилення нової політики від старої;

$\text{clip}(\rho_t(\theta), 1-\varepsilon, 1+\varepsilon)$ – операція обмеження відношення ймовірностей у заданому інтервалі;

$\mathbb{E}_t$ – усереднення за кроками навчальної вибірки.

Зміст цієї формули полягає не лише в математичному обмеженні градієнта, а й у практичному контролі поведінки агента. У світлофорній задачі одиничне вдале рішення не повинно одразу змушувати політику майже завжди повторювати цю дію в подібних станах. Наприклад, якщо продовження зеленої фази для магістрального напрямку в одному епізоді зменшило сумарну затримку, це ще не означає, що така дія буде найкращою за іншого розподілу попиту або за іншого стану сусіднього перехрестя. Обмеження зміни політики зменшує ризик переадаптації до окремих епізодів симуляції та підтримує поступовий характер навчання.

Саме через це PPO часто розглядається як зручний компроміс між гнучкістю методів політик і потребою в стабільному навчанні в складних середовищах (рисунок 2.5) [14].

У практичній реалізації PPO зазвичай використовує архітектуру actor-critic. Актор відповідає за політику, тобто формує розподіл імовірностей дій або параметри керуючого впливу. Критик оцінює стан через функцію V(s),

що дає змогу визначити, наскільки отримана винагорода відрізняється від очікуваного рівня. Для світлофорного регулювання такий поділ є змістовно зручним. Актор безпосередньо задає поведінку контролера, а критик допомагає оцінити, чи була ця поведінка виправданою з погляду черг, затримок і пропуску транспортних засобів. При цьому оцінка критика не повинна трактуватися як окрема інженерна модель транспортного потоку. Вона є допоміжною функцією навчання, що узагальнює досвід симуляційних епізодів і використовується для коригування політики.

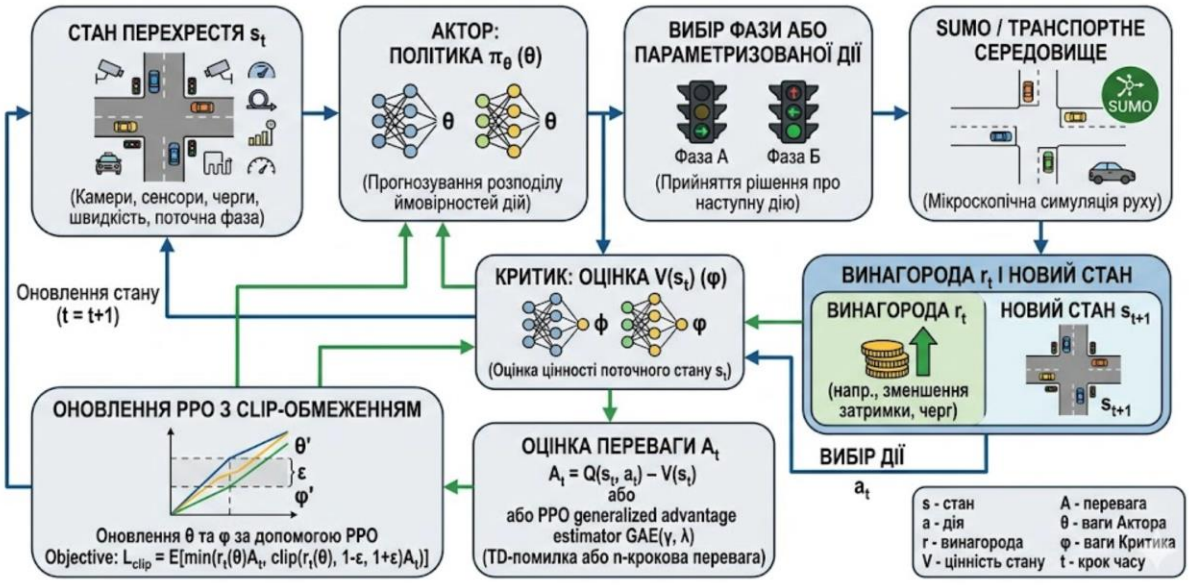


Рисунок 2.5 – Схема навчання світлофорної політики за алгоритмом PPO

Порівняння PPO з DQN у межах цієї роботи доцільно проводити не як протиставлення «кращого» й «гіршого» методу, а як зіставлення двох різних способів навчання світлофорного агента. DQN оцінює якість кожної допустимої дії та обирає дію з найбільшим Q-значенням. Це робить його природним вибором для дискретного керування фазами, де кількість дій невелика й наперед визначена. PPO, навпаки, безпосередньо змінює політику, тому краще пристосований до ситуацій, де потрібно досліджувати стохастичну поведінку, параметризовані дії або плавніше керування

часовими характеристиками сигналів. Водночас PPO зазвичай потребує значної кількості взаємодій із середовищем, уважного налаштування винагороди та контролю стабільності навчання. У симуляційному середовищі SUMO це є прийнятним, оскільки навчальні епізоди можна багаторазово повторювати без ризику для реального руху, однак обчислювальні витрати та чутливість до параметрів залишаються важливими обмеженнями [7]. Таблиця 2.4 містить порівняння методів DQN і PPO для задачі світлофорного регулювання.

Таблиця 2.4 – Порівняння DQN і PPO для задачі світлофорного регулювання

Ознака	DQN	PPO
Основний принцип	Навчання Q-функції для пар «стан – дія»	Безпосереднє навчання параметризованої політики
Тип дій	Переважно дискретні дії	Дискретні, параметризовані або неперервні дії
Вибір світлофорної дії	Через найбільше Q-значення	Через розподіл імовірностей або параметри політики
Стабілізація навчання	Буфер досвіду та цільова мережа	Обмеження зміни політики через clip-функцію
Придатність для фазового керування	Висока для фіксованої множини фаз	Висока для фаз і гнучкіших часових параметрів
Основне обмеження	Менш природна робота з неперервними параметрами	Чутливість до винагороди, параметрів і кількості симуляцій

Для мультиагентної системи світлофорного регулювання PPO має додаткову цінність, оскільки стохастична політика може пом’якшувати проблему передчасного закріплення локально вигідних рішень. Якщо кожен агент швидко фіксує одну дію як найкращу, транспортна мережа може перейти до режиму, у якому окремі перехрестя зменшують власні черги, але погіршують умови для сусідніх вузлів. Поступове оновлення політики дає змогу довше зберігати дослідження альтернативних режимів і краще

враховувати відкладені наслідки керування. Разом із тим мультиагентність ускладнює навчання PPO так само, як і інших методів підкріплювального навчання: середовище для кожного агента змінюється не лише через транспортний попит, а й через оновлення політик інших агентів. Через це для подальшої реалізації важливо визначити, чи агенти навчаються незалежно, чи використовують спільну інформацію, спільний критик або централізовану процедуру навчання з децентралізованим виконанням.

2.6 Actor-Critic підходи в задачах адаптивного керування рухом

Після розгляду DQN як методу оцінювання дій та PPO як методу безпосереднього навчання політики доцільно окремо проаналізувати підхід Actor-Critic, оскільки саме він поєднує ці дві логіки в одній навчальній схемі. Для задачі адаптивного світлофорного керування така комбінація має методичне значення: агент повинен не лише обирати допустиму фазу або параметр її продовження, а й отримувати числову оцінку того, наскільки це рішення було корисним для подальшого стану транспортного потоку.

Actor-Critic підходи належать до класу методів підкріплювального навчання, у яких функції вибору дії та оцінювання якості стану або дії розділяються між двома взаємопов’язаними компонентами. Actor формує політику $\pi\theta(a|s)$, тобто визначає, яку дію доцільно виконати в поточному стані. Critic, своєю чергою, оцінює поведінку актора через функцію цінності $V\varphi(s)$, $Q\varphi(s,a)$ або оцінку переваги A_t. Такий поділ зменшує недолік чистих методів політик, де оновлення може мати високу дисперсію, і водночас не обмежує модель лише жадібним вибором дії за Q-значенням, як це характерно для DQN. У транспортній задачі це означає, що світлофорний агент може зберігати гнучкість стохастичної політики, але навчатися не тільки за фактом отриманої винагороди, а й за відхиленням цієї винагороди від очікуваного рівня [11].

У найпростішій інтерпретації Actor відповідає за керуючу частину світлофорного контролера. На вхід він отримує вектор стану перехрестя, який може містити довжини черг, активну фазу, час її дії, показники очікування та агреговану інформацію про сусідні вузли. На виході Actor формує розподіл імовірностей між допустимими діями або параметри дії, якщо керування передбачає не лише вибір фази, а й зміну тривалості зеленого сигналу. Critic при цьому не керує світлофором безпосередньо. Його роль полягає в оцінюванні того, чи був стан або виконана дія кращими за типовий очікуваний результат для подібної ситуації.

Оцінювання стану критиком можна подати через параметризовану функцію цінності:

$$V_{\varphi}(s_t) \approx \mathbb{E}_{\pi_\theta}[\sum_{k=0}^{T-t} \gamma^k \, r_{t+k} \mid s_t], \tag{2.18}$$

де $V_{\varphi}(s_t)$ – оцінка цінності стану s_t, сформована критиком із параметрами φ;

π_θ – політика актора з параметрами θ;

γ – коефіцієнт дисконтування;

$r_{t+k} \mid s_t$ – винагорода на майбутньому кроці;

T – тривалість епізоду моделювання.

У задачі світлофорного регулювання значення $V_{\varphi}(s_t)$ можна інтерпретувати як прогнозовану якість подальшого розвитку транспортної ситуації за умови, що агент продовжить діяти відповідно до поточної політики. Якщо стан характеризується довгими чергами на кількох підходах, високим часом очікування та ризиком блокування суміжної ділянки, критик повинен формувати нижчу оцінку. Якщо ж поточна фаза забезпечує збалансоване обслуговування напрямків і не створює надмірного накопичення транспортних засобів, оцінка стану має бути вищою. Така інтерпретація важлива тому, що окрема миттєва винагорода не завжди достатньо точно показує наслідок дії. Наприклад, перемикання фази може

тимчасово збільшити чергу на одному підході, але запобігти значно гіршому блокуванню транспортного коридору через кілька наступних кроків.

Зв’язок між Actor і Critic часто реалізується через помилку часової різниці. Вона показує, наскільки фактичний результат після дії відрізняється від попередньої оцінки критика:

$$\delta_t = r_{t+1} + \gamma V_\varphi(s_{t+1}) - V_\varphi(s_t), \tag{2.19}$$

де δ_t – помилка часової різниці;

r_{t+1} – винагорода після виконання дії;

$V_\varphi(s_t)$ – оцінка поточного стану;

$V_\varphi(s_{t+1})$ – оцінка наступного стану;

γ – коефіцієнт дисконтування.

Якщо δ_t має додатне значення, це означає, що результат дії виявився кращим, ніж очікувалося. Для світлофорного агента такою ситуацією може бути вибір фази, після якого зменшилася сумарна затримка або не виникло очікуваного блокування. Якщо δ_t від’ємна, дія дала гірший наслідок порівняно з прогнозом критика: наприклад, продовження зеленої фази для одного напрямку погіршило стан конфліктного підходу або збільшило загальний час очікування. Завдяки цьому Actor отримує не лише грубий сигнал винагороди, а більш інформативний напрям коригування політики.

Оновлення політики в Actor-Critic підходах ґрунтується на тому, що ймовірність корисних дій повинна збільшуватися, а менш корисних – зменшуватися. Для цього використовується градієнт політики з урахуванням оцінки переваги:

$$\nabla_\theta J(\theta) = \mathbb{E}_t[\nabla_\theta \log \pi_\theta(a_t \mid s_t) A_t], \tag{2.20}$$

де $\nabla_\theta J(\theta)$ – напрям оновлення параметрів політики;

$\pi_\theta(a_t \mid s_t)$ – імовірність вибору дії a_t у стані s_t;

A_t – оцінка переваги дії;

θ – параметри актора;

$\mathbb{E}_t$ – усереднення за кроками навчальної вибірки.

Зміст цієї формули полягає в тому, що Actor не просто повторює дії, які випадково дали високу винагороду, а враховує оцінку їхньої переваги відносно очікуваного результату. Це особливо важливо для транспортних потоків, де короткостроковий виграш може бути оманливим. Якщо дія тимчасово зменшила чергу перед одним світлофором, але спричинила перенесення перевантаження до сусіднього вузла, критик має відобразити це через нижчу оцінку подальшого стану. Навпаки, рішення, яке не дає миттєвого максимального виграшу, але підтримує стійкіший режим руху в мережі, може отримати позитивну перевагу. Саме тому Actor-Critic схема краще узгоджується з природою адаптивного керування, ніж методи, які аналізують лише локальну реакцію на окрему дію.

Для алгоритму PPO Actor-Critic архітектура має особливе значення, оскільки обмежене оновлення політики потребує надійної оцінки переваги дій. У попередньому підрозділі було показано, що PPO контролює різкість зміни політики через відношення ймовірностей і clip-обмеження. Проте сама оцінка A_t зазвичай формується саме за участю критика. Тому в практичній реалізації PPO Actor визначає розподіл імовірностей між світлофорними діями, а Critic оцінює, наскільки отриманий результат перевищив або не досяг очікуваного рівня [14]. Для транспортного середовища це дає змогу уникати надмірно різких змін поведінки агента, які могли б спричинити нестабільне перемикання фаз, і водночас підтримувати адаптивність до змін попиту.

Водночас Actor-Critic підходи не усувають усіх складностей задачі. Якість критика залежить від складу стану, стабільності винагороди та різноманітності симуляційних епізодів. Якщо критик систематично переоцінює певні стани, Actor може підсилювати неефективні дії. У мультиагентному світлофорному керуванні ця проблема ускладнюється

тим, що результат дії одного агента залежить від політик інших агентів. Для окремого світлофора середовище стає нестаціонарним: зміна поведінки сусіднього перехрестя може змінити розподіл прибуття транспортних засобів, хоча локальний агент не завжди прямо спостерігає причину такої зміни. Через це в подальшій побудові системи потрібно враховувати, чи використовується незалежне навчання агентів, спільний критик, централізоване навчання з децентралізованим виконанням або обмежений обмін інформацією між сусідніми вузлами [7].

2.7 Мультиагентне підкріплювальне навчання

Мультиагентне підкріплювальне навчання є логічним розвитком попередньо розглянутих методів, оскільки задача світлофорного регулювання в міській мережі не обмежується поведінкою одного ізольованого контролера. Якщо в однокомпонентній постановці агент взаємодіє із середовищем і поступово вдосконалює власну політику, то в мультиагентній системі кілька агентів одночасно приймають рішення, впливають на спільне середовище та змінюють умови навчання один для одного. Для теми цієї роботи така постановка є принциповою: кожне регульоване перехрестя може бути представлене окремим агентом, але наслідки його дій проявляються не лише на локальних підходах, а й у стані суміжних вузлів транспортної мережі.

Формально мультиагентне підкріплювальне навчання часто розглядається як розширення марковського процесу прийняття рішень до марковської гри або стохастичної гри. На відміну від MDP, де діє один агент із власною множиною дій, у такій моделі на кожному кроці формується спільна дія кількох агентів. У транспортній інтерпретації це означає, що стан мережі після чергового кроку залежить не тільки від рішення одного світлофора, а від сукупності рішень усіх керованих перехресть. Якщо один агент продовжує зелену фазу для магістрального напрямку, а сусідній агент

у цей самий момент не готовий прийняти збільшений потік, локально корисна дія може спричинити накопичення транспортних засобів між вузлами. Саме тому в мультиагентній постановці важливо описувати не окрему дію a_t, а вектор дій усіх агентів, а також розрізняти глобальний стан мережі та локальні спостереження окремих перехресть. Узагальнене подання такої моделі можна записати так [15]:

$$G = \langle N, S, \{A^i\}_{i=1}^N, \{O^i\}_{i=1}^N, P, \{R^i\}_{i=1}^N, \gamma \rangle, \tag{2.21}$$

де G – мультиагентна модель навчання;

N – кількість агентів у транспортній мережі;

S – множина глобальних станів середовища;

A^i – множина допустимих дій і-го агента;

O^i – множина локальних спостережень і-го агента;

P – функція переходу між станами;

R^i – функція винагороди і-го агента;

γ – коефіцієнт дисконтування майбутніх результатів.

У задачі світлофорного керування глобальний стан S може містити інформацію про черги, затримки, поточні фази, пропуск транспортних засобів і ознаки блокування в усій досліджуваній мережі. Однак окремий агент зазвичай не має повного доступу до цього стану. Він спостерігає локальний фрагмент середовища: власні підходи, активну фазу, час її дії, локальні черги, час очікування та, за потреби, агреговані дані від сусідніх перехресть.

Через це для світлофорної задачі характерна часткова спостережуваність. Вона не є лише технічною незручністю, а безпосередньо впливає на якість політики: агент може бачити збільшення вхідного потоку, але не завжди знає, чи воно спричинене попитом, дією сусіднього світлофора або тимчасовим блокуванням нижче за потоком.

Спільна дія мультиагентної системи формується як набір локальних рішень усіх світлофорних агентів. Якщо кожне перехрестя має власну політику π^i, то загальна поведінка мережі залежить від поєднання цих політик. У найпростішому децентралізованому випадку агенти обирають дії на основі власних спостережень, і спільну політику можна подати як добуток локальних політик:

$$\pi(a_t \mid o_t) = \prod_{i=1}^{N} \pi^i \left(a_t^i \mid o_t^i\right), \tag{2.22}$$

де $\pi(a_t \mid o_t)$ – спільна політика мультиагентної системи;

a_t– вектор дій усіх агентів на кроці t;

o_t – вектор локальних спостережень агентів;

$\pi^i\left(a_t^i \mid o_t^i\right)$ – політика і-го агента;

a_t^i – дія і-го агента;

o_t^i – локальне спостереження і-го агента.

Такий запис зручний для масштабування, оскільки не потребує побудови єдиного централізованого простору дій для всієї мережі. Проте він також показує одну з головних складностей мультиагентного навчання. Якщо кожен агент змінює власну політику під час навчання, то з погляду інших агентів середовище стає нестаціонарним.

Для окремого світлофора зміна поведінки сусіднього вузла виглядає як зміна транспортного попиту або функції переходів, хоча фактично причиною є оновлення політики іншого агента. У класичному підкріплювальному навчанні агент припускає, що закономірності середовища достатньо стабільні для накопичення досвіду.

У мультиагентній системі це припущення порушується: одна й та сама дія в подібному локальному стані може давати різні наслідки залежно від того, як у цей момент діють інші перехрестя. Для транспортної мережі це особливо помітно в коридорах із послідовними світлофорами, де

несинхронізовані рішення можуть породжувати хвилі зупинок, залишкові черги й блокування проміжних ділянок.

На рисунку 2.6 наведена схема мультиагентного навчання світлофорного регулювання в міській транспортній мережі.

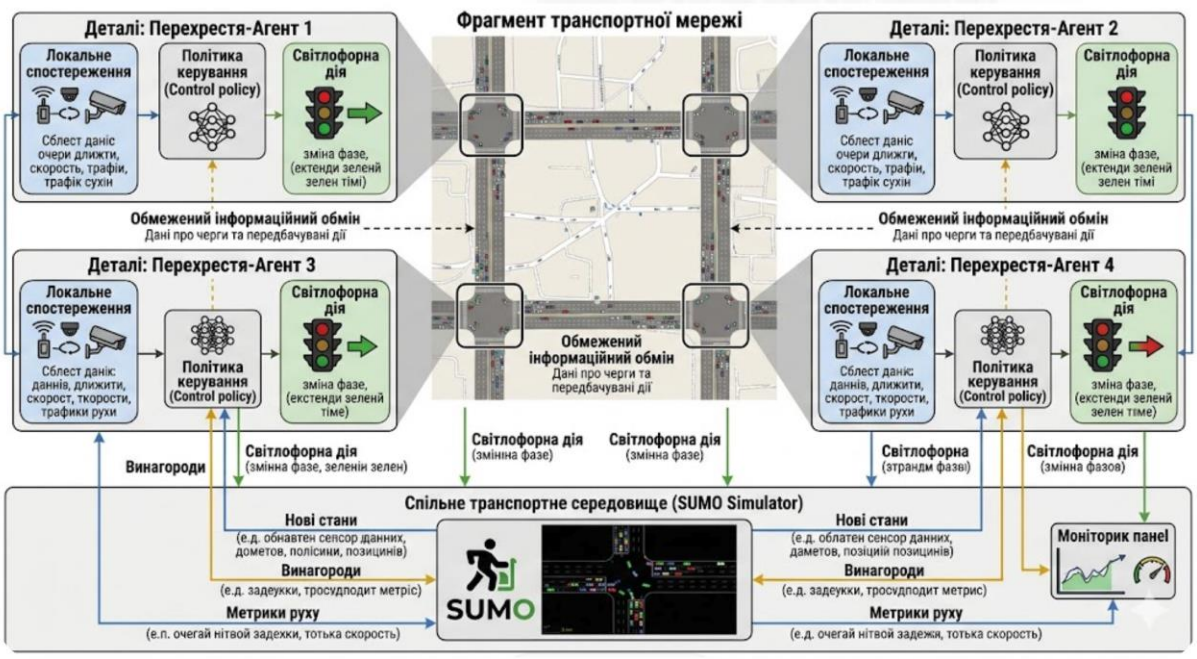


Рисунок 2.6 – Схема мультиагентного навчання світлофорного регулювання в міській транспортній мережі

Одним із ключових методичних питань є вибір між локальною та глобальною винагородою. Локальна винагорода краще пояснює агенту наслідки його власних дій: якщо після вибору фази зменшилася черга на відповідному перехресті, причинний зв’язок є відносно зрозумілим.

Проте суто локальна винагорода може стимулювати егоїстичну поведінку, коли агент швидко очищує власний підхід і передає перевантаження сусідньому вузлу. Глобальна винагорода, навпаки, відображає якість усієї мережі, але ускладнює навчання окремого агента, оскільки йому важко визначити, яка частина загального результату пов’язана саме з його дією.

Тому для задачі світлофорного регулювання доцільним є проміжний підхід, у якому винагорода і-го агента поєднує локальну складову з обмеженим мережевим компонентом:

$$\tilde{r}_t^i = (1-\lambda) r_{local,t}^i + \lambda r_{net,t}^{\mathcal{N}_i}, \tag{2.23}$$

де $\tilde{r}_t^i$ – комбінована винагорода і-го агента;

$r_{local,t}^i$ – локальна винагорода, пов'язана зі станом відповідного перехрестя;

$r_{net,t}^{\mathcal{N}_i}$ – мережевий компонент для околу N_i, тобто для сусідніх перехресть або транспортного коридору;

λ – коефіцієнт балансу між локальним і мережевим критеріями.

Зміст формули (2.24) полягає не в заміні раніше визначеної функції винагороди, а в уточненні її мультиагентної структури.

Локальна складова може враховувати черги, очікування, пропуск транспортних засобів і штрафи за небажане перемикання, як було показано в попередній MDP-постановці. Мережевий компонент доцільно формувати не для всієї міської мережі без обмежень, а для транспортно пов'язаного околу агента: сусідніх перехресть, вихідних ділянок або коридору, у якому дія агента має відчутний відкладений ефект. Це зменшує шум у навчальному сигналі й водночас не дозволяє агенту повністю ігнорувати наслідки власних рішень для інших вузлів.

Значення λ визначає характер поведінки системи: за λ, близького до нуля, агенти діють майже локально; за більшого λ вони сильніше орієнтуються на узгодженість руху в мережі.

У практичній реалізації цей параметр потребує експериментального добору, оскільки надмірна глобалізація винагороди може погіршити здатність агентів навчатися причинно зрозумілих дій.

Таблиця 2.5 містить порівняння підходів до організації мультиагентного навчання світлофорів.

Таблиця 2.5 – Порівняння підходів до організації мультиагентного навчання світлофорів

Підхід	Зміст для транспортної мережі	Переваги	Обмеження
Незалежне навчання агентів	Кожне перехрестя навчає власну політику за локальними спостереженнями	Простота реалізації, добра масштабованість	Слабке врахування впливу на сусідів, нестаціонарність середовища
Обмін даними між сусідами	Агент отримує агреговані показники суміжних перехресть	Краща координація в коридорах і щільних мережах	Потреба в надійних даних, ризик збільшення розмірності стану
Спільна або змішана винагорода	Локальний критерій доповнюється мережевим компонентом	Зменшення локально егоїстичної поведінки	Складніше встановити причинний зв’язок між дією та результатом
Централізоване навчання з децентралізованим виконанням	Під час навчання використовується ширша інформація, а під час виконання агент діє локально	Поєднання координації та автономності	Більші вимоги до навчального середовища й архітектури моделі

Окремої уваги потребує підхід централізованого навчання з децентралізованим виконанням. Його суть полягає в тому, що під час навчання модель може використовувати ширшу інформацію про стан мережі або дії інших агентів, але під час фактичного керування кожен агент приймає рішення на основі власного спостереження та обмеженого обміну з сусідами. Для світлофорного регулювання такий підхід є методично привабливим, оскільки він не вимагає, щоб у реальному часі всі перехрестя залежали від єдиного централізованого рішення. Водночас на етапі навчання можна краще оцінювати міжагентні залежності: які дії призводять

до передавання перевантаження, які підтримують рух транспортного пакета коридором, а які створюють небажане блокування. У працях із мультиагентного Actor-Critic підходу підкреслюється, що врахування політик або дій інших агентів під час навчання може зменшувати проблему нестаціонарності та полегшувати координацію [17].

Для задачі великомасштабного світлофорного керування мультиагентність також пов'язана з проблемою розмірності [18]. Якщо намагатися будувати єдиного агента для всієї мережі, простір дій швидко зростає зі збільшенням кількості перехресть. Наприклад, якщо кожне з N перехресть має m допустимих дій, то кількість комбінацій спільної дії становить m^N. Навіть за невеликого m така залежність робить централізоване керування важким для навчання та обчислення. Це можна подати так:

$$|A| = \prod_{i=1}^{N} |A^i|, \tag{2.24}$$

де $|A|$ – кількість можливих спільних дій усієї системи;

$|A^i|$ – кількість допустимих дій і-го агента;

N – кількість агентів у транспортній мережі.

Дана формула пояснює, чому повністю централізований підхід стає обмеженим для міських мереж із багатьма світлофорними об'єктами. Мультиагентне подання не усуває складність повністю, але розподіляє її між локальними агентами й дозволяє обмежувати взаємодію фізично або інформаційно близькими вузлами. У дослідженнях мультиагентного глибокого підкріплювального навчання для керування світлофорами підкреслюється, що розподіл глобального керування між локальними агентами є одним зі способів подолання масштабованості, хоча він одночасно породжує проблеми часткової спостережуваності та координації [16]. Для цієї роботи це означає, що мультиагентна система повинна проектуватися не як сукупність повністю незалежних моделей, а як

взаємопов’язана структура, у якій локальні політики навчаються з урахуванням транспортної топології.

Практичне значення мультиагентного підходу полягає також у можливості природно врахувати графову структуру міської мережі. Перехрестя відповідають вузлам графа, а дорожні ділянки між ними – ребрам, через які передаються транспортні потоки та інформаційний вплив. У такій інтерпретації агенту не потрібно отримувати стан усіх віддалених перехресть. Достатньо визначити релевантний окіл: попередні й наступні вузли за напрямком руху, суміжні підходи або перехрестя, з якими існує ризик взаємного блокування. Це робить обмін інформацією змістовно обмеженим. Наприклад, агент може враховувати не повну фазову програму сусіда, а агреговані ознаки: довжину черги на вихідній ділянці, поточну завантаженість приймального напрямку або прогнозоване прибуття транспортного пакета. Такий рівень взаємодії є достатньо інформативним для координації, але не перетворює локального агента на компонент громіздкої централізованої системи.

Водночас мультиагентне підкріплювальне навчання має низку обмежень, які потрібно враховувати під час подальшої розробки. По-перше, одночасне навчання кількох політик може призводити до нестабільності: агенти адаптуються один до одного, і досвід, зібраний на ранніх етапах, може втрачати актуальність після зміни політик сусідніх вузлів. По-друге, надто частий або надмірно широкий обмін інформацією збільшує розмірність вхідного стану, ускладнює навчання та може створити залежність від даних, які в реальній системі надходять із затримкою. По-третє, транспортні обмеження залишаються жорсткими незалежно від алгоритму навчання: агент не повинен порушувати фазову сумісність, мінімальні тривалості сигналів, перехідні інтервали та вимоги безпеки. Тому мультиагентна модель має розглядатися як механізм вибору серед допустимих керуючих дій, а не як повністю вільний генератор світлофорних режимів.

2.8 Формування простору станів для агента світлофорного регулювання

Формування простору станів є одним із ключових методичних етапів побудови світлофорного агента, оскільки саме через стан модель навчання отримує уявлення про поточну транспортну ситуацію. У попередніх підрозділах було визначено, що агент обирає дію в межах допустимої фазової структури, а якість рішення оцінюється через зміну черг, затримок, пропуску та мережевої стійкості. Проте ці елементи не можуть працювати ізольовано від способу подання вхідних даних. Якщо стан не відображає суттєві характеристики руху, навіть коректно вибраний алгоритм DQN, PPO або Actor-Critic не матиме достатньої інформації для навчання ефективної політики.

У задачі світлофорного регулювання стан агента доцільно розглядати як вектор ознак, що описує не всю транспортну мережу, а той фрагмент середовища, який має безпосереднє значення для прийняття рішення на конкретному перехресті. До такого фрагмента належать підходи до перехрестя, смуги руху, активна світлофорна фаза, тривалість її дії, поточні черги, середні швидкості транспортних засобів, затримки та ознаки перевантаження вихідних ділянок. Для мультиагентної постановки цей локальний опис може доповнюватися агрегованими показниками сусідніх перехресть, однак їх потрібно включати обмежено. Надмірне розширення стану збільшує розмірність задачі, ускладнює навчання нейронної мережі та може створити залежність від даних, які в реальній системі надходять із затримкою або похибкою. Тому методично виправданим є не максимальне накопичення всіх доступних параметрів, а відбір таких ознак, які пояснюють поточний попит, залишкову чергу, готовність підходу до обслуговування та ризик передавання перевантаження на сусідні ділянки [7].

Формально локальний стан і-го світлофорного агента на кроці t можна подати як поєднання фазових, потокових і мережевих характеристик:

$$s_t^i = \left[p_t^i, \tau_t^i, q_t^i, v_t^i, w_t^i, k_t^i, o_t^i, n_t^i\right], \quad (2.25)$$

де s_t^i – вектор стану і-го світлофорного агента на кроці t;

p_t^i – поточна світлофорна фаза;

τ_t^i – час, протягом якого активна фаза вже діє;

q_t^i – вектор довжин черг за підходами або смугами;

v_t^i – вектор середніх швидкостей;

w_t^i – вектор середніх затримок або часу очікування;

k_t^i – вектор щільності транспортного потоку;

o_t^i – вектор зайнятості смуг;

n_t^i – агреговані показники сусідніх перехресть або суміжних ділянок.

Наявність поточної фази p_t^i у складі стану є обов'язковою, оскільки однакові черги за різних фаз мають різний керуючий зміст. Якщо перевантажений підхід уже отримує зелений сигнал, доцільною дією може бути продовження обслуговування або очікування завершення мінімального зеленого часу. Якщо той самий підхід перебуває під заборонним сигналом і черга швидко зростає, агент повинен розглядати можливість переходу до відповідної фази після виконання обмежень безпеки. Час дії фази τ_t^i також має самостійне значення: без цієї ознаки модель не відрізнятиме ситуацію, коли фаза тільки розпочалася, від ситуації, коли вона вже вичерпала більшу частину корисного зеленого часу. Це важливо для запобігання надто частим перемиканням, які збільшують втрати на перехідні інтервали, і для уникнення надмірного утримання одного напрямку, що погіршує обслуговування конфліктних потоків.

Основними транспортними ознаками стану є довжина черги, швидкість, затримка, щільність і зайнятість смуг. Довжина черги q показує

накопичений незадоволений попит на підході та безпосередньо пов’язана з потребою в зеленому часі. Проте використовувати лише чергу недостатньо. Наприклад, коротка черга на підході з низькою швидкістю може свідчити про початок блокування або нерівномірне прибуття транспортних засобів, тоді як довша черга, яка швидко розсмоктується під час активної фази, не обов’язково потребує радикальної зміни політики. Середня швидкість v уточнює динаміку руху: вона дає змогу відрізнити накопичення стоячих транспортних засобів від щільного, але рухомого потоку. Середня затримка w відображає якість обслуговування з позиції учасників руху, а щільність k та зайнятість смуг o допомагають оцінити, наскільки близькою є ділянка до насичення. У методиках аналізу пропускної здатності регульованих перехресть саме взаємозв’язок попиту, затримки, насичення та використання смуг є основою для оцінювання якості руху [3]. Для навчального агента ці показники виконують роль вхідних ознак, через які модель може пов’язати фазове рішення з фактичним станом транспортного потоку. Таблиця 2.6 містить доцільні ознаки простору станів світлофорного агента.

Таблиця 2.6 – Доцільні ознаки простору станів світлофорного агента

Група ознак	Приклад параметра	Зміст для агента	Методичне обмеження
1	2	3	4
Фазові ознаки	Поточна фаза, час дії фази	Показують, які потоки зараз обслуговуються і чи допустиме перемикання	Потрібно враховувати мінімальні тривалості та перехідні інтервали
Черги	Кількість транспортних засобів у черзі за підходами або смугами	Відображають накопичений попит і залишкове перевантаження	Черга не пояснює швидкість прибуття транспортних засобів сама по собі

Продовження таблиці 2.6

1	2	3	4
Швидкість	Середня швидкість на підході	Допомагає відрізнити рухомий потік від фактичного затору	Потребує усереднення, щоб уникнути шуму від окремих транспортних засобів
Затримка	Середній або сумарний час очікування	Показує якість обслуговування напрямку	Може накопичуватися повільніше, ніж змінюється поточна черга
Щільність і зайнятість	Частка зайнятої смуги, кількість авто на одиницю довжини	Виявляють наближення ділянки до насичення або блокування	Потребують коректного визначення довжини контрольованої зони
Сусідні показники	Черга або зайнятість вихідної ділянки сусіднього перехрестя	Дають змогу враховувати ризик передавання перевантаження	Надмірний обмін збільшує розмірність стану

Окрему увагу потрібно приділити нормалізації ознак, оскільки вхідні параметри мають різні масштаби. Час фази може вимірюватися секундами, довжина черги – кількістю транспортних засобів, швидкість – кілометрами на годину або метрами за секунду, а зайнятість смуги – часткою від 0 до 1. Якщо ці величини безпосередньо подати до нейронної мережі, ознаки з більшими числовими значеннями можуть непропорційно впливати на навчання, хоча їхній зміст не обов’язково важливіший. Тому для практичної реалізації доцільно використовувати масштабування відносно максимально допустимого або характерного значення.

Для довжини черги та затримки це можна записати так:

$$\hat{x}_{t,j}^{i} = \frac{x_{t,j}^{i}}{x_{max,j}^{i}+\varepsilon}, \quad (2.26)$$

де $\hat{x}_{t,j}^{i}$ – нормалізоване значення j-ї ознаки і-го агента на кроці t;

$x_{t,j}^{i}$ – початкове значення відповідної ознаки;

$x_{max,j}^{i}$ – максимальне або опорне значення для масштабування;

ε – мала додатна величина, що запобігає діленню на нуль.

У симуляційному середовищі SUMO такі ознаки можуть бути отримані з мікроскопічної моделі руху, детекторів або агрегованих показників смуг і ребер мережі. Це узгоджується з обраною інструментальною основою роботи, оскільки SUMO підтримує покрокове оновлення положення транспортних засобів, станів світлофорів і показників руху, необхідних для навчального циклу [8].

Водночас під час формування стану не слід покладатися лише на ті змінні, які легко зчитати з симулятора. Ознака має включатися до стану тоді, коли вона пояснює керуючу ситуацію. Наприклад, ідентифікатор поточної фази є змістовним, бо визначає доступні переходи; середня швидкість на підході є змістовною, бо відображає режим руху; абсолютний номер транспортного засобу в симуляції, навпаки, не має прямого значення для вибору світлофорної дії. Такий відбір зменшує ризик перенавчання на технічні деталі експериментального середовища й підвищує ймовірність того, що навчена політика відображатиме саме закономірності транспортного керування.

Для мультиагентної системи важливим є також питання включення сусідніх показників. Повний глобальний стан усієї мережі теоретично може містити більше інформації, але практично він ускладнює модель і погіршує масштабованість. Більш доцільним є локально-мережеве подання, за якого агент отримує власні детальні ознаки та обмежений набір агрегованих характеристик від найближчих вузлів. Такими характеристиками можуть бути довжина черги на вихідній ділянці, зайнятість приймальної смуги,

активна фаза сусіднього перехрестя або очікуваний транспортний пакет, що наближається до контрольованого вузла.

Це безпосередньо пов’язано з проблемою координації, розглянутою в попередньому підрозділі: агент повинен уникати рішень, які локально зменшують власну чергу, але передають перевантаження нижче за потоком. У дослідженнях мультиагентного керування світлофорами підкреслюється, що врахування локальних і сусідніх ознак допомагає поєднати масштабованість із координацією, хоча потребує обережного обмеження інформаційного обміну [16].

Загальне подання локально-мережевого стану можна записати через власні ознаки агента та ознаки його транспортного околу:

$$o_t^i = \left[s_{local,t}^i, \{g_t^j \mid j \in \mathcal{N}_i\}\right], \tag{2.27}$$

де o_t^i – локальне спостереження і-го агента;

$s_{local,t}^i$ – власний локальний стан перехрестя;

g_t^j – агрегований показник j-го сусіднього вузла або суміжної ділянки;

$\mathcal{N}_i$ – множина сусідів і-го агента, які входять до його інформаційного околу.

Такий вектор може включати інформацію про поточну фазу світлофора, довжину черг, інтенсивність транспортного потоку, час очікування транспортних засобів та інші параметри, що характеризують стан перехрестя в певний момент часу. Рисунок 2.7 демонструє схему формування вектора стану світлофорного агента.

Під час вибору складу стану потрібно враховувати компроміс між інформативністю та розмірністю. Занадто бідний стан, наприклад лише поточна фаза і сумарна кількість транспортних засобів на перехресті, не дає змоги агенту розрізняти просторово різні ситуації. У такому випадку одна й та сама кількість автомобілів може відповідати рівномірному навантаженню

всіх підходів або критичному накопиченню на одному напрямку. Занадто детальний стан, навпаки, може включати надлишкові або нестабільні параметри, які збільшують час навчання та погіршують узагальнення. Для цієї роботи доцільним є проміжний варіант: стан має містити фазову інформацію, агреговані показники за підходами або смугами, характеристики затримки й рухомості потоку, а також обмежені ознаки сусідніх вузлів, важливих для координації.

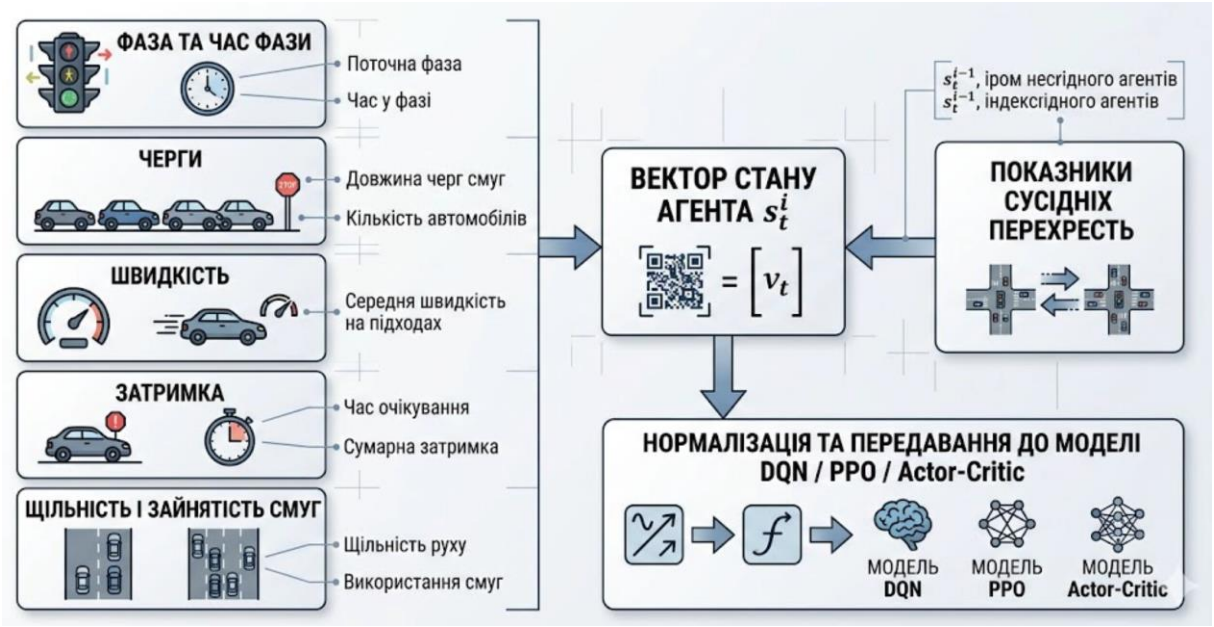


Рисунок 0.1 – Схема формування вектора стану світлофорного агента

2.9 Формування простору дій агента

Після визначення простору станів світлофорного агента необхідно сформувати множину керуючих дій, які агент може виконувати у відповідь на поточну транспортну ситуацію. Якщо стан описує, що саме агент спостерігає на перехресті, то простір дій визначає межі його впливу на світлофорне керування. Для задачі навчання це має принципове значення: навіть інформативний вектор стану не забезпечить ефективної політики,

якщо агент отримує надто широкий, нечіткий або технічно недопустимий набір дій.

У світлофорному регулюванні дія агента не може трактуватися як довільна зміна сигналів. Вона повинна відповідати фазовій структурі перехрестя, правилам сумісності транспортних і пішохідних потоків, мінімальним тривалостям зеленого сигналу, перехідним інтервалам і вимогам безпеки руху [1]. Тому простір дій формується не шляхом повного перебору всіх можливих комбінацій світлофорних сигналів, а через заздалегідь допустимі керуючі варіанти.

До таких варіантів можуть належати збереження поточної фази, перемикання на наступну фазу за встановленою послідовністю, вибір однієї з допустимих фаз, збільшення або зменшення тривалості зеленого сигналу, а також застосування попередньо визначеного плану керування.

У межах навчальної моделі кожна з цих дій має бути не лише формально доступною, а й змістовно пов’язаною з транспортним наслідком: зміною черги, часу очікування, пропуску транспортних засобів або ризику блокування суміжної ділянки. Таблиця 2.7 містить можливі дії світлофорного агента.

Таблиця 2.7 – Можливі дії світлофорного агента

Дія агента	Зміст керування	Потенційний ефект	Основне обмеження
1	2	3	4
Залишити поточну фазу	Поточна група напрямків продовжує отримувати дозвіл на рух	Зменшення черги на активному напрямку	Ризик надмірного очікування конфліктних потоків
Перемкнути фазу	Агент ініціює перехід до наступної дозволеної фази	Обслуговування напрямку з накопиченою чергою	Потрібні жовтий сигнал, міжзелений інтервал і мінімальний зелений час

Продовження таблиці 2.7

1	2	3	4
Обрати наступну фазу	Агент вибирає одну з наперед дозволених фаз	Гнучкіше реагування на нерівномірний попит	Потрібно заборонити конфліктні фазові комбінації
Змінити тривалість зеленого сигналу	Агент подовжує або скорочує обслуговування поточної фази	Точніше використання пропускної здатності	Потрібні нижня й верхня межі тривалості
Застосувати план керування	Агент обирає один із готових режимів роботи	Стабільність у типових сценаріях попиту	Менша адаптивність до короткочасних збурень

Формально простір дій і-го світлофорного агента на кроці t можна подати як скінченну множину допустимих керуючих рішень:

$$A^i = \{a^i_{hold}, a^i_{switch}, a^i_{select,1}, \dots, a^i_{select,m}, a^i_{extend}, a^i_{plan}\}, \qquad (2.28)$$

де A^i – множина допустимих дій і-го агента;

a^i_{hold} – дія збереження поточної фази;

a^i_{switch} – дія перемикання фази;

$a^i_{select,m}$ – вибір m-ї допустимої фази;

a^i_{extend} – дія зміни або продовження зеленого сигналу;

a^i_{plan} – вибір одного з попередньо визначених планів керування.

Найпростішим варіантом є бінарний простір дій, у якому агент на кожному кроці обирає між збереженням поточної фази та переходом до наступної допустимої фази. Така постановка особливо придатна для першої реалізації системи, оскільки вона зменшує кількість варіантів, які має дослідити агент під час навчання. Для DQN це означає меншу кількість вихідних Q-значень, а для PPO або Actor-Critic підходів – простіший розподіл політики. Крім того, бінарна дія краще узгоджується з технічною

логікою світлофорного контролера: агент не створює нову фазову схему, а лише вирішує, чи достатньо довго вже обслуговується поточний напрямок. Водночас така простота має обмеження. Якщо фази завжди перемикаються лише в заздалегідь заданому порядку, агент не може безпосередньо пропустити малозавантажену фазу або швидко перейти до напрямку з критичною чергою. Тому бінарний простір дій доцільно розглядати як базовий етап, придатний для перевірки навчального контуру, а не як остаточну форму адаптивного керування.

Для першої програмної реалізації доцільно обмежити множину дій таким чином:

$$A_0^i = \{0,1\}, \quad 0 \equiv a_{hold}^i, \quad 1 \equiv a_{switch}^i, \tag{2.29}$$

де A_0^i – базовий дискретний простір дій і-го агента;

0 – збереження поточної фази;

1 – перемикання на наступну допустиму фазу після виконання часових і безпекових обмежень.

Практичне значення цієї формули полягає в тому, що вона відокремлює рішення навчального агента від низькорівневої світлофорної логіки. Якщо агент обирає дію перемикання, це не означає миттєвого відкриття нового конфліктного напрямку. Спочатку контролер має перевірити, чи виконано мінімальну тривалість поточної зеленої фази, після чого відпрацьовуються жовтий сигнал і міжзелений інтервал. Лише після цього активується наступна дозволена фаза. Такий поділ є важливим для безпеки та для коректності симуляційного експерименту: модель підкріплювального навчання приймає стратегічне рішення, а середовище або керуючий модуль забезпечує дотримання транспортно-інженерних правил. У середовищі SUMO ця логіка може бути реалізована через керування фазами світлофора на дискретних кроках моделювання, де агент

передає команду, а симулятор оновлює стан транспортного потоку й повертає показники для розрахунку винагороди [8].

Складніший варіант простору дій передбачає прямий вибір наступної фази з множини допустимих фаз. У такому разі агент може гнучкіше реагувати на нерівномірний попит: наприклад, не обслуговувати фазу з майже порожнім підходом і швидше надати зелений сигнал напрямку з довгою чергою. Однак така гнучкість збільшує ризик нестабільного керування. Якщо агент часто змінює фази, транспортний потік втрачає ритм, а частка часу, що витрачається на перехідні інтервали, зростає. Крім того, у мультиагентній мережі вибір фази одним агентом може змінити прибуття транспортних засобів до сусіднього перехрестя, тому простір дій повинен підтримувати не тільки локальну адаптивність, а й можливість координації між вузлами [16].

На рисунку 2.8 наведена схема формування простору дій світлофорного агента.

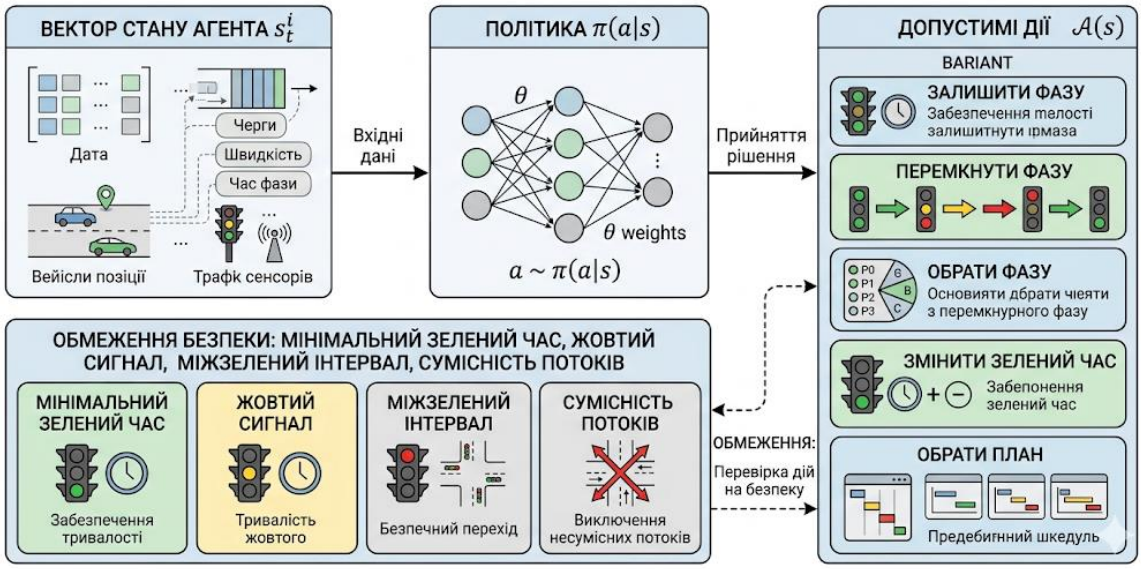


Рисунок 2.8 – Схема формування простору дій світлофорного агента

Керування тривалістю зеленого сигналу може бути подане як параметризована дія. У цьому випадку агент не лише визначає, чи потрібно

залишати фазу активною, а й обирає величину її продовження в межах заданих обмежень. Такий підхід ближчий до реального адаптивного регулювання, оскільки тривалість зеленого часу є одним із головних ресурсів обслуговування транспортного попиту. Проте для початкового етапу дослідження він створює додаткову складність: агент має навчатися не тільки вибору якісно різних фазових рішень, а й кількісного налаштування часу. Якщо тривалість зеленого сигналу задавати неперервно, можуть знадобитися алгоритми, краще пристосовані до неперервних або змішаних просторів дій. Якщо ж час дискретизувати, наприклад на кроки продовження по кілька секунд, простір дій залишається зручним для DQN, але збільшується кількість варіантів, які потрібно дослідити під час навчання [7].

Окремий клас дій становить вибір попередньо визначеного плану керування. У цьому випадку агент не керує кожною фазою окремо, а перемикається між готовими режимами, наприклад планом для низького навантаження, планом для магістрального потоку або планом для перевантаження другорядного напрямку. Така постановка зменшує ризик технічно некоректної поведінки, оскільки кожен план заздалегідь перевіряється з погляду фазової сумісності. Водночас вона обмежує адаптивність: агент може обирати лише між наявними сценаріями, а не формувати точну реакцію на нетипову ситуацію. Для цієї роботи такий варіант доцільно розглядати як проміжний між класичним фіксованим керуванням і повністю адаптивною політикою навчання.

2.10 Проєктування функції винагороди

Після визначення простору станів і множини допустимих дій необхідно сформувати функцію винагороди, яка перетворює наслідки керування на числову оцінку для агента. У задачі світлофорного регулювання ця функція фактично задає критерій навчання: саме вона

визначає, які дії вважатимуться корисними, а які – небажаними. Тому винагорода не повинна бути випадковим набором показників. Вона має відображати мету оптимізації міських транспортних потоків: зменшення затримок і черг, підтримання пропускної здатності, уникнення частих зупинок та запобігання передаванню перевантаження на сусідні перехрестя.

У підкріплювальному навчанні агент не отримує прямого пояснення, чому певна дія є правильною. Він поступово встановлює зв’язок між станом, дією та отриманою винагородою. Через це невдало побудована функція винагороди може сформувати поведінку, яка формально максимізує числовий критерій, але не відповідає транспортній логіці. Наприклад, якщо винагорода враховує лише зменшення черги на активному підході, агент може надмірно продовжувати одну фазу й створювати непропорційні затримки для інших напрямків. Якщо ж оцінювати тільки кількість пропущених транспортних засобів, система може надавати перевагу потужному магістральному потоку, погіршуючи умови для другорядних підходів. У контексті глибокого підкріплювального навчання для транспортних систем вибір винагороди розглядається як один із ключових чинників стабільності й практичної змістовності навченої політики [7].

Для світлофорного агента доцільно використовувати не один показник, а зважену комбінацію кількох компонентів. Основними штрафними складовими можуть бути середня затримка транспортних засобів, довжина черги, кількість зупинок і штраф за часте перемикання фаз. Додатково може враховуватися пропуск транспортних засобів через перехрестя, який має позитивний внесок у винагороду. Таке поєднання дає змогу уникнути надто вузької оптимізації, коли агент покращує один локальний показник за рахунок погіршення інших характеристик руху.

Оскільки затримка, черга, кількість зупинок і пропускна здатність мають різні одиниці вимірювання та діапазони значень, перед об’єднанням їх доцільно нормалізувати. Без нормалізації один показник може домінувати у винагороді лише через більший числовий масштаб, а не через більшу

значущість для задачі керування. Нормалізоване значення окремого показника можна подати так:

$$\tilde{x}_t^i = \frac{x_t^i}{x_{ref}^i + \varepsilon}, \tag{2.30}$$

де $\tilde{x}_t^i$ – нормалізоване значення показника для і-го агента на кроці t;

x_t^i – фактичне значення показника;

x_{ref}^i – опорне або граничне значення для відповідного перехрестя;

ε – мале додатне число, що запобігає діленню на нуль.

З урахуванням наведеного підходу локальну винагороду і-го світлофорного агента можна сформувати як зважену суму штрафів і корисного ефекту від пропуску транспортних засобів:

$$r_t^i = -\alpha_d \tilde{d}_t^i - \alpha_q \tilde{q}_t^i - \alpha_s \tilde{z}_t^i - \alpha_f f_t^i + \alpha_p \tilde{p}_t^i\,, \tag{2.31}$$

де r_t^i – локальна винагорода і-го агента;

$\tilde{d}_t^i$ – нормалізована середня затримка;

$\tilde{q}_t^i$ – нормалізована довжина черги;

$\tilde{z}_t^i$ – нормалізована кількість зупинок;

f_t^i – ознака або інтенсивність перемикання фази;

$\tilde{p}_t^i$ – нормалізована кількість транспортних засобів, пропущених через перехрестя;

α_d, α_q, α_s, α_f, α_p – вагові коефіцієнти відповідних компонентів.

Таблиця 2.8 відображає основні компоненти функції винагороди світлофорного агента, які можуть використовуватися для врахування затримок, довжини черг, кількості зупинок транспортних засобів та загальної ефективності роботи перехрестя.

Таблиця 2.8 – Компоненти функції винагороди світлофорного агента

Компонент винагороди	Зміст показника	Очікуваний вплив на поведінку агента
Середня затримка	Час, втрачений транспортними засобами порівняно з вільним рухом	Стимулює скорочення очікування на підходах
Довжина черги	Кількість транспортних засобів, що очікують обслуговування	Запобігає накопиченню залишкових черг
Кількість зупинок	Частота переходу транспортних засобів у стан зупинки	Сприяє плавнішому руху та меншій фрагментації потоку
Пропуск транспортних засобів	Кількість транспортних засобів, що пройшли через перехрестя	Заохочує ефективне використання зеленого часу
Часте перемикання фаз	Надмірна кількість переходів між фазами	Обмежує втрати часу на жовті та міжзелені інтервали

Штраф за перемикання фаз має окреме методичне значення. Без такого обмеження агент може навчитися часто змінювати фази, реагуючи на короткочасні коливання черги. На перший погляд це може зменшувати локальні затримки, однак у реальному світлофорному керуванні кожне перемикання супроводжується жовтим сигналом, міжзеленим інтервалом і втратою частини ефективного часу обслуговування. Тому функція винагороди повинна не тільки стимулювати швидку реакцію на попит, а й підтримувати стабільність фазової структури. Це узгоджується з попереднім обмеженням простору дій: агент приймає рішення в межах допустимої світлофорної логіки, а винагорода додатково спрямовує його до транспортно доцільної частоти перемикань.

У мультиагентній системі локальної винагороди недостатньо, оскільки дія одного перехрестя впливає на стан сусідніх вузлів. Якщо агент отримує позитивну оцінку лише за очищення власної черги, він може випустити транспортний пакет у напрямку, де наступна ділянка вже перевантажена. Для врахування такого ефекту до локальної винагороди

доцільно додати мережеву складову, яка відображає стан суміжних перехресть або вихідних ділянок. Узагальнено це можна записати так:

$$R_t^i = \lambda r_t^i + (1-\lambda)\frac{1}{|N_i|}\sum_{j\in N_i} r_t^j, \tag{2.32}$$

де R_t^i – підсумкова винагорода і-го агента;

r_t^i – локальна винагорода і-го агента;

N_i – множина сусідніх агентів;

r_t^j – локальна винагорода j-го сусіднього агента;

λ – коефіцієнт балансу між локальною та сусідньою складовими.

Ця формула відображає компроміс між автономністю окремого світлофорного агента та координацією в межах транспортної мережі. Якщо λ наближається до одиниці, агент майже повністю орієнтується на власне перехрестя. Такий варіант простіший для навчання, але може погіршувати мережеву узгодженість. Якщо λ занадто малий, агент починає сильно залежати від стану сусідів, через що складніше визначити внесок його власної дії в отриманий результат. Для мультиагентного керування світлофорами більш виправданим є проміжний варіант, у якому локальна складова залишається основною, але доповнюється обмеженим урахуванням найближчих вузлів. Такий підхід відповідає ідеї масштабованого мультиагентного навчання, де координація досягається не повною централізацією, а врахуванням локально-мережевих залежностей [16].

У межах цієї роботи доцільно обрати комбіновану функцію винагороди, у якій основний акцент робиться на зменшенні затримок і довжин черг, а пропуск транспортних засобів використовується як позитивна складова. Кількість зупинок доцільно враховувати як додатковий показник якості руху, оскільки часте зупинення й рушання погіршує плавність транспортного потоку навіть тоді, коли середня черга

залишається прийнятною. Штраф за перемикання фаз потрібний для того, щоб агент не перетворював адаптивність на нестабільну зміну сигналів. У симуляційному середовищі такі показники можуть обчислюватися на кожному кроці моделювання або за фіксований інтервал агрегації, після чого передаватися до алгоритму навчання як числова винагорода [8].

2.11 Метрики оцінювання ефективності світлофорного керування

Оцінювання ефективності світлофорного керування має відокремлювати критерії навчання агента від критеріїв підсумкового аналізу системи. Функція винагороди, визначена у попередньому підрозділі, використовується під час оптимізації поведінки агента, тоді як метрики оцінювання потрібні для об’єктивного порівняння результатів роботи запропонованої мультиагентної системи з базовим режимом світлофорного регулювання. Через це доцільно використовувати не один інтегральний показник, а набір взаємодоповнювальних характеристик, які відображають затримки, черги, швидкість, кількість зупинок і фактичний пропуск транспортних засобів.

У задачі керування міськими транспортними потоками найбільш змістовною метрикою є середня затримка транспортного засобу. Вона показує, скільки часу втрачається порівняно з умовним вільним рухом без вимушеного очікування на світлофорі, у черзі або в зоні перевантаження. Саме затримка часто використовується для оцінювання рівня обслуговування регульованих перехресть, оскільки вона безпосередньо пов’язана з досвідом учасника руху та з ефективністю використання зеленого часу [3]. Для експериментів у четвертому розділі середню затримку доцільно розраховувати окремо для кожного сценарію моделювання, а потім порівнювати між базовим фіксованим керуванням і навченою мультиагентною політикою. Узагальнено цей показник можна подати так:

$$\bar{D} = \frac{1}{N_c}\sum_{i=1}^{N_c}(T_i^{fact} - T_i^{free}), \quad (2.33)$$

де $\bar{D}$ – середня затримка транспортного засобу;

N_c – кількість транспортних засобів, що завершили поїздку або пройшли контрольовану ділянку;

T_i^{fact} – фактичний час руху і-го транспортного засобу;

T_i^{free} – розрахунковий час руху і-го транспортного засобу за умов вільного потоку.

Черги є другим ключовим класом метрик, оскільки вони показують не лише втрати часу, а й накопичення незадоволеного транспортного попиту. Середня довжина черги відображає типовий рівень завантаження підходів, тоді як максимальна довжина черги потрібна для виявлення критичних ситуацій, які можуть не проявлятися у середньому значенні. Наприклад, система може мати прийнятну середню чергу, але періодично створювати довге накопичення на другорядному напрямку або на вихідній ділянці сусіднього перехрестя. Для мультиагентного керування це особливо важливо, оскільки локально вигідна дія одного агента здатна перенести перевантаження до іншого вузла мережі. Тому в експериментальному аналізі доцільно фіксувати як усереднену, так і пікову чергу:

$$\bar{Q} = \frac{1}{T\cdot|L|}\sum_{t=1}^{T}\sum_{l\in L} q_l(t), \quad (2.34)$$

$$Q_{max} = \max_{t,l} q_l(t), \quad (2.35)$$

де $\bar{Q}$ – середня довжина черги;

Q_{max} – максимальна довжина черги;

T – кількість кроків моделювання;

L – множина контрольованих смуг або підходів;

$q_l(t)$ – довжина черги на смузі або підході l у момент часу t.

Окремого врахування потребує середній час поїздки. На відміну від затримки, він характеризує повну тривалість переміщення транспортного засобу від точки входу до точки виходу із модельованої мережі. Ця метрика корисна тоді, коли оцінюється не лише робота одного перехрестя, а й узгодженість керування в мережі. Якщо мультиагентна система зменшує локальні черги, але збільшує час проходження маршруту через часті зупинки або перенесення заторів, такий результат не можна вважати однозначно кращим. Тому середній час поїздки має використовуватися разом із локальними метриками черг і затримок.

Кількість зупинок доповнює аналіз, оскільки вона відображає плавність руху. Два режими керування можуть мати близьку середню затримку, але суттєво відрізнятися за кількістю вимушених зупинок: у першому випадку транспортний засіб рухається повільніше, але без повної зупинки, а в другому кілька разів зупиняється перед різними перехрестями. Для міського руху така різниця має практичне значення, оскільки часті зупинки погіршують стабільність потоку, збільшують дискомфорт водіїв і можуть спричиняти вторинне накопичення черг. У симуляційному середовищі SUMO відповідні показники можуть збиратися на рівні окремих транспортних засобів, смуг, ребер мережі або завершених поїздок, що робить їх придатними для подальшого порівняльного аналізу [8].

Середня швидкість використовується як агрегований індикатор загального режиму руху. Вона не повинна розглядатися ізольовано, оскільки висока середня швидкість може бути наслідком малої кількості транспортних засобів у мережі або нерівномірного обслуговування окремих напрямків. Проте у поєднанні з чергами, затримками та кількістю завершених поїздок цей показник допомагає зрозуміти, чи справді система підтримує більш рухомий потік, а не лише перерозподіляє очікування між підходами.

$$\bar{V} = \frac{1}{N_o}\sum_{i=1}^{N_o} v_i, \tag{2.36}$$

де $\bar{V}$ – середня швидкість транспортних засобів;

N_o – кількість спостережень швидкості;

v_i – швидкість транспортного засобу або агрегованого транспортного потоку в і-му спостереженні.

Для порівняння базового та запропонованого підходів також необхідно враховувати пропускну здатність і кількість завершених поїздок. Пропускна здатність показує, скільки транспортних засобів система фактично здатна обслужити за одиницю часу, а кількість завершених поїздок фіксує, скільки маршрутів було повністю виконано протягом експериментального інтервалу. Ці показники особливо важливі в перевантажених сценаріях: інтелектуальне керування може зменшувати середню затримку для частини транспортних засобів, але якщо водночас менша кількість автомобілів завершує поїздку, ефективність такого підходу потребує обережної інтерпретації. Узагальнено пропуск транспортних засобів можна визначити так:

$$P = \frac{N_{out}}{T_{exp}}, \tag{2.37}$$

де P – пропуск транспортних засобів за одиницю часу;

N_{out} – кількість транспортних засобів, що залишили контрольовану мережу або пройшли заданий переріз;

T_{exp} – тривалість експерименту.

Таблиця 2.9 подає основні метрики оцінювання ефективності світлофорного керування, що можуть застосовуватися для порівняння різних підходів до управління та визначення якості функціонування світлофорного агента.

Таблиця 2.9 – Метрики оцінювання ефективності світлофорного керування

Метрика	Зміст показника	Очікуваний напрям покращення	Роль у порівнянні підходів
Середня затримка	Середній втрачений час транспортного засобу	Зменшення	Основний показник якості обслуговування
Середня довжина черги	Типовий рівень накопичення транспортних засобів на підходах	Зменшення	Оцінює стабільність обслуговування попиту
Максимальна довжина черги	Найбільше зафіксоване накопичення на смузі або підході	Зменшення	Виявляє критичні локальні перевантаження
Середній час поїздки	Повна тривалість проходження маршруту	Зменшення	Показує мережевий ефект керування
Кількість зупинок	Частота вимушених повних зупинок	Зменшення	Характеризує плавність транспортного потоку
Середня швидкість	Усереднена швидкість руху в мережі	Збільшення	Допомагає оцінити загальний режим руху
Пропускна здатність	Кількість транспортних засобів, обслужених за одиницю часу	Збільшення	Відображає використання ресурсу мережі
Кількість завершених поїздок	Кількість транспортних засобів, що досягли пункту призначення	Збільшення	Показує результативність роботи системи за експериментальний період

2.12 Обмеження застосування методів підкріплювального навчання в реальних транспортних системах

Методи підкріплювального та мультиагентного навчання створюють методичну основу для адаптивного світлофорного керування, однак їх

практичне застосування не може оцінюватися лише за результатами симуляційного експерименту. У попередніх підрозділах було визначено простір станів, дій, функцію винагороди та метрики оцінювання, проте реальна транспортна система містить додаткові обмеження: неповні спостереження, технічні вимоги контролерів, вимоги безпеки, нестабільність попиту та відмінність поведінки водіїв від моделі, закладеної в симуляторі [7].

Одним із головних ризиків є розрив між симуляційним і реальним середовищем. У SUMO або іншому мікроскопічному симуляторі можна точно зчитувати положення транспортних засобів, довжини черг, швидкості, затримки й завершені поїздки, тоді як у реальній мережі такі дані отримуються через детектори, камери, індукційні петлі, GPS-треки або агреговані телематичні джерела. Кожен із цих каналів має власну похибку, затримку та зону видимості. Якщо агент навчався на повному й точному векторі стану, а під час практичного застосування отримує лише часткові або зашумлені дані, політика може неправильно оцінювати транспортну ситуацію. Наприклад, коротка черга біля стоп-лінії не завжди означає низький попит: частина транспортних засобів може бути заблокована вище за потоком або не потрапляти в зону детектування. Через це результати моделювання слід трактувати як перевірку працездатності підходу в контрольованих умовах, а не як безпосереднє підтвердження готовності системи до впровадження.

Для кількісного опису відмінності між симуляційним і реальним середовищем доцільно використовувати показник розбіжності метрик, який може застосовуватися під час калібрування моделі:

$$\Delta_m = \frac{|m_{sim} - m_{real}|}{m_{real} + \varepsilon}, \tag{2.38}$$

де Δ_m – відносна розбіжність для метрики m;

m_{sim} – значення метрики, отримане в симуляційному середовищі;

m_{real} – відповідне значення, виміряне або оцінене в реальній транспортній системі;

ε – мала додатна величина, що запобігає діленню на нуль.

Безпека є окремим класом обмежень, який має вищий пріоритет, ніж оптимізація винагороди. Агент не повинен мати можливості сформувати керуючу дію, що порушує конфліктність фаз, мінімальні тривалості зеленого сигналу, жовтий сигнал, міжзелений інтервал або правила обслуговування пішохідних напрямків [1]. Це означає, що політика підкріплювального навчання має працювати не як безпосередній генератор сигналів, а як надбудова над безпечним світлофорним контролером. Такий контролер обмежує простір допустимих дій і відхиляє рішення, які не відповідають транспортно-інженерним правилам. У симуляції подібні обмеження часто здаються технічною деталлю, але для реальної системи вони визначають межу між дослідницькою моделлю та потенційно придатним до експлуатації алгоритмом.

Ще одним обмеженням є стабільність навченої політики. Для DQN, PPO та мультиагентних Actor-Critic-підходів характерна чутливість до складу стану, параметрів винагороди, випадкової ініціалізації, сценаріїв попиту та гіперпараметрів навчання. Політика, яка демонструє прийнятні середні значення затримки, може водночас створювати небажані локальні ефекти: часті перемикання фаз, нерівномірне обслуговування другорядних напрямків або періодичне перенесення перевантаження до сусіднього вузла. У мультиагентній системі ця проблема посилюється, оскільки середовище для кожного агента змінюється не тільки через транспортний попит, а й через дії інших агентів [16]. Тому під час інтерпретації експериментів потрібно аналізувати не лише середнє покращення, а й розкид результатів, максимальні черги, кількість зупинок і поведінку системи в напружених сценаріях.

Інтерпретованість рішень має не лише наукове, а й експлуатаційне значення. У традиційному світлофорному регулюванні інженер може

пояснити зміну плану через інтенсивність потоку, рівень насичення або необхідність обслуговування певного напрямку. Нейронна політика, навпаки, часто подає рішення як результат внутрішньої апроксимації, що ускладнює аудит. Для транспортної системи це проблемно, оскільки рішення впливають на безпеку, затримки та розподіл пріоритетів між учасниками руху. Тому під час розробки мультиагентної системи доцільно передбачати збереження журналів станів, дій, винагород і ключових метрик. Це дозволяє аналізувати, чому агент обрав певну фазу, чи не виникло надмірне перемикання та чи відповідає поведінка моделі транспортній логіці.

2.13 Висновки до розділу

У результаті теоретичного дослідження було обґрунтовано доцільність застосування методів штучного інтелекту для розв’язання задачі світлофорного регулювання. Світлофорний контролер розглянуто як агент, який спостерігає стан транспортного середовища, обирає допустиму дію та отримує винагороду залежно від зміни черг, затримок, пропуску транспортних засобів і стабільності руху.

Задачу формалізовано як марковський процес прийняття рішень, а для міської мережі – як мультиагентну систему, у якій кожне регульоване перехрестя може виступати окремим агентом.

У межах дослідження розглянуто методи Q-навчання, DQN, PPO та Actor-Critic. Також визначено основні елементи майбутньої моделі: простір станів, простір дій, функцію винагороди та метрики оцінювання. До ключових метрик віднесено середню затримку, середню й максимальну довжину черги, середній час поїздки, кількість зупинок, середню швидкість, пропускну здатність і кількість завершених поїздок.

3 РОЗРОБКА МЕТОДОЛОГІЇ ТА АРХІТЕКТУРИ МУЛЬТИАГЕНТНОЇ СИСТЕМИ НАВЧАННЯ СВІТЛОФОРНОГО РЕГУЛЮВАННЯ

3.1 Обмеження застосування методів підкріплювального навчання в реальних транспортних системах

Перехід від теоретичної постановки задачі до розробки мультиагентної системи світлофорного керування потребує не повторного опису методів підкріплювального навчання, а визначення того, як саме ці методи можуть бути вбудовані в керований транспортний контур.

У межах цієї роботи регульоване перехрестя розглядається як агент, який отримує інформацію про стан руху, обирає допустиму дію щодо світлофорної фази та отримує винагороду за наслідками цього рішення. Водночас агент не діє ізольовано: його рішення змінює прибуття транспортних засобів до суміжних вузлів, тому вся транспортна мережа подається як спільне симуляційне середовище для групи взаємопов’язаних агентів [16].

Фрагменти основних модулів програми подані в додатку А.

У реалізаційному аспекті основним обмеженням є те, що алгоритм навчання не може безпосередньо керувати всіма сигналами світлофора як довільними змінними. Світлофорний об’єкт має наперед визначену фазову структуру, мінімальні тривалості зеленого сигналу, жовтий сигнал, міжзелений інтервал і набір несумісних напрямків руху. Через це в запропонованій системі агент не формує нову світлофорну програму з нуля, а вибирає дію лише з множини безпечних варіантів, дозволених керуючим модулем.

Такий підхід узгоджується з транспортно-інженерною логікою, згідно з якою оптимізація пропуску не може мати вищий пріоритет, ніж безпека конфліктних потоків [1]. Практично це означає, що між політикою

підкріплювального навчання та симулятором повинен бути окремий шар перевірки дій.

У програмній частині відповідна логіка підтверджується класами PhaseSwitchActionSpace і PhaseSelectionActionSpace у файлі actions.py, а також класами TSController і TrafficSignal у файлі traffic_signal.py; фрагменти цього коду подано в додатку А.

Якщо агент пропонує продовжити фазу, модуль контролю перевіряє, чи не перевищено допустиму тривалість. Якщо агент пропонує перемикання, середовище спочатку відпрацьовує перехідні інтервали, а вже потім активує наступну фазу. У такий спосіб транспортна задача перетворюється на задачу вибору серед допустимих керуючих рішень, а не на неконтрольоване генерування сигналів (рисунок 3.1).

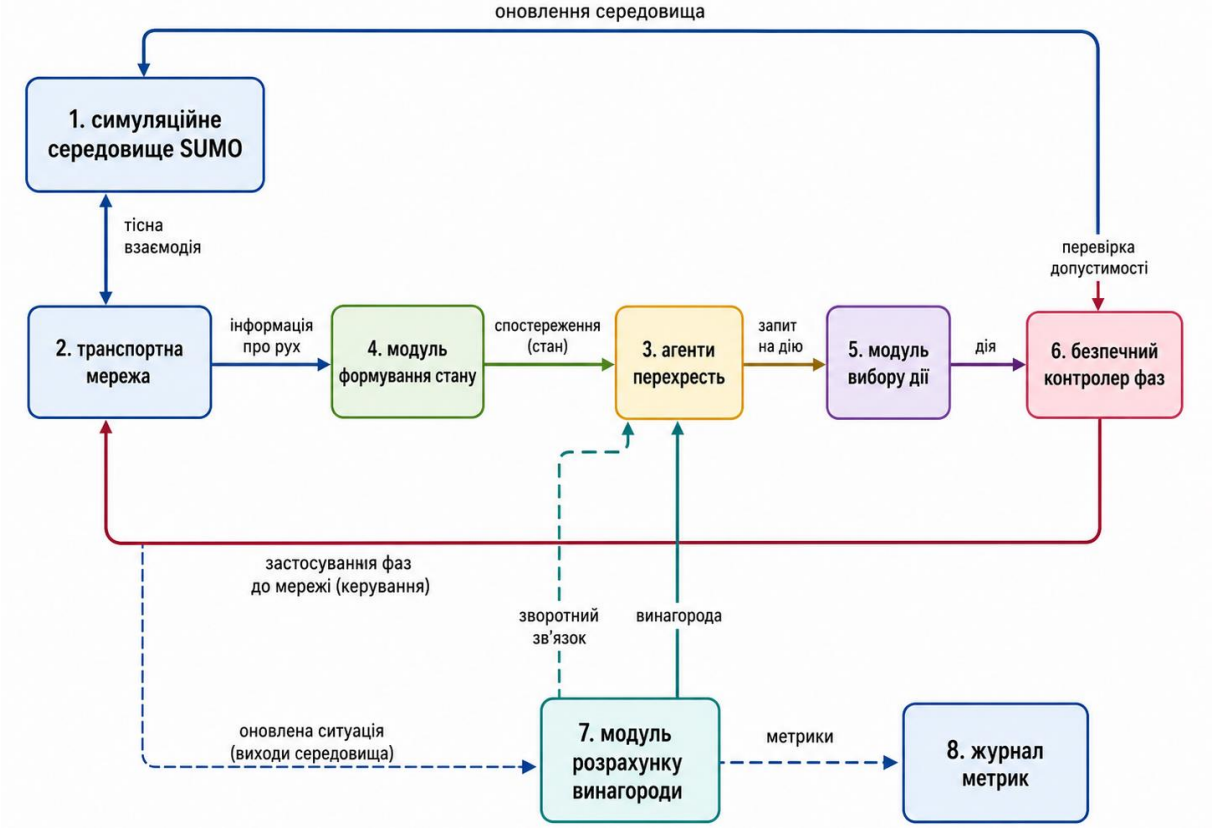


Рисунок 3.1 – Загальна архітектура мультиагентної системи навчання світлофорного регулювання

Стан агента в такій архітектурі формується не як повний опис усієї мережі, а як структурований набір локальних і обмежених мережевих ознак. До локальної частини належать довжини черг на підходах, час очікування, середні швидкості, зайнятість смуг, поточна фаза та тривалість її дії. Мережева частина може містити агреговану інформацію про сусідні перехрестя: наявність черги на вихідній ділянці, завантаженість приймального напрямку або ознаку ризику блокування.

Реалізаційною основою такого подання є простори спостережень PositionMatrix і LaneFeatures у файлі observations.py, а також метод update_stats() класу TrafficSignal у файлі traffic_signal.py; відповідні фрагменти подано в додатку А.

Таке обмеження стану є принциповим. Якщо агент отримуватиме надто багато змінних, модель стане складнішою для навчання, а причинний зв’язок між дією і винагородою послабиться. Якщо ж стан буде надто вузьким, агент зможе реагувати лише на власну чергу й не враховуватиме, що його рішення може передати перевантаження до сусіднього вузла.

Тому в цій роботі доцільною є проміжна архітектура: кожен агент діє децентралізовано, але його спостереження доповнюється стислим описом найближчого транспортного околу.

Алгоритм роботи системи на одному кроці моделювання можна подати так:

1) симуляційне середовище оновлює положення транспортних засобів, швидкості, черги та стан світлофорних об’єктів;

2) для кожного регульованого перехрестя формується вектор спостереження, що містить локальні показники руху та агреговані ознаки сусідніх вузлів;

3) світлофорний агент на основі поточного спостереження обирає дію: збереження активної фази, перехід до наступної фази або вибір іншої допустимої фази залежно від обраної моделі простору дій;

4) модуль безпечного контролю перевіряє дію за обмеженнями фазової сумісності, мінімального зеленого часу, жовтого сигналу та міжзеленого інтервалу;

5) допустима дія передається до симулятора, після чого транспортна мережа переходить у новий стан;

6) для кожного агента обчислюється винагорода на основі зміни черг, затримок, пропуску транспортних засобів, кількості зупинок і можливого впливу на сусідні перехрестя;

7) стан, дія, винагорода та наступний стан зберігаються для навчання політики й подальшого аналізу метрик.

У наведеному алгоритмі важливо, що навчальний агент не замінює всю світлофорну логіку. Він виконує роль адаптивного рівня прийняття рішень, який намагається вибрати кращий момент для продовження або зміни фази. Нижчий рівень залишається відповідальним за технічну коректність виконання.

Такий поділ зменшує ризик нестійкої поведінки, оскільки навіть у разі невдалого рішення агента система не виходить за межі допустимих фазових переходів.

Для симуляційного середовища SUMO це особливо зручно, оскільки модель руху, маршрути транспортних засобів, фазові програми та показники ефективності можуть бути відтворені в контрольованих умовах, а зовнішній керуючий модуль може покроково передавати команди світлофорним об’єктам [8].

Взаємодія із SUMO через TraCI реалізована у файлах simulator.py, retriever.py і traffic_signal.py, фрагменти яких подано в додатку А.

Рисунок 3.2 представляє послідовність блоків перетворення транспортної задачі на задачу мультиагентного навчання.

Рисунок 3.3 показує, що сусідні агенти обмінюються не повним станом, а агрегованими мережевими ознаками.



Рисунок 3.2 – Цикл перетворення транспортної задачі на задачу мультиагентного навчання



Рисунок 3.3 – Агреговані мережеві ознаками обміну між сусідніми агентами

Ще одне суттєве обмеження пов’язане з одночасною дією кількох агентів. Якщо кожне перехрестя навчається незалежно, середовище для нього стає нестаціонарним: зміна політики сусіднього агента змінює потік прибуття, а отже, і наслідки власних дій. У цій роботі така проблема враховується через обмежену координацію між агентами. Вона не передбачає повної централізації керування всією мережею, оскільки такий підхід швидко збільшує розмірність простору дій і ускладнює масштабування.

Натомість кожен агент залишається локальним контролером свого перехрестя, але під час формування стану й винагороди враховує найближчі мережеві наслідки: чи не блокується вихідна ділянка, чи не накопичується черга в суміжному вузлі, чи не передається надлишковий транспортний пакет на вже перевантажений напрямок. Це дозволяє зберегти автономність агентів і водночас уникнути повністю егоїстичної локальної поведінки.

3.2 Архітектура мультиагентної системи

Архітектура мультиагентної системи світлофорного керування в цій роботі розглядається як програмно-алгоритмічний контур, що поєднує симуляційне середовище, агентів регульованих перехресть, модулі обробки транспортних даних, механізм вибору дій, обчислення винагороди, навчання політики, оцінювання результатів і керування експериментальними запусками.

На відміну від загальної постановки мультиагентного навчання, тут архітектура має бути прив’язана до конкретної транспортної логіки: кожен агент відповідає не за абстрактний керуючий об’єкт, а за світлофорний контролер окремого перехрестя, дії якого змінюють черги, затримки та прибуття транспортних засобів до суміжних вузлів.

Загальна структура програмного проєкту підтверджується директорією , у якій виділено піддиректорії common, backends/sumo,

controllers, wrappers, scenarios і training. Окремий прикладний рівень утворюють файли run_once.py, train_marl_dqn.py, train_marl.py, plot_training_metrics.py і analyze_tripinfo.py; фрагменти структури подано в додатках А і Д.

Основою системи є симуляційне середовище SUMO, яке виконує роль динамічної моделі міської транспортної мережі. У ньому задаються дорожня топологія, маршрути транспортних засобів, світлофорні програми, часовий крок моделювання та показники руху, необхідні для навчання агентів [8].

Для базового сценарію ці матеріали розміщені в директорії 2x2_sumo_grid: файл 2x2_sumo_grid.net.xml описує мережу, 2x2_sumo_grid.rou.xml і 2x2_sumo_grid.rou.alt.xml задають маршрути, 2x2_sumo_grid.add.xml містить додаткові детектори, 2x2_sumo_grid.sumocfg визначає запуск симуляції, а config.yaml фіксує параметри середовища; конфігураційні параметри подано в додатку Б, а файли мережі, маршрутів і детекторів – у додатку В.

Зовнішній керуючий контур не замінює внутрішню модель руху симулятора, а взаємодіє з нею покроково: отримує поточні показники, формує стани агентів, передає допустимі дії світлофорним об’єктам і після оновлення середовища збирає нові спостереження.

Така організація дає змогу відокремити транспортну динаміку від алгоритму навчання. Симулятор відповідає за переміщення транспортних засобів, виникнення черг і зміну швидкостей, тоді як мультиагентна система відповідає за прийняття рішень щодо фазового керування.

Це важливо для відтворюваності експериментів: одна й та сама архітектура може запускатися на різних сценаріях попиту, але логіка взаємодії модулів залишається незмінною.

Загальну структуру програмного контуру подана як рисунок 3.4.

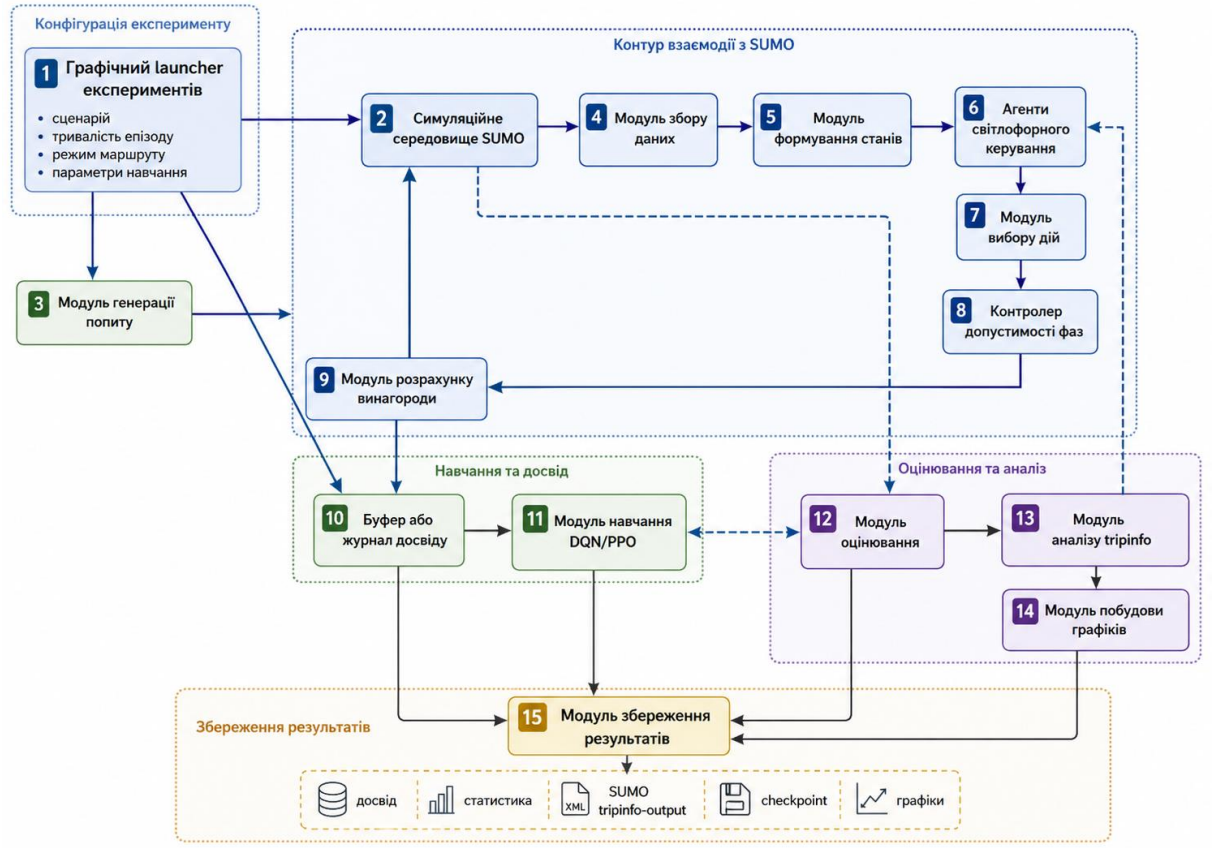


Рисунок 3.4 – Модульна архітектура мультиагентної системи світлофорного керування

У наявних матеріалах проєкту модулі запуску, оцінювання та збереження статистики підтверджуються файлами run_once_gui.py, evaluate.py, run_controllers.py, tripinfo.py і plots.py, а фрагменти журналів і CSV-результатів подано в додатках Г і Д.

Модуль збору даних є проміжним шаром між SUMO та навчальною частиною системи. Його призначення полягає не лише у зчитуванні доступних змінних, а й у перетворенні сирих симуляційних даних на показники, придатні для подальшої обробки. До таких даних належать довжини черг на підходах, швидкості транспортних засобів, час очікування, зайнятість смуг, поточні фази світлофорів, тривалість активної фази та кількість транспортних засобів, що пройшли через контрольовані ділянки. У проєкті зчитування та підписка на змінні SUMO реалізовані класом

Retriever у файлі retriever.py, а агрегування крокових показників виконується класом MetricsParser у файлі metrics.py; ці фрагменти належать до додатка А. Якщо ці показники передавати агентам без структурування, модель навчання буде залежати від технічного формату симулятора, а не від змісту транспортної ситуації. Тому після збору даних виконується їх агрегування за підходами, смугами або перехрестями, а також підготовка до нормалізації відповідно до ознак, визначених у другому розділі.

Модуль формування станів створює для кожного агента локальне спостереження, яке відображає транспортну ситуацію на відповідному перехресті та обмежений стан його найближчого околу. У цій роботі не передбачається передавання кожному агенту повного глобального стану всієї мережі, оскільки такий підхід збільшує розмірність вхідних даних і погіршує масштабованість.

Натомість агент отримує детальний опис власного вузла та стислий набір ознак сусідніх перехресть, наприклад зайнятість вихідної ділянки або наявність черги нижче за потоком. Це узгоджується з попередньо визначеною логікою локально-мережевого стану: агент зберігає автономність, але не приймає рішення в повній ізоляції. Практично такий модуль виконує роль фільтра, який відкидає надлишкові симуляційні змінні й залишає ті характеристики, що мають безпосередній керуючий зміст; програмна логіка формування спостережень розміщена у файлі observations.py, фрагмент якого подано в додатку А.

Центральним елементом архітектури є множина світлофорних агентів. Кожен агент пов’язаний з одним регульованим перехрестям і використовує спільний або індивідуальний механізм вибору дій залежно від обраного варіанта реалізації. У наявній програмній частині така логіка подана класами TSCAgent, TSCGraphAgent, TSCAgentEnsemble і RLController у файлі rl_controller.py, а збережена модель базового агента міститься у файлі agent.th; ці матеріали подано в додатку А.

У базовому варіанті доцільно передбачити спільну структуру політики для однотипних перехресть, оскільки це зменшує кількість параметрів і дає змогу агентам використовувати подібні закономірності керування. Водночас стан кожного агента залишається локальним, тому рішення формується з урахуванням конкретної ситуації на відповідному вузлі. Якщо мережа містить перехрестя з різною фазовою структурою, архітектура повинна підтримувати маскування недопустимих дій або окремі вихідні простори для різних типів світлофорних об’єктів. Це запобігає ситуації, коли модель пропонує фазу, якої фізично немає в конкретному контролері.

Модуль вибору дій перетворює вихід політики на керуюче рішення. Для задачі світлофорного регулювання дія може означати продовження поточної фази, перехід до наступної допустимої фази або вибір однієї з дозволених фаз залежно від реалізованого простору дій. Проте це рішення не передається до симулятора безпосередньо.

Між агентом і середовищем розміщується контролер допустимості фаз, який перевіряє мінімальну тривалість зеленого сигналу, жовтий сигнал, міжзелений інтервал і сумісність напрямків. Простори дій PhaseSwitchActionSpace, PhaseSelectionActionSpace і CentralizedActionSpace реалізовані у файлі actions.py, а виконання фазових переходів через TraCI подано у файлі traffic_signal.py; фрагменти цих модулів подано в додатку А.

Такий контролер є обов’язковою частиною архітектури, оскільки політика навчання не повинна мати технічної можливості порушити транспортно-інженерні обмеження. У цьому полягає відмінність реалізаційного рівня від математичної моделі: навіть якщо агент обирає дію як елемент дискретної множини, фактичне виконання цієї дії проходить через шар безпечного світлофорного керування. Основні модулі системи та їх функції можна узагальнити (таблиця 3.1).

Таблиця 3.1 – Функціональні модулі мультиагентної системи світлофорного керування

Модуль	Призначення в системі	Основний результат роботи
Симуляційне середовище	Відтворення руху транспортних засобів і роботи світлофорів	Оновлений стан транспортної мережі
Модуль збору даних	Отримання черг, швидкостей, затримок, фаз і показників смуг	Первинні транспортні спостереження
Модуль формування станів	Агрегування, нормалізація та структурування ознак агентів	Вектор стану для кожного перехрестя
Модуль вибору дій	Формування рішення агента на основі поточного стану	Запропонована світлофорна дія
Контролер допустимості фаз	Перевірка фазових і часових обмежень	Безпечна дія для передавання в SUMO
Модуль винагороди	Оцінювання наслідків дії за чергами, затримками й пропуском	Числова винагорода агента
Буфер або журнал досвіду	Збереження досвіду взаємодії агента із середовищем	Набір переходів або статистичних записів для навчання й аналізу
Модуль навчання	Оновлення параметрів політики або Q-функції	Навчена модель керування
Модуль оцінювання	Розрахунок метрик ефективності за сценаріями	Порівняльні результати експериментів
Модуль генерації попиту	Формування тимчасових route/sumocfg-файлів для різних значень Traffic Scale	Відтворювані сценарії зі зміненим транспортним навантаженням
Модуль аналізу tripinfo	Узагальнення фактичних завершених поїздок SUMO	Таблиці tripinfo_summary.csv і tripinfo_vehicles.csv
Модуль візуалізації	Побудова графіків навчання, оцінювання, Scale-порівнянь і auto-tuning	Графічні матеріали для аналізу результатів
Модуль збереження	Фіксація моделей, журналів, метрик і конфігурацій	Відтворювані експериментальні дані

Модуль розрахунку винагороди виконує перетворення транспортних наслідків у навчальний сигнал. Він не повинен дублювати всі метрики

оцінювання, оскільки винагорода використовується на кожному кроці навчання, а метрики можуть аналізуватися після завершення епізоду. У запропонованій архітектурі винагорода формується на основі зміни локальних черг, затримок, пропуску транспортних засобів і штрафу за небажані перемикання.

Для мультиагентної постановки до неї може додаватися обмежена мережева складова, пов’язана зі станом сусідніх вузлів. Реалізаційною основою цього компонента є файл reward.py, де подано класи BaseRewardFunction, QueueLength і MaxPressure; фрагмент коду подано в додатку А. Такий підхід не перетворює систему на повністю централізовану, але зменшує ризик егоїстичної поведінки окремого агента, коли локальне очищення підходу створює перевантаження нижче за потоком [16].

Буфер або журнал досвіду використовується для накопичення переходів виду «стан – дія – винагорода – наступний стан», які надалі застосовуються модулем навчання. Для алгоритмів на основі DQN такий буфер є важливим засобом стабілізації навчання, оскільки дає змогу оновлювати модель не лише за останнім спостереженням, а за вибіркою з попереднього досвіду [13]. У доопрацьованій програмній реалізації відповідний компонент реалізовано класом ReplayBuffer у файлі marl_dqn.py, а навчальний контур DQN подано класом MARLDoubleDQNTrainer; фрагменти цього коду подано в додатку Д.

Для PPO аналогічну роль виконує RolloutBuffer у файлі marl_ppo.py, де зберігаються rollout-фрагменти для обчислення переваг і оновлення actor-critic-моделі. Журналювання статистики експериментів підтверджується не лише файлом run_controllers.py, де результати зберігаються у файл stats.csv, а й файлами dqn_train_log.csv, dqn_eval_log.csv, dqn_training_signals.csv, ppo_train_log.csv, ppo_eval_log.csv і ppo_training_signals.csv у директорії Multi-Agent for Traffic Signal Control/training_runs; приклади таких результатів подано в додатках Г і Д. У мультиагентній системі буфер зберігає досвід з урахуванням контексту сценарію: номера епізоду, часової

мітки, інтенсивності попиту, конфігурації мережі та параметрів світлофорного контролера. Без цього подальший аналіз результатів був би неповним, оскільки однакова дія може мати різний ефект у різних транспортних умовах.

Алгоритмічно роботу архітектури під час навчального епізоду можна подати так:

1) ініціалізувати симуляційний сценарій, параметри світлофорних агентів, буфер або журнал досвіду і журнали метрик;

2) для кожного кроку моделювання отримати з SUMO поточні транспортні показники;

3) сформувати локальні стани агентів з урахуванням ознак відповідних перехресть і агрегованих даних сусідніх вузлів;

4) передати стани до модуля вибору дій і отримати запропоновані рішення для кожного світлофорного агента;

5) перевірити запропоновані дії через контролер допустимості фаз;

6) передати безпечні дії до симуляційного середовища та виконати черговий крок моделювання;

7) обчислити винагороди агентів за зміною черг, затримок, пропуску транспортних засобів і мережевих ознак;

8) зберегти переходи в буфер, журнал досвіду або іншу структуру відповідно до обраного алгоритму;

9) оновити параметри моделі навчання за накопиченими переходами;

10) після завершення епізоду розрахувати метрики ефективності, зберегти модель, конфігурацію та журнали результатів.

Окремо в архітектурі виділяється модуль оцінювання. Його завдання полягає в тому, щоб відокремити процес навчання від процесу перевірки якості керування. Під час навчання агент може використовувати механізми дослідження дій, випадкові ініціалізації або оновлення політики, які роблять поведінку нестабільною. Під час оцінювання ці механізми мають бути вимкнені або зафіксовані, щоб результати відображали саме якість навченої

політики. Модуль оцінювання повинен запускати однакові сценарії для базового світлофорного керування та для мультиагентної системи, після чого порівнювати середню затримку, довжини черг, кількість зупинок, пропуск транспортних засобів і завершені поїздки. У програмному проєкті цей компонент пов’язаний із файлами evaluate.py, run_controllers.py і metrics.py; фрагменти коду подано в додатку А, а приклади CSV-вивантажень – у додатку Г. Такий поділ потрібний для коректної інтерпретації експериментів у четвертому розділі.

Модуль збереження результатів забезпечує відтворюваність дослідження. У межах цієї роботи доцільно зберігати не лише підсумкові значення метрик, а й конфігурації сценаріїв, параметри навчання, версії моделей, журнали винагород, історію дій агентів і агреговані показники за епізодами. Наявний скрипт run_controllers.py передбачає збереження підсумкової таблиці stats.csv у директорії результатів, а файл моделі agent.th фіксує ваги базового контролера; приклади таких матеріалів подано в додатку Г.

Це дає змогу пізніше перевірити, за яких умов була отримана певна поведінка системи, і відрізнити стійке покращення від випадкового результату окремого запуску. Для мультиагентної системи така фіксація особливо важлива, оскільки одночасне навчання кількох агентів може призводити до різних траєкторій політики навіть за близьких початкових умов.

3.3 Модель транспортної мережі та її елементи

Модель транспортної мережі в розроблюваній мультиагентній системі визначає межі середовища, у якому світлофорні агенти отримують спостереження, виконують дії та накопичують досвід для подальшого навчання. На відміну від попереднього підрозділу, де було розглянуто загальну архітектуру системи, у цьому підрозділі конкретизується

транспортна основа цієї архітектури: дорожня топологія, регульовані вузли, смуги, маршрути, транспортні потоки, фазова структура світлофорів і детектори, з яких формується інформаційний стан агентів.

У межах роботи транспортна мережа подається як кероване мікроскопічне симуляційне середовище SUMO. Такий вибір зумовлений тим, що SUMO дає змогу описувати рух окремих транспортних засобів, зберігати структуру дорожньої мережі у форматі net.xml, задавати маршрути у файлах rou.xml, підключати додаткові детектори через add.xml і взаємодіяти із симуляцією покроково через зовнішній керуючий контур [8]. Для базового навчального сценарію в проєкті використовується синтетична мережа 2x2_sumo_grid, матеріали якої розміщені в директорії 2x2_sumo_grid і мають бути подані в додатку В. Вона складається з дев’яти регульованих перехресть, організованих у вигляді регулярної міської сітки.

Така структура є достатньо простою для контрольованого аналізу поведінки агентів, але водночас містить ключову властивість міського руху: рішення одного світлофора змінює умови прибуття транспортних засобів до сусідніх вузлів. Окремо в проєкті наявні SUMO-сценарії реальних або наближених до реальних мереж у директоріях cologne і monaco, тому запропонована модель не обмежується лише синтетичним випадком і може бути перенесена на фрагмент реальної міської топології після узгодження параметрів попиту та фазового керування.

Транспортна мережа розглядається як орієнтований граф, у якому вузли відповідають перехрестям або граничним точкам входу й виходу, а ребра відображають дорожні ділянки між ними. У сценарії 2x2_sumo_grid регульованими вузлами є перехрестя з ідентифікаторами A0, A1, A2, B0, B1, B2, C0, C1, C2. Саме ці вузли утворюють множину агентів системи. Ребра задають напрямки руху між сусідніми перехрестями та зовнішніми в’їздами або виїздами мережі, а смуги деталізують пропускну структуру кожного ребра. У файлі 2x2_sumo_grid.net.xml містяться 192 ребра та 360 смуг, що

дає змогу враховувати не лише факт наявності зв’язку між вузлами, а й розподіл транспортних засобів за окремими смугами.

На рисунку 3.5 наведено подання транспортної мережі як середовища взаємодії світлофорних агентів.

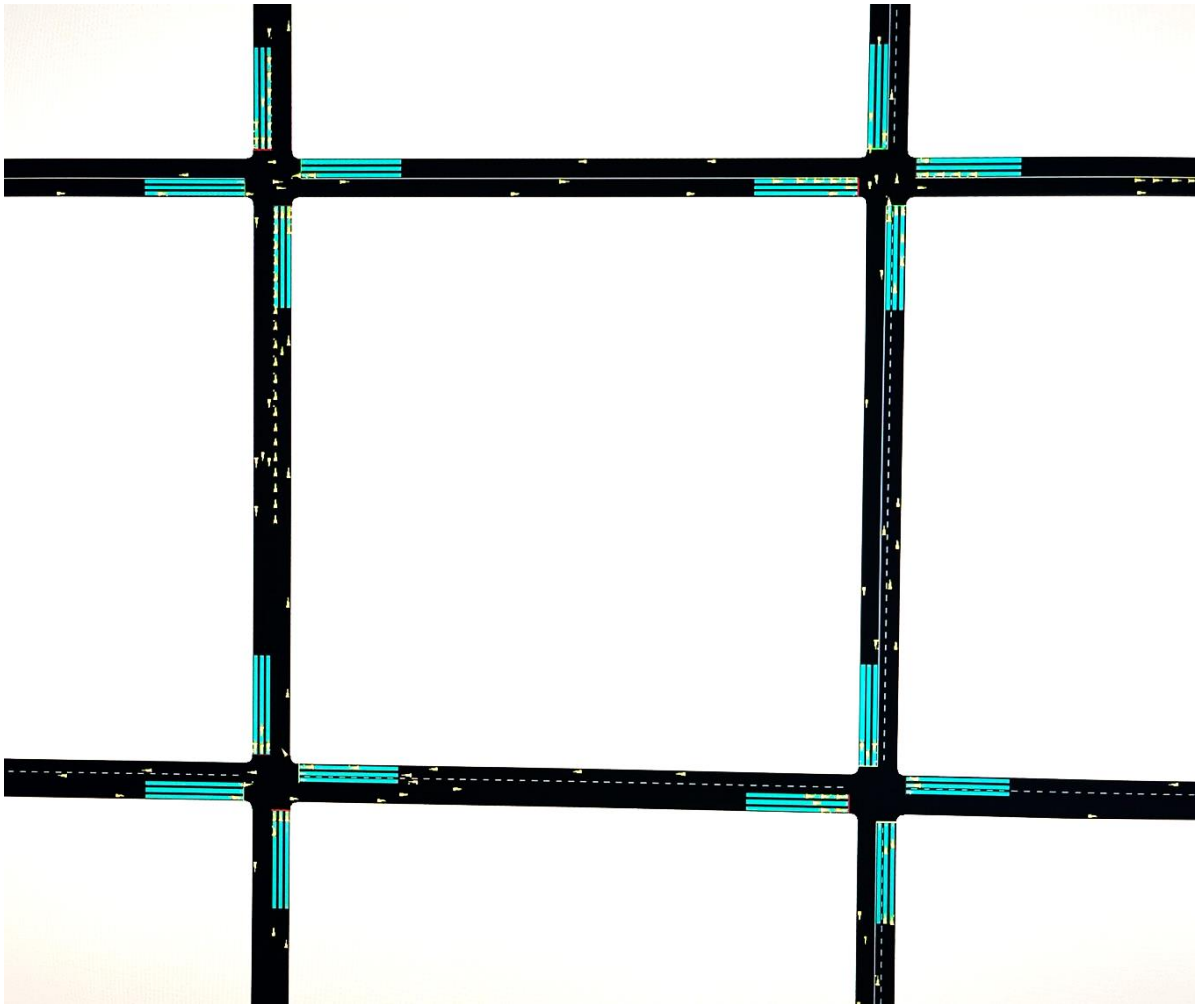


Рисунок 3.5 – Подання транспортної мережі як середовища взаємодії світлофорних агентів

Для задачі навчання це має принципове значення, оскільки черга на одній смузі може не збігатися з навантаженням усього підходу, а фазове рішення світлофора впливає саме на дозволені групи смуг і напрямків. Таблиця 3.2 демонструє основні елементи транспортної мережі.

Таблиця 3.2 – Основні елементи транспортної мережі в мультиагентній системі

Елемент мережі	Подання в симуляційному середовищі	Значення для мультиагентного керування
Вузол	Перехрестя або гранична точка мережі	Визначає місце взаємодії транспортних потоків
Регульоване перехрестя	Світлофорний вузол із фазовою програмою	Відповідає окремому агенту керування
Ребро	Дорожня ділянка між вузлами	Передає транспортний потік між агентами
Смуга	Окремий напрям руху в межах ребра	Використовується для розрахунку черг, швидкості й зайнятості
Маршрут	Послідовність ребер для транспортного засобу	Формує просторову траєкторію руху через мережу
Транспортний потік	Набір транспортних засобів із часами відправлення	Визначає навантаження на мережу протягом епізоду
Світлофорна фаза	Комбінація дозволених і заборонених напрямків	Є об’єктом керуючої дії агента
Детектор	Віртуальний вимірювач на смузі	Забезпечує дані для стану агента та метрик

У базовій мережі кожне регульоване перехрестя має статичну початкову світлофорну програму з двома зеленими фазами та проміжними жовтими фазами. У файлі 2x2_sumo_grid.net.xml для кожного з дев’яти світлофорів задано чотири фази: дві основні фази з тривалістю 42 с і дві жовті фази з тривалістю 3 с. На рівні мультиагентної системи ці програми не розглядаються як остаточне керування, а використовуються як безпечна фазова основа, у межах якої агент може приймати рішення щодо продовження або зміни фази. Конфігурація сценарію у файлі config.yaml додатково задає yellow_time = 5, min_green_time = 5, max_green_time = 60 і режим послідовного проходження фаз round_robin. Це означає, що агент не створює довільну комбінацію сигналів, а працює з уже визначеною множиною допустимих фазових переходів. Такий підхід узгоджується з

вимогою збереження транспортної безпеки: оптимізація затримок і черг не повинна порушувати конфліктну сумісність напрямків руху [1].

Маршрутна складова мережі задається файлами 2x2_sumo_grid.rou.xml та 2x2_sumo_grid.rou.alt.xml, які містять послідовності ребер для окремих транспортних засобів. У базовому маршрутизованому файлі сформовано 8412 транспортних засобів із часами відправлення в межах інтервалу моделювання від 0 до 3600 с. Це створює достатньо насичене транспортне середовище для перевірки того, як агенти реагують на черги, нерівномірні прибуття та перерозподіл потоку між суміжними перехрестями.

Для системи навчання важливо, що попит подається не лише як абстрактна інтенсивність, а як набір конкретних поїздок із маршрутами через мережу. Унаслідок цього дія світлофорного агента має відкладений ефект: розвантаження одного підходу може збільшити прибуття на наступне перехрестя, а надто довге утримання фази може створити черги на перпендикулярному напрямку.

Інформаційний шар транспортної мережі формується за допомогою даних смуг, світлофорів і симуляції. У реалізації через SUMO підписка виконується на змінні смуг, зокрема кількість транспортних засобів, кількість зупинених транспортних засобів, середню швидкість, зайнятість, час очікування, поточний час проїзду та ідентифікатори транспортних засобів на смузі. Для світлофорів зчитується поточна фаза, а для симуляції – кількість завантажених, відправлених, прибулих, телепортованих транспортних засобів, а також показники зіткнень і аварійного гальмування. Відповідна логіка підписки реалізована у файлі retriever.py, а крокове агрегування метрик – у файлі metrics.py; фрагменти цих файлів подано в додатку А. Додатковий файл 2x2_sumo_grid.add.xml містить 108 laneAreaDetector, розміщених на смугах із частотою збору даних 1 с, і належить до матеріалів додатка В. У межах підготовки стану агента ці вимірювання агрегуються до ознак перехрестя: кількості чергових

транспортних засобів, зайнятості вхідних і вихідних смуг, середньої швидкості, середньої затримки, часу очікування, поточної фази та часу перебування на фазі. Саме ці показники забезпечують зв’язок між фізичною мережею та навчальною моделлю.

Алгоритмічно підготовку транспортної мережі до навчального епізоду можна скласти з таких пунктів:

1) завантажити конфігурацію сценарію SUMO, файл дорожньої мережі, файл маршрутів і файл додаткових детекторів;

2) визначити множину регульованих перехресть і створити для кожного з них світлофорного агента;

3) для кожного агента сформувати перелік вхідних і вихідних смуг, допустимих фаз, зелених і жовтих інтервалів;

4) побудувати матрицю суміжності регульованих вузлів на основі дорожніх зв’язків між перехрестями;

5) підключити підписки до змінних смуг, світлофорів і загальних показників симуляції;

6) на кожному кроці моделювання агрегувати дані смуг у локальні ознаки агента;

7) передати сформовані стани до модуля вибору дій і після виконання дій оновити транспортну ситуацію в мережі.

Матриця суміжності є окремим елементом моделі мережі, оскільки саме вона визначає, які агенти можуть вважатися сусідами під час формування локально-мережевого стану. Якщо в конфігурації явно не задано список сусідів, він може бути отриманий із топології SUMO: регульовані вузли вважаються суміжними, коли між ними існує вихідне ребро від одного світлофорного вузла до іншого. Такий механізм реалізовано у властивості adjacency_matrix класу NetworkParser у файлі network_parser.py, фрагмент якого подано в додатку А. Для задачі мультиагентного навчання це дає змогу уникнути повної централізації. Агент не отримує глобальний стан усієї мережі, але може враховувати

ознаки найближчих вузлів, з якими його рішення безпосередньо пов’язане транспортним потоком. Такий підхід відповідає логіці масштабованого мультиагентного керування, у якому локальні політики мають координуватися через топологічно обґрунтовані зв’язки, а не через необмежений обмін усіма змінними мережі [16].

3.4 Представлення світлофорного об’єкта як інтелектуального агента

У розроблюваній мультиагентній системі світлофорний об’єкт розглядається не як пасивний виконавець наперед заданої програми, а як локальний інтелектуальний агент, що приймає рішення на основі поточного стану транспортного потоку.

Такий підхід конкретизує архітектуру, описану в попередніх підрозділах: транспортна мережа задає середовище, а кожне регульоване перехрестя стає окремим керуючим елементом, який спостерігає власні вхідні й вихідні смуги, оцінює фазовий стан та обирає допустиму дію. У межах базового SUMO-сценарію 2x2_sumo_grid це означає формування дев’яти агентів для вузлів A0, A1, A2, B0, B1, B2, C0, C1, C2, що задані у файлі 2x2_sumo_grid.net.xml і мають бути наведені в додатку В.

Інтелектуальність світлофорного агента в цій роботі полягає не в довільному генеруванні сигналів, а в адаптивному виборі керуючого рішення з безпечного простору дій. Світлофорний контролер зберігає фазову структуру, задану в мережевому файлі, а агент визначає, чи доцільно залишити поточну фазу активною або перейти до наступної дозволеної фази.

Таке подання узгоджується з реалізацією PhaseSwitchActionSpace, де дія має бінарний зміст: 0 відповідає збереженню поточної фази, а 1 – перемиканню за послідовною логікою round_robin. Для складніших сценаріїв може використовуватися PhaseSelectionActionSpace, у якому дія задає індекс фази, однак і в цьому випадку остаточне виконання проходить

через маску допустимих перемикань. Обидва класи реалізовані у файлі actions.py, фрагмент якого подано в додатку А.

Це важливо, оскільки агент не повинен порушувати мінімальну тривалість зеленого сигналу, жовтий інтервал або конфліктну сумісність напрямків [1]. Рисунок 3.6 демонструє схему взаємодії світлофорного об’єкта як інтелектуального агента.

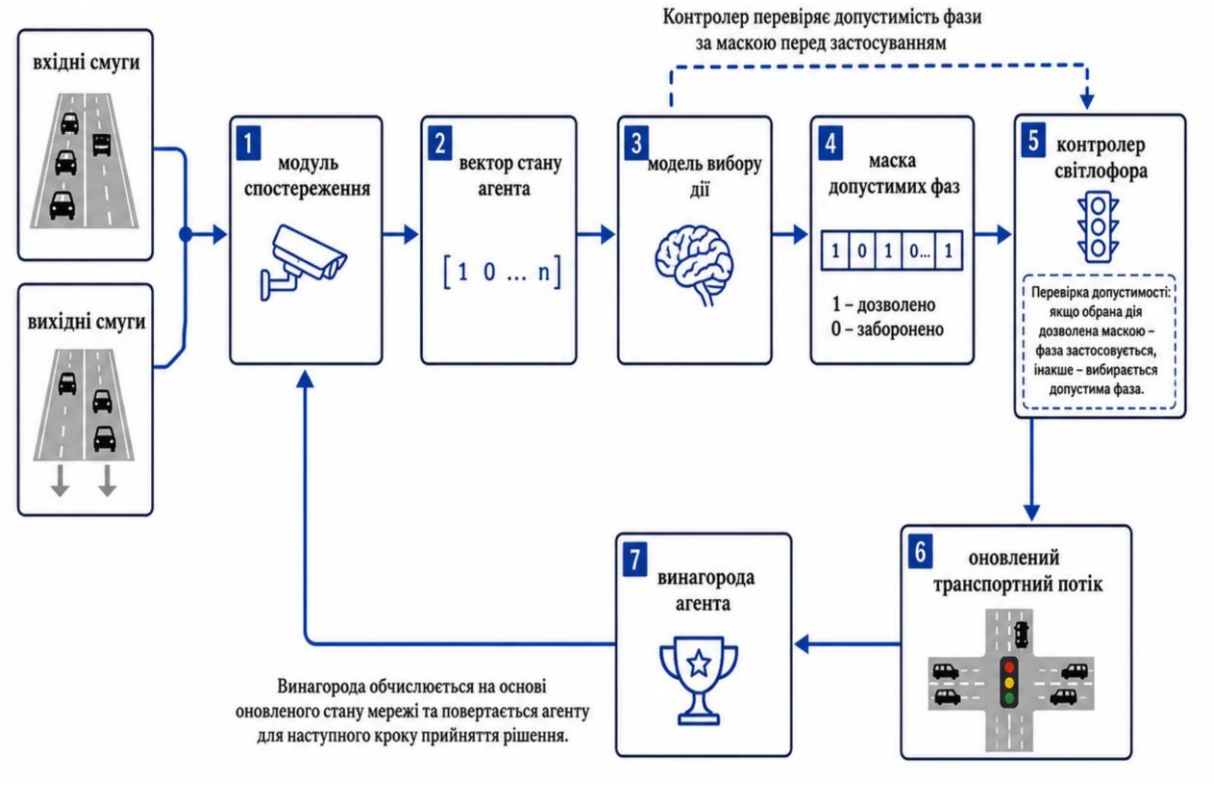


Рисунок 3.6 – Структура світлофорного об’єкта як інтелектуального агента

Локальна область спостереження агента формується навколо відповідного регульованого перехрестя. У програмній реалізації клас TrafficSignal зберігає ідентифікатор світлофорного вузла, перелік вхідних і вихідних смуг, поточну фазу, час перебування на фазі та агреговані показники руху.

Цей клас і метод update_stats() розміщені у файлі traffic_signal.py, фрагмент якого подано в додатку А.

Під час оновлення статистики для кожної вхідної смуги враховуються кількість транспортних засобів у черзі, зайнятість, середня швидкість, середня затримка, середній час очікування та позиційна матриця транспортних засобів у межах заданої видимості.

Для вихідних смуг додатково обчислюється зайнятість, оскільки вона показує, чи може перехрестя безпечно випускати транспортний потік у напрямку суміжного вузла. Саме ця ознака відрізняє агента від простого локального регулятора: він реагує не лише на чергу перед собою, а й на ризик передавання перевантаження нижче за потоком.

Стан агента в моделі не є повним описом усієї транспортної мережі. Він будується як компактне подання релевантних ознак, достатніх для прийняття локального рішення.

У реалізації простору спостережень передбачено кодування статичних характеристик смуг, зокрема довжини, кута, максимальної швидкості та позиції смуги в підході, а також динамічних показників: кількості чергових транспортних засобів, зайнятості та середньої швидкості.

До цього додається one-hot-подання поточної фази світлофора. Відповідна логіка міститься у класах PositionMatrix і LaneFeatures файлу observations.py, а статичні характеристики смуг зчитуються через NetworkParser у файлі network_parser.py; ці фрагменти подано в додатку А.

Для забезпечення однакової розмірності входів використовується доповнення до максимального числа контрольованих смуг і фаз, що дає змогу застосовувати спільну модель до кількох однотипних агентів. Практично це означає, що агент A0 і агент B1 можуть мати різну локальну ситуацію, але їхні спостереження подаються у стандартизованій формі, придатній для обробки нейронною мережею.

Таблиця 3.3 демонструє компоненти світлофорного агента в розроблюваній системі.

Таблиця 3.3 – Компоненти світлофорного агента в розроблюваній системі

Компонент агента	Реалізаційне подання	Призначення в системі
Ідентифікатор агента	ID світлофорного вузла SUMO	Пов’язує агента з конкретним перехрестям
Локальна область спостереження	Вхідні та вихідні смуги	Обмежує дані транспортним околом агента
Стан	Ознаки смуг, поточна фаза, час фази	Описує ситуацію для вибору дії
Простір дій	Збереження фази або перемикання	Формує керуюче рішення агента
Маска дій	Допустимі фазові переходи	Забороняє небезпечні або технічно неможливі дії
Винагорода	Черга, тиск, штраф за перемикання	Перетворює наслідки дії на навчальний сигнал
Пам’ять агента	Прихований стан GRU або досвід переходів	Дає змогу враховувати часову залежність руху
Комунікаційний контекст	Матриця суміжності агентів	Підтримує обмежену координацію між вузлами

Модель вибору дії в реалізаційному контурі може бути подана як нейронний агент світлофорного керування. У наявній структурі TSCAgent спочатку агрегує ознаки смуг, після чого використовує рекурентний блок GRUCell для врахування часової залежності станів і формує оцінки дій через вихідний лінійний шар. Клас TSCAgent і допоміжний агрегатор LaneAttentionAggregator реалізовані у файлі rl_controller.py, фрагмент якого подано в додатку А. Така побудова відповідає природі транспортного потоку: поточна черга або зайнятість смуги не завжди достатньо пояснює ситуацію, оскільки важливо знати, як вона змінювалася протягом попередніх кроків. Якщо черга швидко зростає після прибуття транспортної групи з сусіднього перехрестя, агент повинен відрізняти це від стабільного

слабкого попиту. Рекурентна пам’ять дає змогу частково врахувати таку динаміку без передавання агенту повної історії всієї мережі.

Для варіанта з мережевою координацією використовується графове розширення агента. У класі TSCGraphAgent після локального кодування смуг застосовується механізм GraphAttentionComm, який обмежує обмін інформацією матрицею суміжності регульованих вузлів. Обидва класи також розміщені у файлі rl_controller.py, тому їхній код подано в додатку А разом з описом нейронного контролера. Це означає, що агент не отримує необмежений глобальний стан, а враховує лише інформаційно пов’язані вузли, визначені структурою транспортної мережі. Такий підхід є практичним компромісом між незалежним локальним навчанням і повною централізацією. Повна централізація швидко збільшує розмірність спільного простору дій, тоді як повністю незалежні агенти можуть діяти егоїстично та переносити черги до сусідніх перехресть [16]. Графове подання зберігає автономність світлофорних агентів, але додає можливість враховувати найближчі мережеві наслідки.

Робота світлофорного агента на одному кроці моделювання:

1) отримати з SUMO поточні вимірювання для вхідних і вихідних смуг відповідного світлофорного вузла;

2) оновити локальні показники агента: чергу, зайнятість, середню швидкість, затримку, тиск, поточну фазу та нормований час на фазі;

3) сформувати вектор спостереження з ознак смуг і one-hot-подання активної фази;

4) передати спостереження до моделі вибору дії та отримати оцінки доступних рішень;

5) застосувати маску допустимих дій відповідно до мінімального зеленого часу, максимального зеленого часу, жовтих фаз і режиму round_robin;

6) обрати дію агента: залишити поточну фазу або перейти до наступної дозволеної фази;

7) передати дію до контролера світлофора, який виконує фазове перемикання через TraCI;

8) після оновлення симуляції розрахувати винагороду та зберегти перехід для навчання або оцінювання.

Функція винагороди задає критерій, за яким агент навчається відрізняти корисні рішення від небажаних. У наявній реалізації передбачено два змістовні варіанти: винагорода на основі довжини черги та винагорода на основі транспортного тиску. У першому випадку агент штрафується за накопичення черг на контрольованому перехресті, а також за небажане часте перемикання фаз через коефіцієнт flickering_coef. У другому випадку враховується різниця між завантаженням вхідних і вихідних смуг, що дає змогу оцінювати не тільки локальне очікування, а й баланс пропуску через вузол.

Для мультиагентного режиму локальна винагорода може доповнюватися внеском сусідніх агентів із ваговим зменшенням через параметр reward_gamma. Ця логіка реалізована в класах QueueLength і MaxPressure файлу reward.py, фрагмент якого подано в додатку А. Такий механізм потрібний для того, щоб агент не зменшував власну чергу ціною блокування наступного перехрестя. Водночас винагорода залишається достатньо локальною, щоб не перетворити задачу на повністю централізоване керування.

3.5 Організація взаємодії агентів у транспортній мережі

Організація взаємодії агентів у розроблюваній системі визначає, яким чином окремі світлофорні контролери узгоджують дії в межах спільної транспортної мережі. У попередніх підрозділах перехрестя було подано як локальний інтелектуальний агент, що отримує спостереження зі своїх смуг і вибирає допустиму фазову дію. Проте для міської мережі цього недостатньо: розвантаження одного підходу може збільшити прибуття

транспортних засобів до сусіднього вузла, а невчасне перемикання фази на одному світлофорі здатне сформувати хвилю затримок на суміжних перехрестях. Тому в цій роботі взаємодія агентів будується як обмежена мережева координація, а не як повністю ізольоване керування кожним світлофором.

У базовому сценарії 2x2_sumo_grid мультиагентна система складається з дев'яти агентів, пов'язаних із регульованими вузлами A0, A1, A2, B0, B1, B2, C0, C1, C2. Кожен агент зберігає власну область відповідальності: він безпосередньо керує тільки фазами свого світлофорного об'єкта через простір дій phase_switch або, у розширеному варіанті, phase_selection. Параметр action_space задається в конфігураціях сценаріїв, зокрема у файлі config.yaml, а для базового SUMO-сценарію параметри середовища подано у файлі config.yaml; ці конфігураційні файли подано в додатку Б.

Водночас агенти не розглядаються як незалежні системи, оскільки між ними існують транспортні зв'язки, задані дорожніми ребрами SUMO. Якщо вихідна ділянка одного перехрестя веде до іншого регульованого вузла, відповідні агенти вважаються суміжними. На цій основі формується матриця суміжності, яка використовується як структурне обмеження для обміну інформацією.

Така організація дає змогу враховувати найближчий транспортний контекст без передавання кожному агенту повного глобального стану мережі, що відповідало б уже централізованому керуванню й істотно збільшувало б розмірність задачі [16]. Рисунок 3.7 демонструє схему локально-мережевої взаємодії світлофорних агентів.

Для даного рисунка подано дев'ять агентів регулярної мережі 2x2_sumo_grid, де кожен вузол відповідає одному світлофорному агенту, а ребра між вузлами показують можливий напрям передавання транспортного потоку. Окремо варто позначити, що агент отримує детальні локальні ознаки зі своїх вхідних і вихідних смуг, а від сусідів використовує лише

агрегований комунікаційний контекст: ознаки завантаження, черги або приховане представлення стану, сформоване графовим модулем.

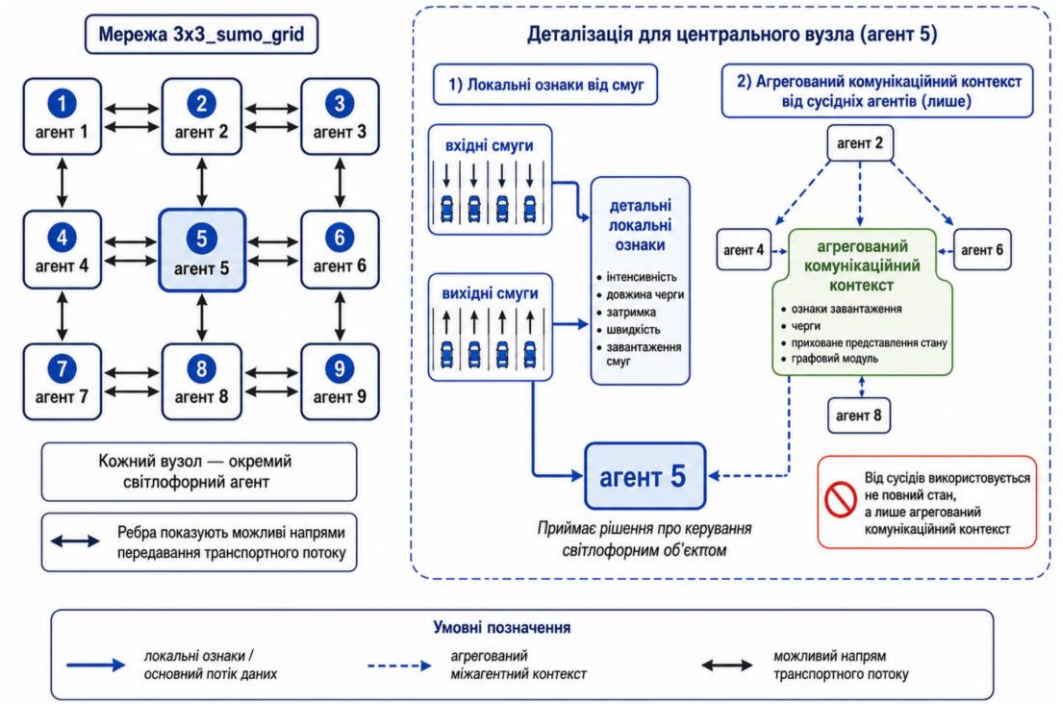


Рисунок 3.7 – Схема локально-мережевої взаємодії світлофорних агентів

У межах роботи передбачається три рівні узгодження дій агентів. Перший рівень відповідає незалежному децентралізованому керуванню, коли кожен агент приймає рішення лише за власним локальним спостереженням. Такий режим потрібний як базова реалізація і контрольний варіант, оскільки він показує, наскільки ефективною може бути адаптація світлофорів без спеціального механізму комунікації. Другий рівень передбачає використання спільної моделі для однотипних агентів: агенти мають різні спостереження, але однакову структуру політики, що дає змогу переносити закономірності керування між перехрестями зі схожою фазовою логікою. Третій рівень реалізує графову взаємодію через TSCGraphAgent, у якому локальні приховані стани агентів обробляються модулем GraphAttentionComm. Обидва компоненти реалізовані у файлі rl_controller.py, тому фрагмент їхнього коду подано в додатку А. У цьому

випадку комунікація не є довільною: механізм уваги обмежується матрицею суміжності, тому агент враховує лише ті вузли, з якими він пов’язаний транспортною топологією. Таблиця 3.4 демонструє рівні взаємодії агентів у розроблюваній мультиагентній системі.

Таблиця 3.4 – Рівні взаємодії агентів у мультиагентній системі

Рівень взаємодії	Реалізаційне подання	Призначення
Незалежні агенти	Локальні спостереження без графового обміну	Базовий децентралізований режим керування
Спільна модель	Єдина структура політики для кількох агентів	Узагальнення досвіду між однотипними перехрестями
Локально-мережева винагорода	Додавання внеску сусідніх агентів через reward_gamma	Зменшення егоїстичної поведінки окремого світлофора
Графова комунікація	TSCGraphAgent і GraphAttentionComm	Урахування станів суміжних вузлів за матрицею суміжності
Централізований простір дій	CentralizedActionSpace	Альтернативний варіант для малих мереж, але не основний у цій роботі

Ключовим рішенням для цієї роботи є вибір децентралізованого керування з локально-мережевою координацією. Повна централізація формально дає змогу одночасно вибирати комбінацію дій для всіх світлофорів, однак для дев’яти агентів навіть бінарний простір phase_switch швидко перетворюється на множину спільних комбінацій. Якщо ж використати вибір конкретної фази, розмірність такого простору зростає ще швидше. Крім того, централізована політика гірше масштабується на інші мережі, оскільки її структура стає прив’язаною до конкретної кількості перехресть. Тому централізований режим у наявній архітектурі доцільно розглядати як допоміжний або експериментальний варіант, тоді як основна логіка системи орієнтується на автономних агентів, здатних ураховувати наслідки своїх рішень для найближчого транспортного оточення [15].

Практично взаємодія агентів відбувається у двох точках навчального циклу. Перша точка пов’язана з формуванням рішення: агент кодує власні ознаки смуг і поточної фази, після чого в графовому варіанті його прихований стан узгоджується з прихованими станами суміжних агентів. Це не означає, що агент отримує повні таблиці черг усіх перехресть. Навпаки, обмін виконується через компактне внутрішнє представлення, яке вже пройшло локальне кодування. Друга точка пов’язана з винагородою. У функціях QueueLength і MaxPressure локальна винагорода агента може доповнюватися внесками сусідів з просторовим дисконтуванням через параметр reward_gamma, який визначений у конфігурації config.yaml і використовується у файлі reward.py. Відповідні матеріали доцільно розподілити між додатком Б для конфігурації та додатком А для програмної реалізації. Завдяки цьому агент отримує негативний сигнал не лише тоді, коли зростає його власна черга або тиск, а й тоді, коли його дія погіршує ситуацію на пов’язаних вузлах. Такий механізм особливо важливий для світлофорного керування, оскільки локально правильне рішення може бути мережево небажаним, якщо воно випускає потік у вже перевантажену ділянку. Узгодження дій світлофорних агентів на одному кроці моделювання:

1) отримати з SUMO поточні показники смуг, фаз і станів світлофорів для всіх регульованих перехресть;

2) для кожного агента сформувати локальне спостереження з ознак вхідних і вихідних смуг, поточної фази та часу перебування на фазі;

3) закодувати локальні ознаки агента за допомогою модуля агрегації смуг;

4) якщо використовується графова модель, передати приховані стани агентів до GraphAttentionComm з маскою, побудованою за матрицею суміжності;

5) для кожного агента сформувати оцінки доступних дій і застосувати маску допустимих фазових переходів;

6) передати вибрані дії до простору дій PhaseSwitchActionSpace або PhaseSelectionActionSpace;

7) виконати крок моделювання в SUMO з урахуванням мінімального зеленого часу, жовтого сигналу та режиму round_robin;

8) обчислити локальні винагороди агентів і доповнити їх внеском сусідніх вузлів відповідно до параметра reward_gamma;

9) зберегти переходи «стан – дія – винагорода – наступний стан» для подальшого навчання та аналізу.

Запропонована схема взаємодії також враховує проблему нестаціонарності мультиагентного навчання. Коли кожен агент змінює свою політику одночасно, умови навчання для інших агентів постійно змінюються: потік прибуття на перехрестя залежить не лише від зовнішнього попиту, а й від рішень попередніх світлофорів. Використання спільної моделі та графового обміну частково зменшує цю проблему. Спільна політика формує однакову логіку реакції на подібні транспортні ситуації, а графова комунікація дає змогу агенту розрізняти випадки, коли продовження зеленої фази справді розвантажує напрямок, і випадки, коли воно лише переносить чергу до сусіднього вузла.

Водночас така координація не усуває потреби в обмеженнях фазової безпеки: остаточне виконання дії залишається під контролем світлофорного модуля, який не дозволяє порушити мінімальні інтервали та допустиму послідовність фаз [8].

3.6 Формування вхідних даних і стану середовища

Формування вхідних даних у розроблюваній мультиагентній системі визначає, яку саме інформацію світлофорний агент отримує перед вибором керуючої дії. У попередніх підрозділах було показано, що агент пов’язується з конкретним регульованим перехрестям і діє в межах локальної області спостереження. Тому в цьому підрозділі стан середовища

розглядається не як повний опис усієї транспортної мережі, а як стандартизований вектор ознак, що відображає поточну ситуацію на вхідних і вихідних смугах відповідного світлофорного вузла.

У межах реалізації на основі SUMO первинні дані надходять із симуляційного середовища після кожного кроку моделювання. Для кожного агента зчитуються показники смуг, пов’язаних із його перехрестям: кількість транспортних засобів у черзі, середня швидкість, зайнятість смуги, середній час очікування, середня затримка та позиційне подання транспортних засобів у межах заданої зони видимості. Окремо фіксується поточна фаза світлофора, час перебування на ній і набір допустимих фазових переходів. Таке подання узгоджується з архітектурою TrafficSignal, у якій вхідні смуги використовуються для опису накопиченого попиту перед перехрестям, а вихідні смуги – для оцінювання здатності вузла прийняти транспортний потік після відкриття відповідного напрямку. Реалізація збору первинних даних пов’язана з файлами retriever.py, traffic_signal.py і observations.py, фрагменти яких подано в додатку А. Це принципово для світлофорного керування: агент не повинен орієнтуватися лише на найбільшу чергу перед собою, оскільки випуск потоку в уже перевантажену вихідну ділянку може перенести затримку до сусіднього перехрестя [8].

У цій роботі до складу стану агента доцільно включити шість основних груп ознак: поточну фазу, кількість транспортних засобів у черзі, середню швидкість, середню затримку, щільність або зайнятість потоку та показник pressure. Поточна фаза задає контекст прийняття рішення, оскільки одна й та сама черга може вимагати різних дій залежно від того, який напрямок уже обслуговується. Черга характеризує невдоволений попит на підході до перехрестя, середня швидкість відображає фактичну рухливість потоку, а затримка показує втрату часу порівняно з вільним рухом. Щільність або зайнятість потрібна для виявлення наближення до перенасичення смуги. Показник pressure доповнює ці ознаки, оскільки

порівнює завантаження вхідних і вихідних напрямків та дозволяє оцінити, чи не призведе дія агента до блокування нижче за потоком [16].

Особливістю вхідного вектора є поєднання статичних і динамічних характеристик. Статичні ознаки описують геометричні властивості смуг: довжину, кут, максимально допустиму швидкість і позицію смуги в підході. Вони не змінюються протягом епізоду, але допомагають моделі розрізняти смуги з різною пропускною роллю. Динамічні ознаки оновлюються на кожному кроці моделювання й відображають фактичний стан транспортного потоку. У реалізації простору LaneFeatures для кожної вхідної смуги формується комбінація статичних параметрів і поточних показників n_queued, occupancy, mean_speed, після чого до загального вектора додається one-hot-подання активної фази. Цей простір реалізовано у файлі observations.py; фрагмент подано в додатку А.

Розширене змістове подання стану в межах цієї роботи також враховує середню затримку, нормований час перебування на фазі та pressure, які обчислюються на рівні світлофорного об’єкта у файлі traffic_signal.py. Таке розділення важливе для подальшого навчання: модель отримує не лише «знімок» поточної черги, а й контекст того, наскільки довго триває активна фаза та чи є вихідний напрямок здатним прийняти додатковий потік. Таблиця 3.5 демонструє структуру вхідного вектора стану світлофорного агента.

Таблиця 3.5 – Структура вхідного вектора стану світлофорного агента

Група ознак	Реалізаційне подання	Призначення для прийняття рішення
1	2	3
Поточна фаза	One-hot-вектор phase_id	Визначає активний напрямок обслуговування
Час на фазі	Нормоване значення norm_time_on_phase	Дає змогу враховувати мінімальні та максимальні тривалості фаз

Продовження таблиці 3.5

1	2	3
Черга	n_queued для вхідних смуг	Показує накопичений попит перед перехрестям
Середня швидкість	mean_speed	Відображає рівень рухливості транспортного потоку
Середня затримка	mean_delay або похідна від швидкості	Характеризує втрати часу на підходах
Щільність / зайнятість	occupancy для вхідних і вихідних смуг	Дозволяє виявляти насичення підходів і виходів
Pressure	Різниця між завантаженням входів і виходів	Оцінює баланс пропуску через вузол
Статичні ознаки смуг	Довжина, кут, максимальна швидкість, індекс смуги	Забезпечують однакову інтерпретацію смугових показників

Щоб різні агенти могли використовувати спільну нейронну модель, вхідні вектори мають однакову розмірність. У транспортній мережі окремі перехрестя можуть мати різну кількість контрольованих смуг, тому без вирівнювання модель отримувала б несумісні за довжиною спостереження. У програмній структурі це вирішується доповненням вектора до максимального числа контрольованих смуг і фаз. Для відсутніх смуг використовується службове значення pad_value, а фазовий компонент доповнюється до фіксованої максимальної кількості фаз. Параметр pad_value задається у конфігурації config.yaml, а функція доповнення використовується в реалізації спостережень observations.py; відповідні фрагменти подано в додатках Б і А відповідно. Такий підхід не змінює фізичний зміст стану: агент не отримує вигаданих транспортних даних, а лише приводить локальне спостереження до формату, придатного для пакетної обробки нейронною мережею. Крім того, фіксована структура стану спрощує використання спільної політики для агентів A0, A1, B1 та

інших вузлів базової мережі 2x2_sumo_grid, оскільки всі вони передають до моделі однаково організовані ознаки. Формування вектора стану світлофорного агента:

1) отримати з SUMO поточні значення для смуг, пов’язаних із відповідним світлофорним вузлом;

2) для кожної вхідної смуги визначити кількість транспортних засобів у черзі, зайнятість, середню швидкість, середній час очікування та середню затримку;

3) для вихідних смуг обчислити зайнятість, необхідну для оцінювання можливості пропуску потоку після зміни фази;

4) визначити поточну фазу світлофора та подати її у вигляді one-hot-вектора;

5) обчислити агреговані показники перехрестя: середню швидкість, середню затримку, сумарну чергу, нормований час перебування на фазі та pressure;

6) об’єднати статичні ознаки смуг, динамічні показники руху та фазові ознаки в єдиний вектор стану;

7) виконати нормалізацію та доповнення вектора до фіксованої розмірності;

8) передати сформований стан до нейронної моделі агента для вибору допустимої керуючої дії.

Сформований таким чином стан має бути достатньо інформативним, але не надмірним. Якщо включити до нього всі доступні змінні симулятора, модель може почати навчатися на технічних деталях сценарію, які погано узагальнюються на іншу мережу або інший попит. Якщо ж залишити лише чергу, агент не зможе відрізнити короткочасне накопичення транспортних засобів від стійкого перенасичення, а також не врахує стан вихідних ділянок. Тому обраний набір ознак є компромісом між локальною спостережуваністю та практичною придатністю для навчання. Він відображає саме ті характеристики, які мають безпосередній зв’язок із

фазовим рішенням: кого обслуговує поточна фаза, скільки транспортних засобів очікує, наскільки повільно рухається потік, чи накопичується затримка та чи не створюється надлишковий тиск на сусідні вузли [13].

3.7 Проєктування простору керуючих дій

Проєктування простору керуючих дій визначає, які саме рішення може приймати світлофорний агент після отримання вектора стану. У межах цієї роботи дія не трактується як довільне керування всіма сигналами світлофора, оскільки фазова структура вже задана у сценарії SUMO і повинна зберігати сумісність транспортних напрямків. Тому агент впливає на світлофорний об’єкт через обмежений набір фазових рішень, а виконання цих рішень додатково перевіряється контролером допустимості фаз [1].

У базовій реалізації мультиагентної системи використовується дискретний бінарний простір дій, що відповідає класу PhaseSwitchActionSpace. Для кожного світлофорного агента дія 0 означає збереження поточної фази, а дія 1 – ініціювання переходу до наступної фази відповідно до послідовності round_robin. Клас PhaseSwitchActionSpace розміщений у файлі actions.py, фрагмент якого подано в додатку А. Такий вибір узгоджується з фактичною структурою сценарію 2x2_sumo_grid, де світлофорні об’єкти мають наперед визначені зелені та жовті фази, а конфігурація config.yaml задає мінімальний зелений час, максимальний зелений час і тривалість жовтого сигналу. Отже, агент не створює нових фаз і не змінює конфліктні напрямки вручну, а лише визначає момент, коли поточне обслуговування доцільно продовжити або завершити. Це зменшує складність навчання, оскільки модель має оцінити два варіанти поведінки замість повного набору фазових комбінацій. Рисунок 3.8 демонструє простір керуючих дій світлофорного агента.

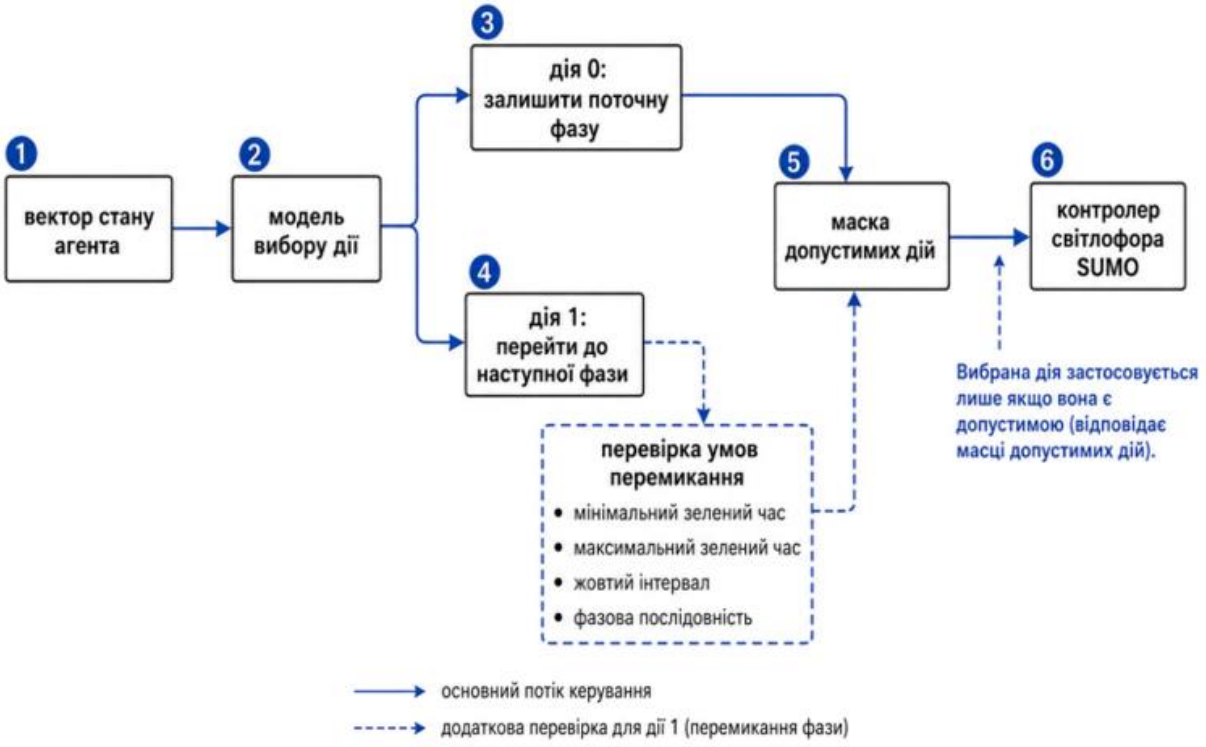


Рисунок 3.8 – Простір керуючих дій світлофорного агента

Практичний зміст бінарного простору дій полягає в тому, що агент навчається не повному синтезу світлофорної програми, а адаптивному вибору моменту фазового переходу. Якщо на активному напрямку зберігається значна черга, а вихідна ділянка здатна прийняти транспортний потік, дія збереження фази може бути доцільною. Якщо ж черга на поточному напрямку зменшилась або на конфліктному підході накопичується затримка, агент може ініціювати перемикання. Водночас команда switch не означає миттєвого відкриття нового зеленого сигналу. У програмній логіці спочатку визначається поточний індекс фази, потім обчислюється індекс наступної фази, після чого метод action_to_phase передає рішення до світлофорного контролера. Якщо перемикання не дозволене, відповідна дія блокується маскою, сформованою на основі get_allowable_phase_switches. Саме ця маска відокремлює навчальне рішення від технічно допустимого виконання в симуляторі [8]. Таблиця 3.6 демонструє базові керуючі дії світлофорного агента.

Таблиця 3.6 – Базові керуючі дії світлофорного агента

Код дії	Реалізаційне подання	Зміст рішення агента	Умова виконання
0	hold	Залишити активною поточну фазу	Дозволено, якщо поточна фаза може бути продовжена без порушення максимального часу
1	switch	Перейти до наступної фази за послідовністю round_robin	Дозволено після виконання мінімального зеленого часу та з урахуванням жовтого інтервалу

Особливістю такого простору дій є його придатність для першої реалізації та подальшого порівняння з розширеними варіантами. Бінарна дія дає змогу простіше інтерпретувати поведінку агента: кожне рішення можна пояснити як продовження обслуговування поточного напряму або підготовку до переходу на наступний. Це важливо для третього розділу, оскільки система повинна бути не лише формально побудована, а й придатна для подальшого аналізу журналів рішень, фаз і транспортних метрик. Недоліком такого підходу є обмежена гнучкість. Агент не може безпосередньо вибрати довільну зелену фазу з усього набору, тому в ситуаціях нерівномірного попиту він змушений рухатися за встановленою фазовою послідовністю. Проте для базового навчального контуру це обмеження є прийнятним, оскільки воно знижує ризик нестійкої поведінки та спрощує перевірку впливу дій на черги, затримки й пропуск транспортних засобів [16].

Вибір і застосування керуючої дії світлофорного агента:

– отримати поточний вектор стану агента, сформований за ознаками смуг, фази, часу перебування на фазі та показниками завантаження;

– передати стан до моделі вибору дії та отримати оцінки для дій 0 і 1;

– сформувати маску допустимих дій через контролер світлофорного об’єкта;

– заблокувати дії, які порушують мінімальний зелений час, максимальний зелений час або послідовність фаз;

– якщо вибрано дію 0, передати до контролера індекс поточної фази;

– якщо вибрано дію 1, визначити індекс наступної фази за логікою round_robin і передати його до контролера;

– виконати крок симуляції в SUMO з урахуванням жовтого інтервалу та оновити транспортний стан;

– зберегти пару «стан – дія» разом із наступним станом, винагородою та службовими метриками епізоду.

У системі також передбачена можливість розширення простору керуючих дій через PhaseSelectionActionSpace, де дія задає індекс конкретної фази. Цей клас реалізовано у файлі actions.py, а приклади використання параметра phase_selection наявні в конфігураціях сценаріїв cityflow і в базовому файлі config.yaml. Такий варіант може бути корисним для сценаріїв, у яких фази не повинні проходити лише за жорсткою циклічною послідовністю. Наприклад, якщо один із підходів має стійке перевантаження, прямий вибір фази потенційно дозволяє швидше надати йому зелений сигнал. Однак для цієї роботи такий простір доцільно розглядати як наступний етап розвитку системи, а не як основний початковий режим. Причина полягає в тому, що збільшення кількості дій ускладнює навчання, підвищує ймовірність частих перемикань і потребує ретельнішого аналізу масок допустимості. Крім того, у мультиагентній мережі прямий вибір фази одним агентом може різко змінити прибуття транспортних засобів до сусіднього перехрестя, тому така гнучкість повинна поєднуватися з механізмами локально-мережевої координації.

3.8 Розробка функції винагороди для оптимізації транспортного потоку

Функція винагороди в розроблюваній мультиагентній системі визначає, за яким критерієм світлофорний агент оцінює наслідки власної дії. Якщо у підрозділі 2.10 було обґрунтовано загальну структуру винагороди для задачі підкріплювального навчання, то в цьому підрозділі конкретизується її реалізаційне подання для запропонованої системи. У межах третього розділу винагорода розглядається не як абстрактна математична функція, а як модуль, що отримує показники з симуляційного середовища SUMO, пов’язує їх із діями агента та формує числовий навчальний сигнал для оновлення політики [8].

У розроблюваній системі кожне регульоване перехрестя подається як окремий агент, тому винагорода має бути локальною, але не повністю ізольованою. Основним джерелом оцінки є стан підходів до відповідного світлофорного об’єкта: кількість транспортних засобів у черзі, середня затримка, поточна фаза та факт перемикання сигналу. Такий вибір показників безпосередньо пов’язаний із транспортною метою керування. Черга відображає накопичений незадоволений попит перед стоп-лінією, тоді як затримка показує, наскільки довго транспортні засоби фактично втрачають час через обраний режим регулювання. Якщо агент орієнтуватиметься лише на кількість автомобілів у черзі, він може недооцінювати ситуації, у яких черга ще не є великою, але очікування вже стає непропорційним для окремого напрямку. Якщо ж використовувати тільки затримку, навчальний сигнал може повільніше реагувати на швидке накопичення транспортних засобів після різкої зміни попиту. Тому в цій роботі винагорода будується як штрафна комбінація двох базових критеріїв: n_queued і mean_delay. Додатково вводиться штраф за часте перемикання фаз, оскільки кожне перемикання супроводжується жовтим сигналом,

втратою ефективного зеленого часу та потенційним погіршенням плавності руху [1].

У фактичній структурі проєкту така логіка узгоджується з модулем reward.py, де передбачено базовий інтерфейс BaseRewardFunction і окремі реалізації винагород, зокрема QueueLength та MaxPressure. Клас QueueLength уже використовує штраф за кількість транспортних засобів у черзі та коефіцієнт flickering_coef, який відповідає за небажаність надмірного перемикання фаз. Водночас у модулі метрик SUMO metrics.py для мережі обчислюється mean_delay, а у файлі traffic_signal.py цей показник розраховується для окремого світлофорного об'єкта. Відповідні фрагменти коду подано в додатку А. Запропонований критерій не потребує введення штучних або недоступних змінних: усі його складові можуть бути отримані з уже наявного контуру збору симуляційних показників.

Локальну складову винагороди для і-го агента на кроці t доцільно задати як від'ємну оцінку транспортного перевантаження на контрольованому перехресті:

$$R_t = -\left(\alpha_q \cdot Q_t + \alpha_d \cdot D_t + \alpha_s \cdot S_t\right), \tag{3.1}$$

де R_t – винагорода світлофорного агента на кроці моделювання t;

Q_t – кількість транспортних засобів у черзі на підходах до відповідного перехрестя;

D_t – середня затримка транспортних засобів на підходах до перехрестя;

S_t – індикатор перемикання фази світлофора на поточному кроці, St = 1, якщо фазу було перемкнено, і St = 0, якщо поточну фазу було залишено без зміни;

α_q – ваговий коефіцієнт штрафу за довжину черги;

α_d – ваговий коефіцієнт штрафу за середню затримку;

α_s – ваговий коефіцієнт штрафу за перемикання фази.

Знак мінус у формулі (3.1) має принципове значення: агент не отримує позитивну винагороду за сам факт виконання дії, а намагається мінімізувати накопичений штраф. Якщо після продовження зеленої фази черга на активному напрямку зменшується, а затримка на інших підходах не зростає критично, абсолютне значення штрафу стає меншим. Якщо ж агент надто довго утримує фазу або перемикає її без реальної потреби, значення q_t^i, d_t^i або f_t^i збільшується, що погіршує навчальний сигнал. Така побудова є зручною для реалізації, оскільки вона безпосередньо використовує показники, які вже зберігаються у статистиці світлофорного об'єкта: ts.n_queued, ts.mean_delay і ts.controller.program.phase_changed; вони формуються у файлах traffic_signal.py і reward.py. Таблиця 3.7 демонструє реалізаційні компоненти функції винагороди.

Таблиця 3.7 – Реалізаційні компоненти функції винагороди

Компонент	Джерело в системі	Зміст для агента	Вплив на винагороду
Черга	ts.n_queued	Кількість транспортних засобів, що очікують на вхідних смугах	Збільшує штраф при накопиченні попиту
Середня затримка	ts.mean_delay	Втрата швидкості відносно умовного вільного руху	Збільшує штраф при погіршенні обслуговування
Перемикання фази	phase_changed	Ознака зміни світлофорної фази на кроці моделювання	Обмежує надмірно часті перемикання
Просторове узгодження	reward_gamma, k_hop_neighbors	Урахування винагород сусідніх агентів	Зменшує ризик перенесення затору до суміжного вузла

Особливістю мультиагентної постановки є те, що локально корисна дія не завжди є корисною для транспортної мережі. Наприклад, агент може швидко зменшити власну чергу, випустивши потік на вихідну ділянку,

однак якщо сусіднє перехрестя вже перевантажене, така дія лише переносить затор далі по маршруту. Тому в запропонованій системі локальна винагорода доповнюється просторово дисконтованою складовою, яка враховує найближчих сусідів. У наявній реалізації для цього використовується параметр reward_gamma та структура k_hop_neighbors: агент отримує власний штраф повністю, а штрафи сусідніх вузлів додаються з коефіцієнтом, що зменшується зі збільшенням відстані в графі мережі. Параметр reward_gamma подано у файлі config.yaml, структура сусідів формується у файлі network_parser.py, а розрахунок дисконтованої винагороди реалізовано у файлі reward.py; ці матеріали подано в додатках Б і А.

Такий механізм не перетворює систему на централізований контролер, але змушує агентів частково враховувати наслідки власних рішень для прилеглих перехресть. Це відповідає логіці масштабованого мультиагентного керування, де кожен агент приймає рішення локально, проте навчальний сигнал містить обмежений мережевий контекст [16].

Розрахунок винагороди світлофорного агента:

1) після виконання дії агента оновити стан транспортної мережі в SUMO;

2) для кожного світлофорного об’єкта зчитати кількість транспортних засобів у черзі n_queued;

3) обчислити середню затримку mean_delay за показниками швидкості на вхідних смугах;

4) визначити, чи відбулося перемикання фази на поточному кроці;

5) розрахувати локальну штрафну винагороду за формулою (3.1);

6) для кожного агента знайти сусідні світлофорні об’єкти за структурою k_hop_neighbors;

7) додати до локальної винагороди дисконтовані винагороди сусідів із використанням коефіцієнта reward_gamma;

8) передати підсумкову винагороду до модуля навчання разом зі станом, дією та наступним станом.

Запропонована функція винагороди має також важливе інтерпретаційне значення для подальших експериментів. Якщо після навчання спостерігається зменшення середньої черги, але водночас зростає кількість перемикань, це означатиме, що ваговий коефіцієнт αf є недостатнім для стримування нестабільної поведінки. Якщо зменшується локальна черга, але погіршуються показники сусідніх перехресть, потрібно посилити мережеву складову через reward_gamma або обмежити глибину врахування сусідів. Якщо ж агент занадто консервативно утримує фази, доцільно переглянути співвідношення між αq і αd, оскільки затримка на конфліктних підходах може бути недооцінена. Отже, параметри винагороди виконують не лише роль числових коефіцієнтів, а й задають транспортну поведінку системи: пріоритет швидкого розвантаження черг, рівномірного обслуговування напрямків або стабільності фазових переходів.

3.9 Вибір і порівняння підходів до навчання мультиагентної системи на основі DQN та PPO

Вибір алгоритму навчання в розроблюваній мультиагентній системі не зводиться до формального порівняння точності різних нейромережевих моделей. Для світлофорного регулювання важливішим є те, яку логіку прийняття рішень алгоритм вбудовує в роботу агента: оцінювання цінності окремих дій або безпосереднє навчання політики керування. З урахуванням уже визначеного простору станів, бінарного простору дій phase_switch, функції винагороди та локально-мережевої взаємодії агентів у цій роботі доцільно розглядати два альтернативні навчальні контури: DQN як ціннісний підхід і PPO як підхід оптимізації політики [13], [14].

DQN у межах цієї системи доцільно використовувати як базову нейромережеву модель для дискретного фазового керування. Кожен

світлофорний агент отримує вектор стану, сформований за ознаками вхідних і вихідних смуг, поточної фази, часу перебування на фазі та агрегованих показників сусідніх вузлів. На виході модель формує значення для допустимих дій, які в базовому режимі відповідають рішенню залишити поточну фазу або перейти до наступної дозволеної фази. Така логіка безпосередньо узгоджується з класом PhaseSwitchActionSpace у файлі actions.py, де простір дій має розмірність 2, а маска допустимості не дозволяє агенту порушувати фазові обмеження. Реалізаційною основою для цього варіанта є також файл rl_controller.py, у якому класи TSCAgent, TSCGraphAgent і TSCAgentEnsemble перетворюють ознаки смуг і фаз у вихідний вектор дій; фрагменти цього коду подано в додатку А.

Перевага DQN полягає в простоті інтерпретації: якщо для певного стану Q-значення дії перемикання перевищує оцінку продовження фази, агент має підставу змінити сигнал. Разом із тим така модель чутлива до складу винагороди, якості буфера досвіду та різноманітності сценаріїв попиту. Якщо навчання відбувається на вузькому наборі транспортних ситуацій, агент може переоцінити локально вигідні рішення й погіршити стан сусідніх перехресть.

PPO розглядається як альтернативний варіант навчального контуру, у якому агент не лише оцінює дії, а навчає політику їх вибору. У цьому випадку нейронна модель формує розподіл імовірностей між допустимими фазовими рішеннями, а дія може вибиратися стохастично під час навчання або детерміновано під час оцінювання.

Для розроблюваної системи це має практичне значення, оскільки транспортне середовище є нестаціонарним: зміна поведінки одного агента впливає на потоки, що надходять до інших вузлів. У таких умовах жорсткий вибір дії з найбільшим Q-значенням не завжди є достатнім, особливо на ранніх етапах навчання. PPO дає змогу оновлювати політику поступово, обмежуючи різкі зміни поведінки агента, що важливо для уникнення надмірних перемикань, нестійкого обслуговування підходів і випадкового

закріплення невдалих фазових рішень [14]. У доопрацьованій програмній структурі цей підхід реалізовано у файлі marl_ppo.py, де клас MARLPPOTrainer використовує спільну actor-critic-модель SharedActorCritic, rollout-буфер, GAE, clipping, entropy bonus, gradient clipping, checkpoint і журналювання evaluation-метрик. Такий варіант не змінює транспортну логіку системи, але змінює спосіб оновлення параметрів моделі та дає змогу порівнювати DQN і PPO в одному SUMO-контурі. Рисунок 3.9 демонструє порівняння навчальних контурів DQN та PPO у мультиагентній системі світлофорного керування.

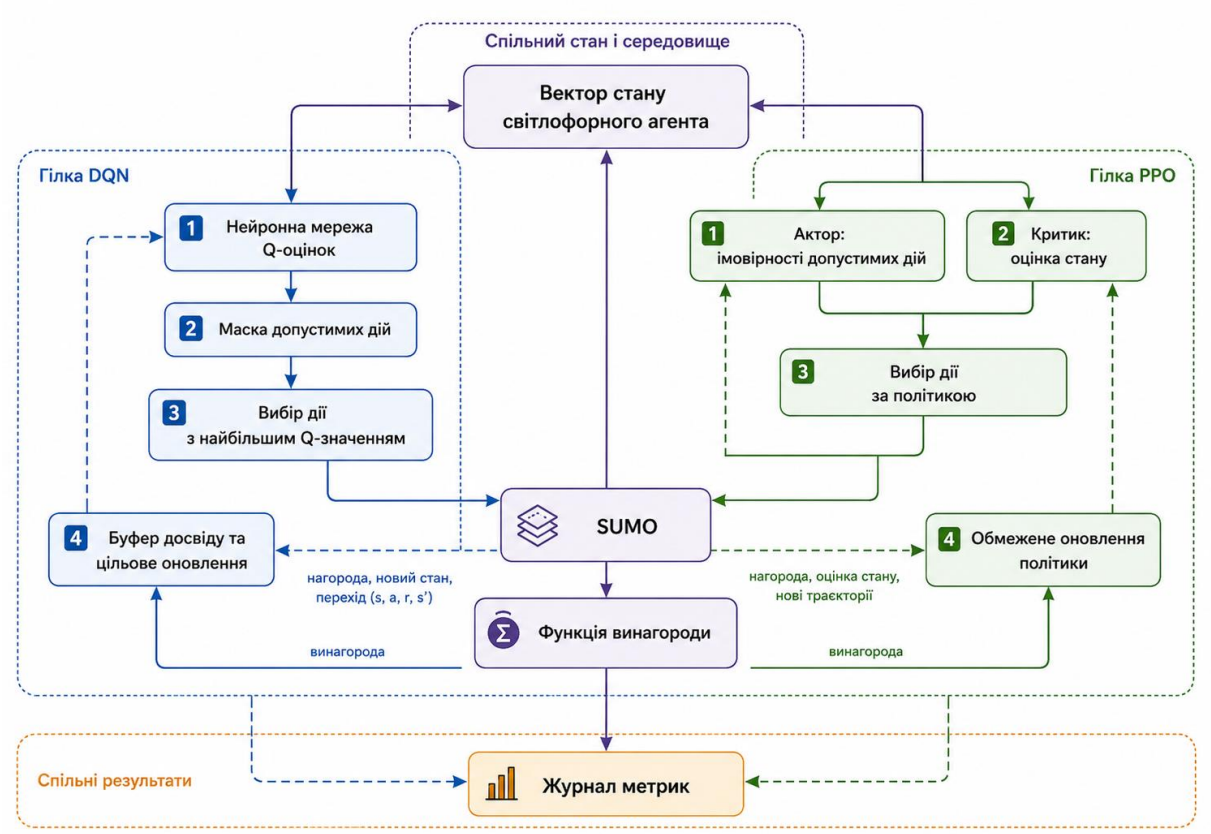


Рисунок 3.9 – Порівняння навчальних контурів DQN та PPO у мультиагентній системі світлофорного керування

Порівняння DQN і PPO в цій роботі потрібно виконувати не як пошук універсально кращого алгоритму, а як перевірку двох різних способів

організації керування. DQN відповідає ситуації, коли агент має невелику множину чітко визначених дій і повинен навчитися оцінювати, яка з них є найкориснішою в поточному стані. Це добре узгоджується з базовим режимом phase_switch, де рішення має зрозумілу транспортну інтерпретацію. PPO, навпаки, корисний тоді, коли важливо контролювати плавність зміни поведінки політики та підтримувати дослідження альтернативних дій без різкого руйнування вже знайдених режимів керування. Таблиця 3.8 демонструє роль DQN та PPO у розроблюваній мультиагентній системі.

Таблиця 3.8 – Роль DQN та PPO у розроблюваній мультиагентній системі

Ознака порівняння	DQN	PPO
Логіка навчання	Оцінювання цінності дій через Q-функцію	Безпосереднє навчання політики вибору дій
Вихід моделі	Q-значення для допустимих дій	Імовірності дій і оцінка стану через actor-critic
Основний простір дій	phase_switch: залишити фазу або перемкнути	Той самий дискретний простір із маскуванням недопустимих дій
Перевага для системи	Проста інтерпретація рішень і зручність для базового порівняння	Стабільніше оновлення політики в складному середовищі
Основне обмеження	Чутливість до буфера досвіду, цільових оцінок і локальних максимумів	Потреба в більшій кількості симуляційних епізодів і налаштуванні політики
Роль у квал. роботі	Базовий ціннісний підхід до керування агентами	Альтернативний політичний підхід для експериментального зіставлення

Для світлофорної мережі це особливо суттєво в напружених сценаріях, де одна дія може тимчасово покращити локальну чергу, але погіршити приймальну здатність сусіднього вузла. Тому обидва підходи мають використовувати однаковий інтерфейс до середовища: однакове формування стану, однаковий простір допустимих дій, однакову функцію винагороди та однакові метрики оцінювання результатів. Лише за такої

умови подальше експериментальне порівняння буде відображати відмінність алгоритмів навчання, а не різницю в підготовці даних або конфігурації симуляції [8], [16].

Організація порівняльного навчання DQN та PPO:

1) ініціалізувати SUMO-сценарій, набір світлофорних агентів, простір дій phase_switch, маски допустимих фазових переходів і параметри винагороди;

2) для кожного агента сформувати вектор стану за ознаками смуг, поточної фази, часу перебування на фазі та локально-мережевих показників;

3) у режимі DQN передати стан до мережі Q-оцінок, застосувати маску допустимих дій і вибрати дію за найбільшим доступним значенням або за дослідницькою стратегією;

4) у режимі PPO передати стан до актора, застосувати маску допустимих дій, сформувати розподіл імовірностей і вибрати дію відповідно до поточної політики;

5) передати вибрані дії до світлофорного контролера, виконати крок симуляції та оновити транспортний стан у SUMO;

6) розрахувати винагороду за чергами, затримками, перемиканнями фаз і просторовим внеском сусідніх агентів;

7) зберегти переходи, службові показники та метрики епізоду у форматі, придатному для подальшого аналізу;

8) оновити параметри DQN через цільові Q-оцінки або параметри PPO через обмежене оновлення політики;

9) після завершення епізодів порівняти алгоритми за середньою затримкою, довжиною черги, кількістю зупинок, стабільністю перемикань і узгодженістю поведінки агентів.

Для коректного зіставлення алгоритмів важливо зберегти однакову архітектурну основу. Обидва підходи повинні працювати з тими самими даними, що надходять із модулів observations.py, traffic_signal.py і retriever.py, та застосовувати ті самі обмеження дій із actions.py. У цьому

разі різниця між результатами DQN і PPO буде пов’язана саме з механізмом навчання. Якщо DQN покаже кращу поведінку в простому дискретному сценарії, це підтвердить достатність ціннісного підходу для базового фазового перемикання. Якщо PPO забезпечить меншу кількість різких перемикань або стабільніші показники в умовах змінного попиту, це свідчитиме про перевагу політичного підходу для складніших режимів мультиагентного керування.

3.10 Архітектура нейромережевих моделей DQN та PPO для світлофорного агента

Архітектура нейромережевої моделі в розроблюваній системі визначає спосіб перетворення транспортного стану перехрестя на керуюче рішення. У попередніх підрозділах уже було визначено структуру стану, бінарний простір дій phase_switch, функцію винагороди та логіку порівняння DQN і PPO. Тому в цьому підрозділі акцент робиться не на загальному описі алгоритмів, а на тому, як саме ці підходи можуть бути побудовані для світлофорного агента в межах програмного контуру Multi-Agent for Traffic Signal Control/. Обидві моделі використовують однаковий вхідний опис перехрестя: поточну фазу, черги, середню швидкість, середню затримку, щільність потоку та показник транспортного тиску, що дає змогу порівнювати їх за однакових умов [13], [14].

У базовій реалізаційній логіці стан агента передається до нейронної моделі як стандартизований вектор фіксованої розмірності. Це важливо для мультиагентної мережі, оскільки різні перехрестя можуть мати різну кількість контрольованих смуг і фаз, але спільна модель повинна обробляти їх в одному форматі. У наявній програмній структурі така ідея підтримується класами LaneFeatures, PositionMatrix, TSCAgent, TSCGraphAgent і TSCAgentEnsemble, розміщеними у файлах observations.py та /rl_controller.py; відповідні фрагменти подано в додатку А. Для

компактного варіанта DQN вектор стану може подаватися безпосередньо до повнозв'язної мережі, однак у розроблюваній системі доцільно зберегти поділ на кодування смугових ознак і фазового стану. Статичні та динамічні характеристики смуг спочатку перетворюються у приховане подання розмірності 64, після чого воно поєднується з one-hot-поданням поточної фази. Така структура дає змогу моделі не змішувати механічно різні типи ознак: геометрія смуги, поточна черга, швидкість і фазовий стан мають різну природу, але разом визначають доцільність продовження або перемикання сигналу. Рисунок 3.10 демонструє Архітектура нейромережевих моделей DQN та PPO для світлофорного агента.

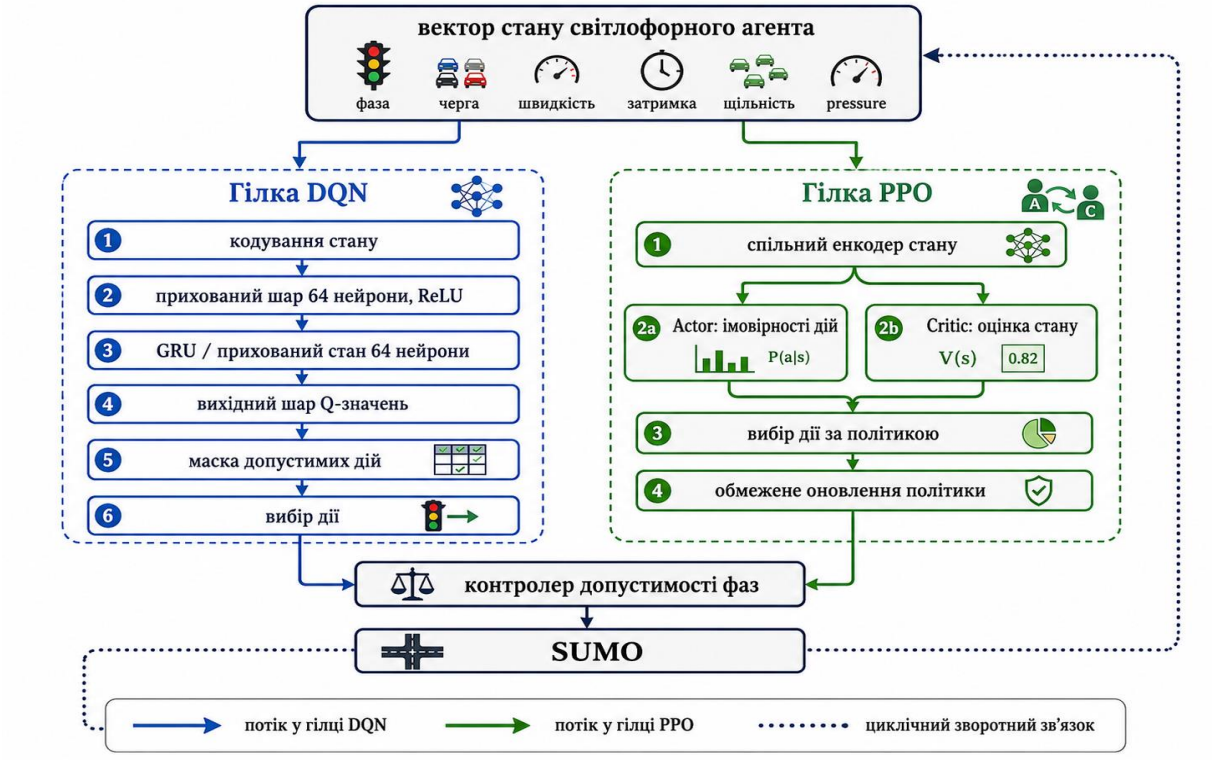


Рисунок 3.10 – Архітектура нейромережевих моделей DQN та PPO для світлофорного агента

DQN-модель у цій роботі виконує роль Q-мережі, яка для кожного стану оцінює корисність кожної доступної дії. Якщо використовується базовий простір phase_switch, вихідний шар має два нейрони: перший

відповідає дії збереження поточної фази, другий – дії переходу до наступної дозволеної фази. Прихована частина моделі може бути побудована як послідовність з енкодера стану, шару з 64 нейронами та функцією активації ReLU, рекурентного блоку GRUCell з 64 прихованими одиницями й лінійного вихідного шару. Рекурентний блок є доцільним саме для світлофорного керування, оскільки поточний стан перехрестя не завжди достатньо пояснює транспортну ситуацію. Наприклад, однакова черга може означати короткочасне прибуття транспортного пакета або наслідок тривалого недообслуговування напряму. Прихований стан допомагає моделі враховувати попередню динаміку без явного збереження довгої історії спостережень у векторі стану.

Після обчислення Q-значень модель не передає дію безпосередньо до SUMO. Спочатку до вихідного вектора застосовується маска допустимих дій, сформована простором PhaseSwitchActionSpace. Якщо перемикання фази неможливе через мінімальний зелений час або інше фазове обмеження, відповідна дія блокується і не може бути вибрана навіть за високого Q-значення. Під час навчання для DQN доцільно використовувати дослідницьку стратегію ε-greedy: з певною ймовірністю агент обирає випадкову допустиму дію, а в інших випадках – дію з найбільшим доступним Q-значенням. Параметри мережі можуть оновлюватися оптимізатором Adam за похибкою між поточними Q-оцінками та цільовими значеннями, сформованими на основі винагороди й наступного стану. У тексті роботи це варто трактувати як проєктне рішення навчального контуру, оскільки наявний файл rl_controller.py передусім фіксує структуру агента й механізм вибору дії, а не повний тренувальний цикл.

PPO має іншу архітектурну логіку, оскільки не зводить рішення до вибору найбільшого Q-значення. Для нього доцільно використати Actor-Critic-побудову зі спільним кодувальним блоком стану. Після обробки вхідного вектора приховане подання розмірності 64 передається у дві вихідні голови. Actor формує логіти або ймовірності доступних дій, тобто

показує, з якою ймовірністю агент має залишити поточну фазу або виконати перемикання. Critic формує одне скалярне значення, яке оцінює корисність поточного стану незалежно від конкретної дії. У дискретному просторі phase_switch вихід Actor також має два елементи, але їхній зміст відрізняється від DQN: це не Q-оцінки, а параметри розподілу політики. Саме тому PPO навчає не окрему цінність дій, а поведінкову політику світлофорного агента [14]. Таблиця 3.9 демонструє Архітектурні відмінності DQN та PPO для світлофорного агента.

Таблиця 3.9 – Архітектурні відмінності DQN та PPO для світлофорного агента

Компонент	DQN	PPO
Вхід моделі	Вектор стану агента: фаза, черги, швидкість, затримка, щільність, pressure	Той самий вектор стану агента
Приховане подання	Енкодер ознак, прихований шар 64 нейрони, ReLU, можливий GRUCell 64	Спільний енкодер стану 64 нейрони, ReLU
Вихід моделі	Два Q-значення для дій hold і switch	Actor: дві ймовірності дій; Critic: одне значення стану
Вибір дії	Максимальне допустиме Q-значення або ε-greedy	Вибір за розподілом політики з маскуванням недопустимих дій
Оптимізатор	Adam для мінімізації похибки Q-оцінки	Adam для обмеженого оновлення політики та оцінки стану
Роль у системі	Оцінювання корисності фазових рішень	Навчання стабільної поведінкової політики агента

У PPO маска допустимих дій застосовується до виходу Actor до формування остаточного розподілу ймовірностей. Це необхідно з тієї самої причини, що й у DQN: нейронна модель не повинна мати можливості порушити фазові обмеження світлофорного контролера. Після маскування недопустимі дії отримують нульову ймовірність, а решта нормуються через

softmax. Під час навчання дія може вибиратися стохастично, що дає змогу агенту досліджувати альтернативні рішення в однакових транспортних станах. Під час оцінювання доцільно використовувати дію з найбільшою ймовірністю, щоб поведінка системи була відтворюваною. Critic у цій схемі потрібний для зменшення нестабільності навчання: він оцінює, наскільки поточний стан є сприятливим або проблемним, і допомагає відокремити випадкові коливання винагороди від справді корисних змін політики. Для міської транспортної мережі це суттєво, оскільки винагорода одного агента залежить не лише від його власної дії, а й від рішень сусідніх світлофорів [16].

3.11 Методологія навчання та підбору параметрів для DQN і PPO

Методологія навчання DQN і PPO в розроблюваній мультиагентній системі має забезпечити не лише отримання працездатних моделей, а й коректне порівняння двох різних підходів до світлофорного керування. У попередніх підрозділах було визначено архітектуру агентів, простір дій phase_switch, функцію винагороди та спільний інтерфейс взаємодії із SUMO. Тому на цьому етапі основним завданням є побудова такого експериментального контуру, у якому змінюються саме параметри навчання DQN і PPO, а не умови транспортного середовища. Це дає змогу оцінювати якість алгоритмів за однаковою мережею, однаковими маршрутами, однаковими ознаками стану та однаковою логікою винагороди [8], [16].

Експериментальна частина роботи передбачає кілька послідовних запусків навчання для кожного алгоритму. Одноразове навчання моделі не є достатнім, оскільки результати підкріплювального навчання залежать від тривалості взаємодії із середовищем, початкового накопичення досвіду, розміру навчальної вибірки, частоти оцінювання та режиму збереження проміжних версій моделі.

Для світлофорного керування ця залежність має практичний зміст. Якщо навчання є надто коротким, агент може пристосуватися лише до початкових фрагментів транспортного попиту й не навчитися реагувати на складніші стани мережі. Якщо ж кількість кроків або оновлень надмірна, покращення метрик може бути незначним порівняно з додатковими обчислювальними витратами. Через це в роботі доцільно розглядати навчання як процедуру підбору конфігурацій, а не як запуск одного фіксованого сценарію.

Для DQN основним керованим параметром є total_env_steps, який визначає загальну кількість кроків взаємодії агента із симуляційним середовищем. У межах кожного такого кроку система отримує спостереження з модулів observations.py і traffic_signal.py, формує дію через Q-мережу, застосовує маску допустимості з PhaseSwitchActionSpace, виконує крок SUMO та зберігає перехід «стан – дія – винагорода – наступний стан» для подальшого оновлення моделі.

Параметр warmup_steps задає початковий інтервал, протягом якого досвід накопичується без активного або з обмеженим оновленням ваг. Такий етап потрібний для DQN, оскільки навчання за дуже малою кількістю переходів може зробити модель чутливою до випадкових початкових ситуацій: наприклад, до короткочасної черги на одному напрямку або до випадкового розподілу транспортних засобів на перших хвилинах моделювання [13].

Параметр batch_size визначає кількість переходів, що використовуються для одного оновлення Q-мережі. Малий розмір вибірки дає частіші, але шумніші оновлення; більший розмір може стабілізувати градієнт, однак збільшує вимоги до пам’яті та часу обчислень.

Рисунок 3.11 демонструє загальну схема навчання, валідації та підбору параметрів DQN і PPO.

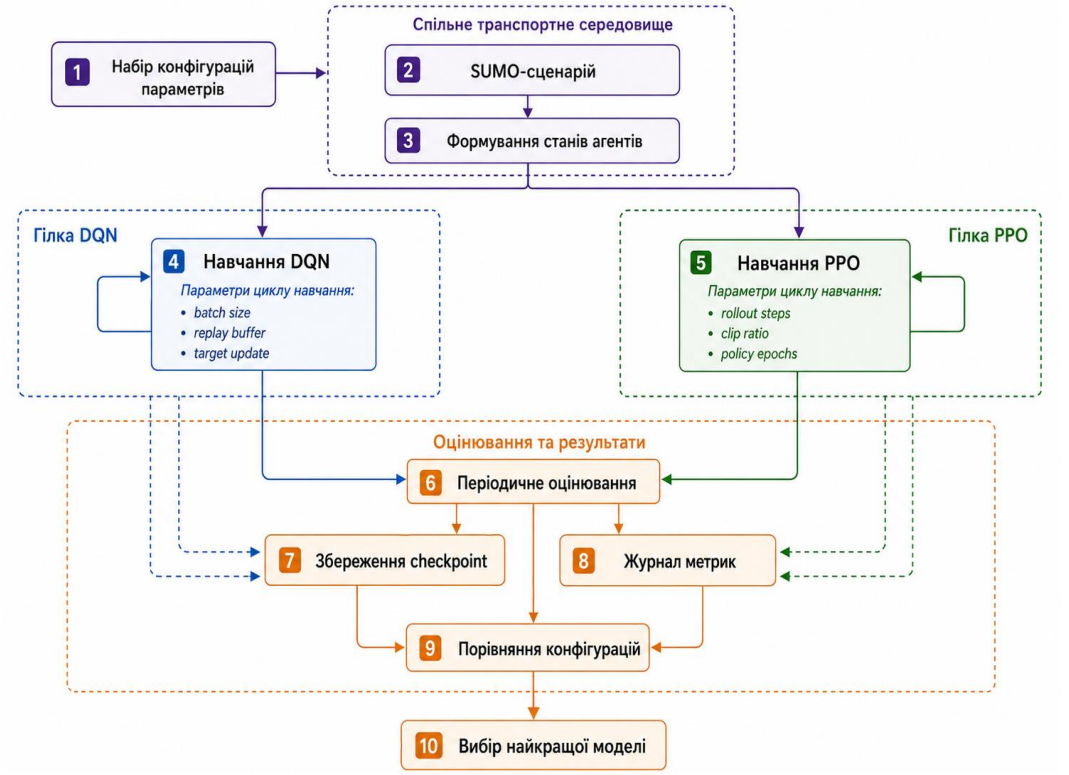


Рисунок 3.11 – Загальна схема навчання, валідації та підбору параметрів DQN і PPO

Параметри eval_interval і checkpoint_interval у DQN виконують службову, але методично важливу роль. eval_interval задає періодичність проміжного оцінювання моделі без змішування навчальної поведінки з підсумковою якістю керування. Під час такого оцінювання доцільно вимикати випадкову складову вибору дії або зменшувати її до мінімуму, щоб аналізувати не дослідницькі дії агента, а фактично сформовану політику керування. checkpoint_interval визначає, як часто зберігається стан моделі. Це дає змогу не лише відновити навчання після переривання, а й порівняти поведінку агента на різних етапах: рання модель може бути нестабільною, проміжна – демонструвати швидке зменшення черг, а пізніша – стабілізувати кількість перемикань. У наявній структурі проєкту контур оцінювання та збереження статистики узгоджується з файлами evaluate.py і run_controllers.py, де передбачено запуск контролерів,

журналювання метрик і формування CSV-результатів; приклади таких результатів доцільно подавати в додатку Г.

Для PPO логіка підбору параметрів має інший акцент, оскільки цей алгоритм оновлює не Q-значення окремих дій, а параметри політики та критика. Параметр total_updates визначає загальну кількість оновлень політики. Його роль подібна до total_env_steps у DQN, але одиниця вимірювання інша: PPO накопичує фрагмент взаємодії із середовищем, після чого виконує оптимізацію політики за зібраними даними.

Параметр rollout_steps задає довжину такого фрагмента. Якщо він занадто малий, оцінка переваг може бути нестабільною, оскільки агент бачить лише короткі наслідки перемикання фаз. Якщо rollout_steps надто великий, оновлення відбуваються рідше, а зібрані дані можуть гірше відповідати поточній політиці.

Для транспортної мережі це суттєво: наслідок дії світлофорного агента часто проявляється не миттєво, а через кілька кроків, коли випущений потік досягає сусіднього перехрестя або коли затримка накопичується на конфліктному напрямку [14], [16]. Таблиця 3.10 демонструє параметри навчання та валідації DQN і PPO.

Таблиця 3.10 – Параметри навчання та валідації DQN і PPO

Група параметрів	DQN	PPO	Методичне значення
1	2	3	4
Тривалість навчання	total_env_steps	total_updates	Визначає обсяг взаємодії моделі із середовищем або кількість оновлень політики
Початкове накопичення даних	warmup_steps	Не є основним параметром	Дає DQN достатній набір переходів перед активним навчанням
Розмір навчального фрагмента	batch_size	rollout_steps	Впливає на стабільність оновлення ваг або політики

Продовження таблиці 3.10

1	2	3	4
Проміжне оцінювання	eval_interval	eval_interval	Дозволяє відстежувати якість моделі під час навчання
Збереження моделі	checkpoint_interval	checkpoint_interval	Забезпечує відновлення навчання та аналіз проміжних станів моделі
Архітектура мережі	hidden_layers, hidden_dim, activation	hidden_layers, hidden_dim, activation	Визначає складність апроксимації функції дії або політики
Дослідження дій	epsilon_start, epsilon_end, epsilon_decay_steps	entropy_coef	Регулює баланс між пошуком нових рішень і використанням набутої політики
Стійкість градієнтного оновлення	target_update_interval, gradient_updates	gae_lambda, clip_epsilon, max_grad_norm	Обмежує різкі або нестабільні зміни моделі під час навчання
Режим транспортного попиту	demand_mode, demand_scale, demand_max_vehicles	demand_mode, demand_scale, demand_max_vehicles	Забезпечує порівняння моделей за однакових умов інтенсивності руху

Проміжне оцінювання PPO також задається параметром eval_interval, а збереження контрольних версій – параметром checkpoint_interval. Для коректного порівняння з DQN ці процедури повинні використовувати ту саму транспортну мережу, ті самі сценарії попиту та ті самі метрики: середню затримку, кількість транспортних засобів у черзі, середню швидкість, кількість зупинок, пропуск транспортних засобів і стабільність фазових перемикань. Інакше порівняння може виявитися зміщеним. Наприклад, PPO може виглядати стабільнішим лише тому, що його оцінювали на менш інтенсивному попиті, або DQN може демонструвати кращу середню чергу через інший режим формування стану. Тому в межах цієї роботи доцільно зафіксувати спільні компоненти експерименту:

сценарій SUMO, конфігураційні файли мережі й маршрутів, функцію винагороди, простір дій, маски допустимості та набір метрик. Змінюватися мають тільки параметри навчання відповідного алгоритму.

У доопрацьованій програмній реалізації методика підбору параметрів підтримується не лише окремими запусковими скриптами train_marl_dqn.py і train_marl.py, а й графічним launcher-інтерфейсом run_once_gui.py, фрагмент якого подано в додатку Д.

У ньому передбачено режими Run, Train DQN, Train PPO, Train DQN then PPO і Auto tune stages. Параметр Duration задає тривалість одного SUMO-епізоду для звичайного запуску, навчання DQN, навчання PPO та поетапного автоматичного підбору. Окремий блок Traffic demand дозволяє запускати сценарій або з оригінальним route-файлом, або з тимчасово згенерованим попитом. У другому випадку модуль demand.py формує нові route/sumocfg-файли в директорії результатів, масштабуючи кількість транспортних засобів, обмежуючи їх максимальну кількість, змінюючи часовий інтервал відправлення та, за потреби, повторюючи попит після завершення початкового route-файлу. Це дає змогу перевіряти політики не лише на фіксованій конфігурації, а й за різних рівнів завантаження мережі.Режим Auto tune stages реалізує послідовний експериментальний підбір параметрів.

Спочатку виконується screening за інтенсивністю транспортного попиту, після чого окремо перебираються архітектурні параметри DQN і PPO, крок навчання DQN, швидкість спадання epsilon, крок навчання PPO, поєднання rollout_steps і ppo_epochs, а також ваги функції винагороди DQN. Після кожного етапу найкраща конфігурація фіксується та використовується на наступному етапі, що зменшує кількість ручних запусків і робить процедуру відтворюваною. Критерій вибору кандидата задається як сума фактичного середнього часу поїздки, середнього часу очікування, середніх втрат часу та штрафу за малу кількість завершених поїздок:

$$S = \overline{T}_r + \overline{T}_w + \overline{T}_l + \frac{1000}{N_c}, \quad (3.2)$$

де S – узагальнений показник якості кандидата (score);

$\overline{T}_r$ – середній фактичний час поїздки транспортного засобу (duration_mean);

$\overline{T}_w$ – середній час очікування транспортного засобу (waitingTime_mean);

$\overline{T}_l$ – середні втрати часу під час руху (timeLoss_mean);

N_c – кількість транспортних засобів, які завершили поїздку (n_completed).

Менше значення цього показника відповідає кращому кандидату, оскільки воно одночасно враховує швидкість проходження маршруту, втрати часу та здатність системи доводити транспортні засоби до завершення поїздки. Результати auto-tuning зберігаються в директорії Output folder/auto_tuning/, зокрема у файлі auto_tuning_summary.json, а графіки етапів формуються засобами plots.py; відповідні матеріали доцільно винести в додаток Д.

Методика навчання, валідації та вибору параметрів DQN і PPO:

– підготувати спільний SUMO-сценарій, набір світлофорних агентів, простір дій phase_switch, функцію винагороди та перелік метрик оцінювання;

– визначити режим route-файлів: original для використання наявного попиту або generated для створення тимчасового попиту з параметрами demand_scale, demand_max_vehicles, depart_start, depart_end і seed;

– сформувати набір конфігурацій DQN із різними значеннями total_env_steps, warmup_steps, batch_size, buffer_capacity, learning_rate, epsilon_decay_steps, hidden_dim, activation, queue_weight, delay_weight і switch_penalty;

– сформувати набір конфігурацій PPO із різними значеннями target_global_steps, rollout_steps, ppo_epochs, minibatch_size, learning_rate, gae_lambda, clip_epsilon, hidden_dim, activation і reward_mode;

– для кожної конфігурації виконати навчання моделі з фіксацією проміжних checkpoint, журналу навчання, журналу оцінювання та покрокових сигналів середовища;

– через задані інтервали виконувати оцінювання моделі на однаковому валідаційному сценарії транспортного попиту;

– узагальнити SUMO tripinfo-output у файли tripinfo_summary.csv і tripinfo_vehicles.csv;

– порівняти конфігурації за середньою затримкою, чергами, середнім часом поїздки, середнім часом очікування, втратами часу, кількістю завершених поїздок, стабільністю перемикань і часом обчислень;

– за потреби виконати Auto tune stages, у якому параметри попиту, архітектури, кроку навчання, exploration schedule, rollout-фрагментів і ваг винагороди підбираються послідовно;

– використати відібрані моделі для підсумкового порівняння в експериментальному розділі та побудови графіків українською мовою.

Вибір найкращої конфігурації не повинен ґрунтуватися лише на одному мінімальному значенні середньої затримки. Для світлофорного керування важливо оцінювати поведінку моделі як цілісний режим керування мережею. Конфігурація, яка зменшує середню затримку, але створює часті перемикання фаз, може бути менш придатною, ніж модель із трохи гіршим середнім показником, але стабільнішою поведінкою. Аналогічно, зменшення черги на окремих перехрестях не є достатнім доказом ефективності, якщо водночас збільшується затримка на сусідніх вузлах. Тому під час вибору параметрів доцільно враховувати не тільки середні значення, а й розкид результатів між епізодами, наявність різких погіршень, кількість збережених фазових перемикань і час, потрібний для досягнення прийнятного рівня якості. Такий підхід відповідає практичній

логіці мультиагентного керування: модель має не лише навчитися покращувати винагороду, а й підтримувати передбачувану транспортну поведінку в мережі.

3.12 Сценарії експериментального дослідження та схема порівняння DQN і PPO

Експериментальне дослідження в межах розроблюваної мультиагентної системи має перевіряти не лише те, чи здатні DQN і PPO покращити окремі транспортні метрики, а й те, наскільки стабільно вони працюють за різних режимів попиту. У попередніх підрозділах було визначено архітектуру агентів, простір дій phase_switch, структуру стану, функцію винагороди та методику підбору параметрів. Тому на цьому етапі основним завданням є формування набору сценаріїв, у яких обидва алгоритми оцінюються в однаковому симуляційному контурі SUMO [8].

Для цієї роботи доцільно побудувати експериментальну перевірку на основі кількох типів транспортного попиту та кількох масштабів мережі. У доопрацьованій програмній реалізації основними робочими сценаріями launcher-а є 2x2_sumo_grid, 2x2_sumo_grid_low і 2x2_sumo_grid, які розміщені в директорії Multi-Agent for Traffic Signal Control//scenarios/sumo. Сценарій 2x2_sumo_grid_low використовується як полегшений варіант для smoke test і швидкої перевірки працездатності навчального контуру, 2x2_sumo_grid – як базовий сценарій для порівняння DQN і PPO, а 2x2_sumo_grid – як складніша мережа з більшою кількістю агентів і сильнішою взаємозалежністю суміжних світлофорних рішень. Конфігураційні параметри цих сценаріїв подано в додатку Б, а файли мережі, маршрутів і детекторів – у додатку В.

Окремий напрям експериментальної перевірки пов’язаний із генерацією транспортного попиту. На відміну від початкового варіанта, де інтенсивність руху задавалася переважно наявними route-файлами

сценарію, доопрацьована реалізація підтримує режим generated, у якому перед запуском формується тимчасовий route/sumocfg у директорії Output folder/generated_routes. Модуль demand.py дозволяє масштабувати кількість транспортних засобів через параметр Scale, задавати верхню межу Max vehicles, змінювати часовий інтервал Depart start – Depart end, обирати стабільний або випадковий спосіб відбору маршрутів і фіксувати Seed для відтворюваності.

Якщо кінець інтервалу відправлення перевищує тривалість початкового route-файлу, генератор може повторити попит після оригінального максимуму, щоб довгі епізоди не перетворювалися на порожнє моделювання після зникнення нових автомобілів. Така можливість особливо важлива для перевірки стійкості DQN і PPO до різних рівнів завантаження: низька інтенсивність виявляє схильність політики до зайвих перемикань, середня показує типовий режим роботи, а підвищена дозволяє оцінити, чи не переносить агент чергу з одного вузла на інший.

Кожний сценарій має виконуватися за однаковою процедурою, щоб результати відображали саме відмінності між алгоритмами, а не різницю в умовах запуску. Спочатку запускається базове світлофорне керування, наприклад fixed_time, sotl, greedy, max_pressure або random, доступне в контурі controllers. Після цього в тому самому сценарії оцінюються конфігурації DQN через train_marl_dqn.py або режим Train DQN у run_once_gui.py.

Далі за аналогічною логікою перевіряються конфігурації PPO через train_marl.py або режим Train PPO. Якщо потрібно забезпечити однаковий бюджет кроків для двох алгоритмів, у PPO використовується параметр target_global_steps, а в DQN – total_env_steps. Контур запуску й збереження результатів узгоджується з файлами marl_dqn.py, marl_ppo.py, tripinfo.py і plots.py; фрагменти цих матеріалів подано в додатках Г і Д. Таблиця 3.11 демонструє експериментальні сценарії перевірки DQN і PPO.

Таблиця 3.11 – Експериментальні сценарії перевірки DQN і PPO

Сценарій	Зміст транспортної ситуації	Призначення в експерименті	Очікуваний ризик для моделі
2x2_sumo_grid_low	Полегшена сітка з меншою кількістю транспортних засобів	Smoke test, перевірка запуску DQN/PPO і базова оцінка стабільності	Надмірна реакція агента на малі черги
2x2_sumo_grid	Базова сітка 2x2 з інтенсивнішим транспортним потоком	Основне порівняння DQN і PPO за однакової топології	Пристосування до локальних закономірностей малого сценарію
2x2_sumo_grid	Сітка з більшою кількістю регульованих вузлів	Перевірка масштабування мультиагентної політики	Перенесення черг між сусідніми агентами
generated demand, низький Scale	Зменшений попит, сформований тимчасовим route/sumocfg	Перевірка відсутності зайвих перемикань і шумової поведінки	Часті перемикання без реальної потреби
generated demand, підвищений Scale	Збільшена кількість транспортних засобів у тому самому сценарії	Перевірка стійкості до перевантаження	Накопичення заторів і зменшення кількості завершених поїздок

Набір метрик для всіх сценаріїв має залишатися незмінним. Основними показниками є середня затримка, середня довжина черги, середній час очікування, середній час поїздки, середня швидкість, кількість завершених поїздок і стабільність навчання. Частина цих показників підтримується модулем metrics.py, де обчислюються n_queued, mean_speed, mean_delay, average_travel_time, average_wait_time, n_exited та інші службові характеристики. Додатково доопрацьована реалізація зберігає покрокові сигнали навчання у файлах dqn_training_signals.csv і

ppo_training_signals.csv, де фіксуються mean_n_queued, mean_occupancy, mean_delay, mean_pressure, mean_reward, sum_n_queued, sum_pressure і дії агентів. Для фактичної оцінки завершених поїздок використовується SUMO tripinfo-output, який обробляється модулем tripinfo.py у файли dqn_tripinfo_summary.csv, dqn_tripinfo_vehicles.csv, ppo_tripinfo_summary.csv і ppo_tripinfo_vehicles.csv. Важливо враховувати, що tripinfo-output містить дані лише для транспортних засобів, які завершили маршрут, тому ці метрики доцільно використовувати для оцінювання якості моделі після або під час evaluation, а не як єдине джерело навчального сигналу.

Схема експериментального порівняння DQN і PPO:

– вибрати SUMO-сценарій (2x2_sumo_grid) та зафіксувати відповідні файли мережі, маршрутів і конфігурації;

– визначити режим маршруту: original або generated; для generated задати Scale, часовий інтервал відправлення, спосіб відбору маршрутів і Seed;

– запустити базове світлофорне керування й зберегти метрики в журналі результатів;

– для кожної конфігурації DQN виконати навчання, проміжне оцінювання та підсумковий тест у тому самому сценарії;

– для кожної конфігурації PPO виконати навчання, проміжне оцінювання та підсумковий тест у тому самому сценарії;

– зберегти train_log, eval_log, training_signals, checkpoint, конфігураційний JSON і SUMO tripinfo-output;

– обробити tripinfo-output у summary-таблиці завершених поїздок;

– побудувати графіки винагороди, loss, часу поїздки, часу очікування, втрат часу, завершених поїздок, черг і Scale-порівнянь;

– порівняти фінальну якість керування з базовим режимом і між алгоритмами DQN/PPO;

– узагальнити, у яких умовах DQN або PPO є доцільнішим для мультиагентного світлофорного керування.

Окремо потрібно враховувати, що найкраща фінальна метрика не завжди означає найкращий алгоритм для практичного використання. DQN може швидше знаходити прийнятну політику в дискретному просторі hold/switch, оскільки безпосередньо оцінює корисність кожної дії.

Проте в напружених або нестаціонарних сценаріях такий підхід може бути чутливішим до шуму винагороди й до випадкових змін у розподілі черг [13]. PPO, навпаки, може потребувати більшої кількості оновлень або довших rollout-фрагментів, але його Actor-Critic-структура дає змогу обмежувати різкі зміни політики та підтримувати плавніше навчання [14]. Для світлофорного керування ця відмінність має прямий транспортний зміст: нестабільна політика може створювати часті перемикання фаз, тоді як надто повільна політика може не встигати реагувати на раптове зростання попиту.

Узагальнене порівняння має виконуватися за чотирма групами критеріїв. Перша група – якість керування, що відображається через затримку, черги, час очікування, час поїздки, швидкість і завершені поїздки. Друга група – стабільність навчання, яка характеризується коливанням винагороди, розкидом метрик між епізодами та наявністю різких погіршень після оновлення моделі. Третя група – швидкість збіжності, тобто кількість кроків середовища або оновлень політики, потрібних для досягнення прийнятного рівня керування. Четверта група – обчислювальна складність, що включає час навчання, потребу в пам’яті, частоту збереження моделей і витрати на оцінювання.

Такий підхід узгоджується з мультиагентною природою задачі, оскільки політика повинна бути не лише ефективною в середньому, а й передбачуваною для всієї мережі [16].

3.13 Висновки до розділу

У розглянутих підрозділах сформовано цілісну основу для реалізації та подальшого експериментального дослідження мультиагентної системи світлофорного регулювання. Запропонована архітектура поєднує мікроскопічне симуляційне середовище SUMO, локальних світлофорних агентів, модулі збору й агрегування даних, безпечний простір керуючих дій, функцію винагороди з локально-мережевою складовою, а також два альтернативні навчальні контури на основі DQN і PPO. Головною особливістю цієї системи є те, що методи підкріплювального навчання застосовуються не ізольовано, а в межах транспортно обґрунтованого контуру, де адаптивність агента поєднується з фазовою безпекою, масштабованістю, відтворюваністю експериментів і можливістю об’єктивного порівняння різних алгоритмів керування.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ, ЕКСПЕРИМЕНТАЛЬНЕ НАЛАШТУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ

4.1 Програмна реалізація мультиагентної системи світлофорного регулювання

Програмна реалізація мультиагентної системи світлофорного регулювання в четвертому розділі розглядається не як повторення архітектурної схеми, визначеної в третьому розділі, а як перевірка того, що запропонований контур справді доведений до експериментального інструментарію. Якщо в розділі 3 було обґрунтовано подання перехресть як агентів, структуру стану, простір дій hold/switch, винагороду та логіку навчання DQN і PPO, то в цьому підрозділі акцент перенесено на програмні файли, режими запуску, формування результатів і керування серіями експериментів.

Реалізаційною основою є директорія Multi-Agent for Traffic Signal Control, а фактичні результати запусків збережені в директорії Test_Sumo, де містяться конфігурації, журнали навчання, checkpoint-файли, SUMO tripinfo-output, підсумкові CSV-таблиці та графіки українською мовою.

Центральним прикладним компонентом системи є файл run_once_gui.py, який виконує роль графічного launcher-інтерфейсу для керування експериментами. Його призначення полягає в тому, щоб об’єднати окремі сценарії запуску в один керований інтерфейс: вибір симулятора, сценарію, контролера, тривалості епізоду, режиму SUMO GUI, папки результатів, режиму формування транспортного попиту та параметрів навчання.

Такий підхід має практичне значення для експериментального розділу, оскільки порівняння алгоритмів не повинно залежати від ручного редагування команд або випадкового пропуску параметрів.

У launcher-і передбачено режим Run для запуску класичного контролера через run_once.py, режим Train DQN для навчання Parameter Sharing Double DQN через train_marl_dqn.py, режим Train PPO для навчання PPO через train_marl.py, режим Train DQN then PPO для послідовного запуску двох моделей і режим Auto tune stages для автоматичного поетапного підбору параметрів. Фрагмент реалізації цих режимів наведено в додатку Д, а фрагменти базових модулів взаємодії агента із середовищем подано в додатку А.

Рисунок 4.1 демонструє програмну структуру експериментального комплексу мультиагентного світлофорного регулювання.

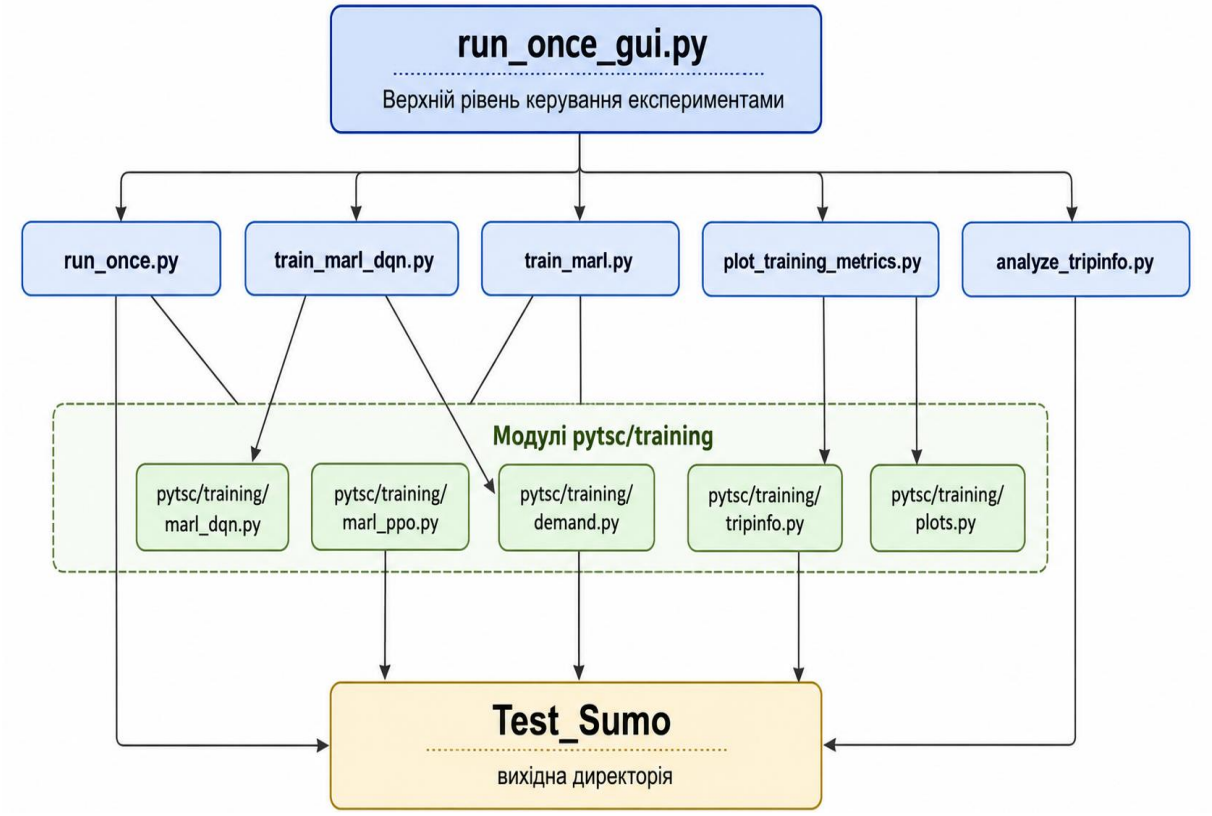


Рисунок 4.1 – Програмна структура експериментального комплексу мультиагентного світлофорного регулювання

Таблиця 4.1 демонструє призначення основних програмних файлів експериментального комплексу.

Таблиця 4.1 – Призначення основних програмних файлів експериментального комплексу

Компонент	Призначення в реалізації	Пов’язані результати
run_once_gui.py	Графічний launcher для вибору режиму запуску, сценарію, контролера, тривалості, папки результатів і параметрів навчання	Командний запуск експериментів, серії Scale, auto-tuning
run_once.py	Запуск одного сценарію з класичним контролером	Базові оцінки для порівняння з MARL
train_marl_dqn.py	Запуск Parameter Sharing Double DQN	dqn_train_log.csv, dqn_eval_log.csv, dqn_training_signals.csv, dqn_checkpoints
train_marl.py	Запуск PPO у спільному SUMO-контурі	ppo_train_log.csv, ppo_eval_log.csv, ppo_training_signals.csv, ppo_checkpoints
/training/demand.py	Генерація route/sumocfg-файлів із заданим масштабом попиту	generated_routes/*_generated.rou.xml, *_demand.json
/training/tripinfo.py, analyze_tripinfo.py	Обробка SUMO tripinfo-output	tripinfo_summary.csv, tripinfo_vehicles.csv
/training/plots.py, plot_training_metrics.py	Побудова графіків навчання, оцінювання, Scale-порівнянь та auto-tuning	директорія plots_ua

Практична логіка реалізації полягає в тому, що всі режими використовують однаковий транспортний контур SUMO, але різні керуючі рівні. У режимі Run файл run_once.py створює середовище, вибирає контролер із множини fixed_time, greedy, max_pressure, random, sotl або іншого доступного контролера, передає параметри тривалості та за потреби формує тимчасовий попит через generate_sumo_demand(). Цей режим

потрібний не тільки для швидкої перевірки сценарію, а й для отримання базового рівня, з яким надалі можна порівнювати навчальні моделі. У режимах Train DQN і Train PPO launcher формує команди з однаковими параметрами сценарію, тривалості епізоду, масштабу попиту та папки результатів, але передає їх до різних навчальних скриптів. Для DQN використовуються параметри total_env_steps, warmup_steps, batch_size, buffer_capacity, epsilon_decay_steps, queue_weight, delay_weight, switch_penalty, eval_interval і checkpoint_interval. Для PPO задаються target_global_steps, rollout_steps, ppo_epochs, minibatch_size, gae_lambda, clip_epsilon, entropy_coef, value_coef і reward_mode. Такий розподіл відповідає теоретичній відмінності між ціннісним навчанням DQN і оптимізацією політики PPO [13, 14], але на рівні реалізації обидва алгоритми залишаються підключеними до однакової транспортної моделі SUMO [8].

Окрему роль у системі виконує модуль генерації попиту training/demand.py. Він дає змогу не обмежуватися одним route-файлом, а формувати серії експериментів із різним значенням demand_scale, часовим інтервалом відправлення транспортних засобів, режимом вибірки та фіксованим seed. У директорії dqn_scale_0_25/generated_routes збережено приклад такого запуску: файл dqn_2x2_sumo_grid_generated_demand.json фіксує, що вихідний route-файл сценарію 2x2_sumo_grid містив 5662 транспортні засоби, а для масштабу 0,25 було сформовано 1416 транспортних засобів у проміжку приблизно від 0 до 3597,46 с.

Відповідний підхід використано і для PPO. Завдяки цьому експериментальна частина не зводиться до одного випадкового запуску, а містить серії для Scale 0,25, 0,5, 0,75 і 1,0, що підтверджується папками dqn_scale_* і ppo_scale_* у директорії для збереження результатів. Конфігураційні параметри сценаріїв доцільно пов’язувати з додатком Б, а фрагменти генерації попиту й launcher-режимів – з додатком Д.

Узагальнена послідовність роботи програмного комплексу під час експериментального запуску:

1) користувач у run_once_gui.py вибирає симулятор, сценарій, режим роботи, тривалість епізоду, параметри попиту та папку результатів;

2) launcher формує команду для run_once.py, train_marl_dqn.py або train_marl.py залежно від вибраного режиму;

3) за потреби модуль demand.py створює згенеровані route/sumocfg-файли для SUMO;

4) навчальний або класичний контролер взаємодіє із SUMO, отримує стани світлофорних агентів, застосовує дії та збирає транспортні метрики;

5) після заданих інтервалів зберігаються журнали навчання, оцінювання, покрокові сигнали середовища та checkpoint-файли моделей;

6) SUMO tripinfo-output обробляється у tripinfo_summary.csv і tripinfo_vehicles.csv;

7) модуль plot_training_metrics.py формує графіки навчання, порівнянь DQN/PPO, Scale-серій та auto-tuning у директорії plots_ua.

Фактична структура результатів у Test_Sumo підтверджує, що програмна система підтримує не лише запуск моделі, а й повний цикл експериментального аналізу.

Для DQN у кожній Scale-директорії зберігаються dqn_config.json, dqn_train_log.csv, dqn_eval_log.csv, dqn_training_signals.csv, папка dqn_checkpoints, згенеровані маршрути та оброблені tripinfo-файли. Для PPO аналогічно створюються ppo_config.json, ppo_train_log.csv, ppo_eval_log.csv, ppo_training_signals.csv, checkpoint моделі та підсумки поїздок. Наприклад, для dqn_scale_0_25 підсумковий файл dqn_tripinfo_summary.csv містить 50869 завершених записів поїздок, зібраних із 38 XML-файлів tripinfo, із середньою тривалістю 97,14 с, середнім часом очікування 6,61 с і середніми втратами часу 22,76 с.

Для ppo_scale_0_25 за тієї самої кількості tripinfo-файлів зафіксовано 50868 завершених записів, середню тривалість 96,69 с, середній час

очікування 6,13 с і середні втрати часу 22,30 с. Ці значення не слід інтерпретувати як остаточний висновок щодо переваги алгоритму, оскільки подальші підрозділи мають врахувати всі масштаби попиту, але вони показують, що реалізація зберігає достатні дані для кількісного порівняння.

Режим Auto tune stages є важливою частиною експериментального інструментарію, оскільки автоматизує підбір параметрів і зменшує ризик суб'єктивного вибору конфігурації. У файлі run_once_gui.py цей режим будує послідовний план: пошук архітектури DQN і PPO, вибір learning rate, налаштування epsilon_decay_steps для DQN, підбір rollout_steps і ppo_epochs для PPO, перевірка ваг винагороди DQN і фінальна оцінка стійкості за різними масштабами попиту. Підсумок цього процесу збережено у файлі Test_Sumo/auto_tuning/auto_tuning_summary.json. У ньому зафіксовано сценарій 2x2_sumo_grid, тривалість епізоду 3600 с, завершений статус підбору та вибрані параметри: для DQN – hidden_dim=256, activation=leaky_relu, learning_rate=0.001, epsilon_decay_steps=10000; для PPO – hidden_dim=256, activation=tanh, learning_rate=0.0003, rollout_steps=256, ppo_epochs=4. Критерій вибору реалізовано як сума середньої тривалості поїздки, середнього часу очікування, середніх втрат часу та штрафного доданка 1000 / n_completed, тобто система оцінює не тільки навчальну винагороду, а й транспортні наслідки, отримані з SUMO tripinfo-output.

Побудова графіків винесена в окремий модуль, що також має методичне значення. Файл plot_training_metrics.py звертається до /training/plots.py і формує візуалізації за журналами DQN/PPO, зокрема для винагороди, функції втрат, часу поїздки, часу очікування, втрат часу, кількості завершених поїздок, черг і порівняння Scale-серій. У директорії Test_Sumo/plots_ua наявні графіки для етапів auto-tuning, фінального порівняння моделей і параметрів. Це відокремлення є доцільним: навчальні скрипти відповідають за генерацію результатів, tripinfo.py – за їхнє агрегування, а plots.py – за подання у формі, придатній для подальшого

аналізу в дипломній роботі. Для задачі світлофорного регулювання така структура важлива, оскільки метрики машинного навчання, зокрема loss, не можуть самостійно показати транспортну якість керування. Потрібно зіставляти їх із середньою затримкою, чергами, завершеними поїздками й часом очікування, що відповідає практичній природі систем інтелектуального транспортного керування [8], [16].

4.2 Підготовка SUMO-сценаріїв, генерація транспортного попиту та baseline-контролери

Підготовка експериментів у четвертому розділі спирається на модельні положення, визначені раніше: транспортна мережа задає кількість світлофорних агентів, топологію їхньої взаємодії, склад смуг, маршрути транспортних засобів, фазову структуру й набір детекторів. У підрозділі 4.1 було показано програмний контур запуску, навчання та збереження результатів, тому на цьому етапі доцільно конкретизувати, які саме SUMO-сценарії використовуються як експериментальна основа, як формується транспортний попит і які класичні контролери застосовуються для порівняння з DQN та PPO.

У програмній директорії /scenarios/sumo підготовлено основний сценарій 2x2_sumo_grid. Він має однакову логіку побудови синтетичної мережі, але різняться кількістю регульованих вузлів і рівнем транспортного навантаження. Сценарій 2x2_sumo_grid має чотири світлофорні агенти, однак його route-файл містить 5662 транспортні засоби, тому він використовується як основний робочий сценарій для порівняння DQN і PPO. Сценарій має тривалість моделювання 3600 с, мінімальний зелений час 5 с, максимальний зелений час 60 с і жовтий сигнал 5 с, що зафіксовано у відповідних config.yaml та sumocfg-файлах. Конфігураційні параметри цих сценаріїв пов’язані з матеріалами додатка Б, а сам механізм запуску через SUMO відповідає підходу мікроскопічного транспортного моделювання [8].

У реалізованій системі передбачено два режими використання маршрутних файлів. Режим original запускає сценарій із вихідним route-файлом, указаним у sumocfg, без зміни кількості транспортних засобів і часу їхнього відправлення. Такий режим потрібний для перевірки базової коректності сценарію та відтворення початкових умов мережі. Режим generated, навпаки, перед запуском створює тимчасові route/sumocfg-файли в директорії Output folder/generated_routes. Саме цей режим використано у збережених серіях dqn_scale_* і ppo_scale_*, оскільки він дає змогу запускати DQN і PPO в однаковому середовищі, але за різних рівнів транспортного навантаження. Для основного сценарію 2x2_sumo_grid у директорії Test_Sumo зафіксовано чотири масштаби попиту: Scale=0.25, Scale=0.5, Scale=0.75 і Scale=1.0. Відповідні generated route-файли містять 1416, 2831, 4246 і 5662 транспортні засоби. Однакові значення масштабу, однаковий seed=0 і однаковий режим відбору every_n використано як для DQN, так і для PPO, що забезпечує коректність порівняння між алгоритмами.

Генерація попиту реалізована у файлі demand.py і має не лише допоміжне, а й методичне значення. Параметр Scale змінює кількість транспортних засобів відносно вихідного route-файлу; Max vehicles може обмежувати максимальну кількість відібраних автомобілів; Depart start і Depart end задають часовий інтервал появи транспортних засобів; Sampling визначає спосіб відбору записів із вихідного route-файлу; Seed забезпечує відтворюваність випадкової вибірки; Rescale departs дає змогу стискати або розтягувати часи відправлення у задане вікно. У фактичних експериментах, збережених у визначеній для цього папці, застосовано стабільний режим every_n, за якого транспортні засоби відбираються рівномірно з вихідного потоку. Це важливо для порівняння DQN і PPO: різниця в результатах не повинна пояснюватися випадковою зміною складу маршрутів. Наприклад, файл demand.json фіксує, що з вихідних 5662 транспортних засобів було сформовано 1416, а часовий інтервал відправлення залишився в межах

приблизно 0–3597,46 с. Для Scale=1.0 кількість транспортних засобів дорівнює 5662, тобто generated-режим відтворює повний вихідний попит, але водночас зберігає єдиний формат метаданих для подальшої обробки.

Практичний результат такої підготовки полягає в тому, що транспортне навантаження стало керованим параметром експерименту. У файлах dqn_config.json і ppo_config.json для кожної Scale-директорії збережено однакові параметри сценарію, тривалість епізоду 3600 с, режим generated, значення demand_scale, спосіб вибірки every_n і demand_seed=0. Це дає змогу зіставляти не окремі випадкові запуски, а впорядковані серії, у яких поступово зростає складність транспортної ситуації. Збережені підсумки tripinfo_summary.csv підтверджують очікувану тенденцію: зі збільшенням масштабу попиту зростають середня тривалість поїздки, середній час очікування та середні втрати часу.

Наприклад, для DQN середній час очікування змінюється від 6,61 с при Scale=0.25 до 45,16 с при Scale=1.0, а для PPO – від 6,13 с до 118,82 с. Ці значення не є остаточним висновком щодо переваги конкретного алгоритму, оскільки їх потрібно аналізувати разом із кількістю завершених поїздок, параметрами навчання й поведінкою моделі на проміжних етапах. Однак вони показують, що підготовлена схема попиту справді створює різні рівні навантаження, а не формально дублює один і той самий сценарій.

Для порівняння з нейромережевими моделями в системі передбачено набір baseline-контролерів: fixed_time, sotl, greedy, max_pressure і random. Вони реалізовані у файлі controllers.py, а їх вибір доступний через run_once.py і launcher-інтерфейс run_once_gui.py, фрагменти якого подано в додатку Д. Контролер fixed_time відтворює фіксоване фазове керування з наперед заданою тривалістю зеленого сигналу, тому є найближчим до традиційного світлофорного плану. sotl використовує адаптивну логіку Self-Organizing Traffic Lights: рішення про перемикання залежить від накопичення транспортних засобів на червоних і зелених напрямках, а також від мінімального часу зеленої фази. greedy вибирає доступну фазу з

найбільшою локальною чергою, тобто прагне швидко обслуговувати найбільш завантажений напрямок без довгострокового прогнозу. max_pressure орієнтується на різницю між вхідним і вихідним навантаженням для фази та є сильним класичним baseline для мережевого світлофорного керування. random не оптимізує транспортну метрику й використовується як нижня контрольна межа, що дає змогу відокремити справді осмислену політику від випадкових фазових рішень.

Baseline-контролери в цій роботі не розглядаються як самостійний теоретичний результат, оскільки їхня роль є експериментальною. Вони створюють контрольні режими, з якими порівнюються DQN і PPO після навчання. Це принципово для оцінювання мультиагентної системи: нейромережева модель може демонструвати зміну винагороди або зменшення функції втрат, але такі внутрішні показники не доводять покращення транспортного керування. Політика DQN має перевірятися не лише як апроксимація Q-значень [13], а PPO – не лише як стабілізована оптимізація політики [14], а як практичний контролер, що зменшує час поїздки, очікування, втрати часу й черги порівняно з простішими підходами.

Через це baseline-запуски повинні використовувати ті самі SUMO-сценарії, ті самі route-файли або generated route-файли, ту саму тривалість епізоду та той самий контур збору tripinfo-output. Лише за таких умов різниця між результатами може бути пов’язана з логікою керування, а не з різними початковими умовами експерименту.

4.3 Реалізація навчання моделей DQN і PPO в мультиагентному середовищі

Реалізація навчання моделей DQN і PPO у програмній системі виконує роль зв’язувальної ланки між модельними положеннями третього розділу та фактичними експериментальними результатами, збереженими в директорії Test_Sumo. Якщо раніше було визначено структуру стану світлофорного

агента, бінарний простір дій hold/switch, безпечне перемикання фаз і винагороду на основі черги, затримки та штрафу за зміну сигналу, то в цьому підрозділі ці елементи розглядаються як реалізовані навчальні контури. Основними файлами є marl_dqn.py і marl_ppo.py, а фрагменти відповідного програмного інструментарію наведено в додатку Д.

У DQN-контурі реалізовано варіант Parameter Sharing Double DQN. Це означає, що всі світлофорні об’єкти залишаються окремими агентами з власними локальними станами, але для оцінювання дій використовується одна спільна Q-мережа SharedQNetwork. Така організація має практичне значення для сіткового сценарію 2x2_sumo_grid: чотири регульовані перехрестя не потребують чотирьох незалежних моделей, оскільки їхні спостереження мають однакову структуру, а дії описуються спільним дискретним простором. У файлі marl_dqn.py стан кожного агента формується як конкатенація phase_id, часу перебування на фазі, кількості транспортних засобів у черзі, зайнятості, середньої затримки та тиску. Отже, навчальна модель не працює з абстрактним станом, а отримує безпосередньо ті локальні ознаки транспортного потоку, які були обґрунтовані в підрозділах 3.4, 3.7 і 3.8. Простір дій також збережено без зміни: дія 0 означає утримання поточної фази, а дія 1 – перехід до наступної дозволеної фази через механізм безпечного фазового контролю.

Навчання DQN побудоване як послідовність взаємодій із середовищем SUMO, збереження переходів у ReplayBuffer та оновлення параметрів Q-мережі за мініпакетами. У буфер записуються стан агента, вибрана дія, винагорода, наступний стан, ознака завершення епізоду, маска допустимих дій і маска допустимих дій для наступного стану. Це важливо саме для світлофорного керування, оскільки модель не може вибирати дію тільки за найбільшим Q-значенням: якщо перемикання фази в поточний момент заборонене мінімальним зеленим часом або іншими обмеженнями, відповідна дія маскується. У реалізації це виконано через masked_fill(action_mask <= 0, -1e9), після чого вибір дії здійснюється серед

допустимих альтернатив. Для дослідження використовується ε-greedy стратегія: на початку навчання ймовірність випадкової допустимої дії дорівнює 1.0, а після спадання за epsilon_decay_steps=10000 досягає значення 0.05.

У збережених конфігураціях dqn_scale_1/dqn_config.json зафіксовано total_env_steps=20000, batch_size=64, buffer_capacity=50000, target_update_interval=500, gamma=0.99, learning_rate=0.0005, optimizer=adam, eval_interval=2000 і checkpoint_interval=5000.

Функція винагороди в DQN-контурі є програмною конкретизацією критерію, розробленого в підрозділі 3.8, а не новою теоретичною формулою. У методі _compute_rewards() для кожного світлофорного агента обчислюється штрафна оцінка формула (3.1). У фактичних експериментах для Scale-серій використано queue_weight=1.0, delay_weight=1.0 і switch_penalty=0.1, що підтверджено файлами dqn_config.json. Такий сигнал прямо пов’язує навчання з транспортними наслідками дії: накопичення черги збільшує штраф, середня затримка відображає втрати часу на підходах до перехрестя, а доданок switch_penalty обмежує надмірно часті перемикання. Double DQN-оновлення реалізовано через окремі online_net і target_net: наступна дія вибирається за онлайн-мережею, але її Q-значення береться з цільової мережі. Це зменшує ризик переоцінювання дій, що є суттєвим для дискретного простору світлофорних рішень [13].

PPO-контур реалізовано у файлі marl_ppo.py як actor-critic MARL trainer зі спільною моделлю SharedActorCritic. На відміну від DQN, ця модель не формує Q-оцінки для дій, а повертає два виходи: policy_head задає логіти політики, а value_head оцінює стан. Дії вибираються з категоріального розподілу після застосування тієї самої маски допустимих дій, тому PPO також не може порушити фазові обмеження середовища. У режимі оцінювання використовується детермінований вибір дії за найбільшим допустимим логітом. Така реалізація відповідає положенням підрозділів 3.9 і 3.10: PPO розглядається як політичний підхід із Actor-Critic-

архітектурою, але працює в тому самому SUMO-контурі, що й DQN, з тим самим сценарієм, простором дій і механізмом формування спостережень [14].

Принципова відмінність PPO полягає в організації навчальних даних. Замість довготривалого replay buffer використовується RolloutBuffer, у який записуються фрагменти взаємодії поточної політики із середовищем: спостереження, маски дій, вибрані дії, логарифми ймовірностей, винагороди, оцінки критика та ознаки завершення епізоду. Після збору rollout-фрагмента обчислюються returns і advantages за допомогою GAE з параметрами gamma=0.99 і gae_lambda=0.95. Далі модель оновлюється протягом кількох епох за PPO-цільовою функцією з clipping: у конфігурації ppo_scale_1/ppo_config.json зафіксовано rollout_steps=256, ppo_epochs=4, minibatch_size=64, clip_epsilon=0.2, value_coef=0.5, entropy_coef=0.01, max_grad_norm=0.5, target_global_steps=20000 і learning_rate=0.0003. Наявність entropy bonus підтримує дослідження альтернативних фазових рішень, а gradient clipping обмежує різкі зміни параметрів, що важливо для стабільності навчання в симуляційному середовищі з кількома агентами.

У PPO додатково передбачено вибір режиму винагороди через параметр reward_mode. Якщо встановлено local, кожен агент отримує власну винагороду з train_env.get_rewards(). Якщо використовується інший режим, глобальна винагорода середовища розподіляється між усіма агентами як спільний сигнал. У збережених Scale-експериментах використано reward_mode="local", що узгоджується з локальною природою станів світлофорних агентів і дає змогу оцінити поведінку кожного перехрестя через його безпосередні транспортні наслідки. Водночас сама наявність глобального режиму є важливою для подальших експериментів, оскільки дає змогу перевіряти, чи покращується координація агентів за мережевою оцінкою, а не лише за локальним зменшенням черги. Такий підхід відповідає мультиагентній постановці задачі, у якій локально корисна дія може переносити затор до сусіднього вузла [16].

Обидва навчальні контури підключені до однакової системи експериментального журналювання. Для DQN зберігаються dqn_train_log.csv, dqn_eval_log.csv, dqn_training_signals.csv, checkpoint-файли в dqn_checkpoints і підсумки SUMO tripinfo-output. Для PPO створюються аналогічні файли ppo_train_log.csv, ppo_eval_log.csv, ppo_training_signals.csv, checkpoint у ppo_checkpoints і підсумкові tripinfo-таблиці.

У dqn_training_signals.csv додатково фіксується поточне значення epsilon та середня частка перемикань mean_switched, тоді як у ppo_training_signals.csv основний акцент зроблено на номері оновлення, глобальному кроці, чергах, зайнятості, затримці, тиску, винагороді та фактичних діях агентів. Для Scale=1.0 DQN сформував checkpoint-и на кроках 5000, 10000, 15000 і 20000, а PPO зберіг фінальний checkpoint checkpoint_00020.pt, що відповідає 20 оновленням і 20000 глобальним крокам.

Фактичні результати підтверджують, що DQN і PPO порівнюються в одному експериментальному контурі, але демонструють різну поведінку зі зростанням транспортного попиту.

За Scale=0.25 підсумкові tripinfo-таблиці показують близькі значення: для DQN середня тривалість поїздки становить 97,14 с, середній час очікування – 6,61 с, середні втрати часу – 22,76 с; для PPO відповідні значення дорівнюють 96,69 с, 6,13 с і 22,30 с.

За Scale=1.0 різниця стає помітнішою: DQN має середню тривалість 189,57 с, середній час очікування 45,16 с і середні втрати часу 119,19 с, тоді як PPO у відповідній серії має 248,23 с, 118,82 с і 174,49 с.

Ці значення не слід тлумачити ізольовано від кількості завершених поїздок і налаштувань навчання, однак вони показують, що реалізоване журналювання забезпечує достатню основу для кількісного аналізу ефективності моделей.

4.4 Автоматичний підбір параметрів DQN і PPO та організація експериментальних запусків

Автоматичний підбір параметрів у четвертому розділі виконує роль проміжного етапу між реалізацією навчальних контурів DQN/PPO та їхнім фінальним порівнянням за транспортними метриками. У підрозділі 3.11 було обґрунтовано, що одноразовий запуск алгоритму не дає достатньої підстави для висновку про його ефективність, оскільки результати підкріплювального навчання залежать від тривалості взаємодії із середовищем, темпу оновлення моделі, параметрів дослідження дій і структури нейронної мережі. У програмній реалізації це положення конкретизовано через launcher-інтерфейс run_once_gui.py, у якому передбачено окреме меню Training settings, навчальні пресети та режим Auto tune stages. Фрагменти цього інструментарію наведено в додатку Д, а зведені результати автоматичного підбору доцільно подати в додатку Е, оскільки останніми наявними додатками є А, Б і Д.

Меню Training settings використовується як єдина точка задання параметрів навчання для двох алгоритмів. Для DQN у ньому визначаються total_env_steps, warmup_steps, batch_size, buffer_capacity, train_frequency, gradient_updates, target_update_interval, кількість і розмір прихованих шарів, функція активації, learning_rate, gamma, параметри ε-greedy-дослідження, ваги компонентів винагороди, eval_interval і checkpoint_interval.

Для PPO аналогічно задаються target_global_steps, total_updates, rollout_steps, ppo_epochs, minibatch_size, параметри actor-critic-мережі, learning_rate, gamma, gae_lambda, clip_epsilon, коефіцієнти value/entropy-складових, max_grad_norm, режим винагороди, частота оцінювання та збереження контрольних версій.

У межах цього експерименту основне значення мають не всі перелічені поля, а ті, що безпосередньо визначають бюджет навчання й відтворюваність запуску: для DQN це total_env_steps, warmup_steps,

batch_size, eval_interval і checkpoint_interval, а для PPO – total_updates, rollout_steps, eval_interval і checkpoint_interval. Саме ці параметри задають, скільки разів модель взаємодіє із SUMO, коли починається оновлення мережі, яким є розмір навчальної вибірки, як часто виконується проміжна оцінка та в які моменти зберігаються checkpoint-файли.

Для практичного керування обчислювальними витратами в launcher-інтерфейсі реалізовано три навчальні пресети: Smoke test, Baseline і Long research run. Smoke test призначений для короткої технічної перевірки працездатності контуру: він дає змогу швидко переконатися, що SUMO запускається, route-файли генеруються, модель отримує стани агентів, а журнали навчання створюються без помилок. Baseline використовується як основний дослідницький режим для порівнюваних запусків із помірним бюджетом навчання. Long research run орієнтований на триваліший фінальний експеримент, коли вже перевірено коректність сценарію, папки результатів і параметрів попиту. Така організація важлива для експериментального розділу: вона відокремлює технічну валідацію від дослідницького запуску й зменшує ризик того, що фінальне порівняння DQN і PPO буде виконано на випадково вибраних або неперевірених параметрах.

Режим Auto tune stages запускається лише за узгоджених умов експерименту: у launcher-і має бути вибрано Simulator backend = sumo, режим маршрутів Route = generated і задано папку Output folder. Це обмеження не є суто технічною перевіркою. Автоматичний підбір має порівнювати кандидатів у однаковому середовищі, тому фіксуються сценарій 2x2_sumo_grid, тривалість епізоду 3600 с, generated-режим формування маршрутів і спільний контур збору tripinfo-output. У файлі /auto_tuning/auto_tuning_summary.json зафіксовано, що auto-tuning завершено успішно, а результати кожного кандидата збережено в окремій піддиректорії /auto_tuning/.

Для DQN і PPO створюються власні каталоги етапів, зокрема 01_dqn_architecture, 01_ppo_architecture, 02_dqn_learning_rate, 02_dqn_epsilon_decay, 03_ppo_learning_rate, 03_ppo_rollout_epochs, 04_dqn_reward_weights, 05_final_dqn_scale_robustness і 05_final_ppo_scale_robustness. У кожній такій директорії містяться конфігурації JSON, CSV-журнали навчання й оцінювання, підсумки tripinfo та checkpoint-файли, що забезпечує можливість повторної перевірки отриманого рішення.

Критерій вибору найкращого кандидата в auto-tuning не збігається з функцією винагороди, за якою навчається агент. Він використовується після запуску кандидата для порівняння транспортних наслідків керування за агрегованими SUMO-метриками. У програмі цей критерій має вигляд формула (3.2).

Менше значення score відповідає кращому кандидату. Перевага такого критерію полягає в тому, що він поєднує якість руху та здатність системи доводити транспортні засоби до завершення маршруту. Оцінювання лише за середнім часом очікування могло б приховати випадки, коли частина транспорту не завершує поїздку або накопичується в мережі, тоді як доданок 1000 / n_completed м’яко віддає перевагу конфігураціям із більшою кількістю завершених поїздок за близьких значень часових метрик. Це узгоджується з підходом до оцінювання транспортного керування через сукупність показників, а не через одну ізольовану метрику [8].

На першому етапі автоматичного підбору перевірялася архітектура нейронних моделей. Для DQN і PPO перебиралися значення hidden_dim 64, 128 і 256 у поєднанні з функціями активації relu, leaky_relu і tanh. За результатами auto_tuning_summary.json для DQN найкращим кандидатом стала конфігурація hidden_256_leaky_relu: вона дала score = 184,724, середню тривалість поїздки 120,913 с, середній час очікування 18,138 с, середні втрати часу 45,667 с і 140985 завершених поїздок. Для PPO на архітектурному етапі найкращою стала конфігурація hidden_256_tanh із

score = 207,033, середньою тривалістю 130,418 с, очікуванням 21,267 с, втратами часу 55,340 с і 139968 завершеними поїздками. Отриманий результат показує, що в заданому сценарії менші мережі не забезпечили такого самого балансу між часовими метриками й пропуском, тоді як ширина 256 дала кращу здатність апроксимувати залежність між локальним станом перехрестя та керуючою дією. Водночас різні функції активації для DQN і PPO є очікуваними, оскільки DQN апроксимує Q-значення дискретних дій [13], а PPO навчає стохастичну actor-critic-політику з обмеженням зміни розподілу дій [14].

Наступні етапи уточнювали параметри навчання, які найбільше впливають на стабільність і темп збіжності моделей. Для DQN перевірялися learning_rate = 0.0001, 0.0003, 0.0005 і 0.001; найкращим стало значення 0.001, для якого score зменшився до 182,776. Після цього було перевірено epsilon_decay_steps 5000, 10000 і 20000, причому кращим залишився варіант 10000. Це означає, що надто швидке згортання дослідження дій може передчасно закріпити політику на локально вигідних рішеннях, а надто повільне – довше підтримувати випадковість у фазових діях.

Для PPO перевірялися learning_rate = 0.0001, 0.0003 і 0.0005; найкращим залишилося значення 0.0003. Додатково порівнювалися комбінації rollout_steps і ppo_epochs: 128/4, 256/4, 512/4 і 256/8. Найкращою стала конфігурація rollout_steps = 256, ppo_epochs = 4, що відповідає помірній довжині rollout-фрагмента без надмірного багаторазового проходу по тих самих даних. У підсумку автоматичний підбір зафіксував для DQN hidden_dim=256, activation=leaky_relu, learning_rate=0.001, epsilon_decay_steps=10000, queue_weight=1.0, delay_weight=1.0, switch_penalty=0.1, а для PPO – hidden_dim=256, activation=tanh, learning_rate=0.0003, rollout_steps=256, ppo_epochs=4.

Окремий зміст має етап перевірки ваг винагороди DQN. У ньому порівнювалися збалансований варіант, конфігурація з підсиленням штрафу за чергу, конфігурація з підсиленням штрафу за затримку та варіант із

більшим штрафом за перемикання фаз. Найкращим залишився збалансований набір queue_weight=1.0, delay_weight=1.0, switch_penalty=0.1, який уже використовувався в базових Scale-запусках. Це важливий практичний результат: для заданого сценарію надмірне зміщення винагороди до однієї складової не покращило інтегральну транспортну оцінку. Якщо збільшити вагу черги, агент може надто агресивно обслуговувати накопичення на окремих підходах; якщо посилити вагу затримки, політика стає чутливішою до локальних часових втрат; якщо збільшити штраф за перемикання, система може рідше змінювати фази, але ризикує гірше реагувати на зміну попиту. Баланс цих складових виявився доцільним для подальшого порівняння моделей.

Фінальні етапи auto-tuning перевіряли стійкість обраних параметрів за масштабами попиту 0,25, 0,5, 0,75 і 1,0. Для DQN значення score зростало від 126,783 при Scale=0,25 до 354,374 при Scale=1,0; для PPO – від 125,085 до 354,146. За малих масштабів PPO мав трохи нижчий інтегральний критерій: при Scale=0,25 середня тривалість поїздки становила 96,670 с, час очікування 6,111 с, втрати часу 22,284 с і 50874 завершені поїздки. Для DQN у тому самому масштабі відповідні значення дорівнювали 97,220 с, 6,697 с, 22,847 с і 50865 завершених поїздок. При Scale=0,75 DQN показав помітно кращу стійкість: score = 211,348 проти 243,729 у PPO, а кількість завершених поїздок становила 149520 проти 141184. При Scale=1,0 інтегральні оцінки двох алгоритмів стали майже однаковими: 354,374 для DQN і 354,146 для PPO, однак структура метрик дещо відрізнялася. PPO мав нижчий середній час очікування 44,678 с проти 45,999 с у DQN, але DQN мав трохи менші середні втрати часу 119,028 с проти 119,513 с. Автоматичний підбір не тільки визначив кращі параметри, а й показав, що перевага алгоритму залежить від масштабу транспортного навантаження.

Таблиця 4.2 демонструє найкращі кандидати за етапами автоматичного підбору параметрів.

Таблиця 4.2 – Найкращі кандидати за етапами автоматичного підбору параметрів

Алгоритм	Основні зафіксовані параметри	Найкращий етапний результат
DQN	hidden_dim=256, activation=leaky_relu, learning_rate=0.001, epsilon_decay_steps=10000, queue_weight=1.0, delay_weight=1.0, switch_penalty=0.1	score = 182,776 на етапах learning rate, epsilon decay і reward weights
PPO	hidden_dim=256, activation=tanh, learning_rate=0.0003, rollout_steps=256, ppo_epochs=4	score = 207,033 на етапах architecture, learning rate і rollout/epochs

Отримані результати auto-tuning формують основу для фінального порівняння моделей у наступних експериментальних підрозділах. Після кожного етапу кращі параметри не залишалися ізольованим результатом, а автоматично фіксувалися в launcher-інтерфейсі та використовувалися на наступних етапах. Завдяки цьому підбір мав послідовний характер: спочатку визначалась архітектура, потім уточнювалися параметри навчання, після цього перевірялися ваги винагороди DQN і лише наприкінці виконувалася перевірка стійкості за різними масштабами попиту. Такий підхід є більш обґрунтованим, ніж ручний вибір конфігурації, оскільки кожне рішення спирається на однаковий критерій score, сформований із фактичних tripinfo-результатів SUMO. У підсумку експериментальна система отримує не просто навчені DQN і PPO, а дві параметризовані моделі, підготовлені до коректного порівняння в однаковому транспортному середовищі.

4.5 Збір результатів, аналіз метрик і порівняння ефективності DQN та PPO

Збір результатів у четвертому розділі завершує експериментальний контур, описаний у підрозділах 4.1–4.4: після запуску SUMO-сценарію,

навчання DQN або PPO, проміжного оцінювання та автоматичного підбору параметрів система має сформувати не лише checkpoint моделі, а й набір файлів, за якими можна перевірити транспортну якість керування. У межах цієї роботи основним джерелом фактичних даних є директорія Test_Sumo, де збережено окремі серії dqn_scale_*, ppo_scale_*, піддиректорію auto_tuning і графіки plots_ua. Така організація результатів має принципове значення: вона дає змогу відокремити навчальні сигнали агента від транспортних метрик завершених поїздок і не зводити оцінювання світлофорного регулювання лише до значення винагороди.

Для кожного запуску DQN у папці результатів створюється однакова структура файлів: dqn_config.json, dqn_train_log.csv, dqn_eval_log.csv, dqn_training_signals.csv, dqn_tripinfo_summary.csv, dqn_tripinfo_vehicles.csv, директорія dqn_checkpoints із файлами .pt і директорія dqn_tripinfo з XML-файлами SUMO tripinfo-output.

Для PPO використовується симетрична структура: ppo_config.json, ppo_train_log.csv, ppo_eval_log.csv, ppo_training_signals.csv, ppo_tripinfo_summary.csv, ppo_tripinfo_vehicles.csv, ppo_checkpoints і ppo_tripinfo. Файли dqn_config.json і ppo_config.json фіксують умови запуску: сценарій 2x2_sumo_grid, backend sumo, тривалість епізоду 3600 с, режим попиту generated, значення demand_scale, параметри навчання, частоту evaluation і checkpointing.

Завдяки цьому результати не є ізольованими числовими таблицями: кожен CSV-файл можна пов’язати з конкретною конфігурацією експерименту, а за потреби повторити запуск у тому самому режимі. Фрагменти конфігураційних параметрів і програмного інструментарію, що формує такі результати, подано в додатках Б і Д.

Важливо розрізняти два рівні даних, які зберігаються після навчання. Перший рівень утворюють покрокові навчальні сигнали середовища, записані у dqn_training_signals.csv і ppo_training_signals.csv. У цих файлах фіксуються локальні показники, які безпосередньо пов’язані з роботою

агентів на кожному кроці моделювання: mean_n_queued, mean_occupancy, mean_delay, mean_pressure, mean_reward, sum_n_queued, sum_pressure, а для DQN додатково epsilon і mean_switched.

Саме ці величини відображають, на які сигнали реагує нейромережева модель під час взаємодії із середовищем: довжина черги, затримка, транспортний тиск, зайнятість смуг, штраф за перемикання та локальна винагорода. Вони є корисними для аналізу стабільності навчання, але не можуть самі по собі вважатися остаточною оцінкою транспортної якості, оскільки формуються на кожному кроці та відображають поточний стан мережі, а не завершений маршрут транспортного засобу.

Другий рівень даних утворює SUMO tripinfo-output, який після обробки зберігається у файлах dqn_tripinfo_summary.csv, dqn_tripinfo_vehicles.csv, ppo_tripinfo_summary.csv і ppo_tripinfo_vehicles.csv. На відміну від покрокових сигналів, tripinfo-output містить фактичні характеристики тільки тих транспортних засобів, які завершили поїздку. Тому підсумкові файли tripinfo_summary.csv доцільно використовувати для оцінювання якості моделі після або під час evaluation, а не як заміну навчальній винагороді. Основними фактичними метриками є duration_mean, waitingTime_mean, timeLoss_mean і n_completed: середній час поїздки, середній час очікування, середні втрати часу та кількість завершених поїздок. Саме це розмежування є важливим для дипломної роботи, оскільки воно показує різницю між тим, на чому модель навчається, і тим, за чим оцінюється її практична ефективність у транспортній мережі [8], [16].

У прямих Scale-запусках директорії збереження результатів для DQN і PPO використано однаковий сценарій 2x2_sumo_grid, однакову тривалість епізоду 3600 с, generated-попит і чотири значення масштабу: 0,25, 0,5, 0,75 і 1,0. Підсумкові tripinfo_summary.csv у кожній папці сформовано на основі 38 XML-файлів tripinfo, що дає змогу порівнювати алгоритми за однаковою процедурою збору даних.

Для малих масштабів попиту різниця між DQN і PPO є незначною: при Scale=0,25 DQN має середній час поїздки 97,144 с, середній час очікування 6,605 с і середні втрати часу 22,763 с, тоді як PPO має відповідно 96,689 с, 6,130 с і 22,301 с. При Scale=0,5 ситуація також близька: DQN дає 104,344 с середнього часу поїздки та 9,717 с очікування, PPO – 104,189 с і 9,528 с. Це свідчить, що за помірного попиту обидва алгоритми знаходять прийнятну політику керування, а відмінності між ними не мають різкого характеру. Таблиця 4.3 демонструє підсумкові tripinfo-метрики DQN і PPO за масштабами попиту.

Таблиця 4.3 – Підсумкові tripinfo-метрики DQN і PPO за масштабами попиту

Алгоритм	Scale	n_completed	duration_mean, с	waitingTime mean, с	timeLoss_ mean, с
DQN	0,25	50869	97,144	6,605	22,763
PPO	0,25	50868	96,689	6,130	22,301
DQN	0,50	101569	104,344	9,717	29,194
PPO	0,50	101607	104,189	9,528	29,037
DQN	0,75	148615	138,112	24,460	63,371
PPO	0,75	148452	142,088	25,686	67,379
DQN	1,00	123273	189,568	45,160	119,186
PPO	1,00	171580	248,235	118,821	174,487

Зі збільшенням масштабу попиту відмінності стають помітнішими. При Scale=0,75 DQN показує нижчий середній час поїздки, менший час очікування і менші втрати часу, ніж PPO, хоча кількість завершених поїздок у двох алгоритмів залишається близькою: 148615 проти 148452. При Scale=1,0 картина стає складнішою. PPO завершує більше поїздок у збереженому наборі /ppo_scale_1 – 171580 проти 123273 у /dqn_scale_1, однак часові метрики PPO суттєво гірші: duration_mean дорівнює 248,235 с проти 189,568 с у DQN, waitingTime_mean – 118,821 с проти 45,160 с, а timeLoss_mean – 174,487 с проти 119,186 с. Такий результат не можна інтерпретувати одновимірно.

Більша кількість завершених поїздок свідчить про активніше проходження транспортних засобів через мережу, проте різке зростання часу очікування й втрат часу показує, що це проходження супроводжувалося значним накопиченням затримок. Для транспортного керування важливим є саме баланс між пропуском і часовими втратами, тому оцінювання потребує одночасного врахування всіх чотирьох метрик.

Покрокові навчальні сигнали підтверджують, що зростання Traffic Scale ускладнює задачу для обох алгоритмів. У файлі /dqn_scale_1/dqn_training_signals.csv на останньому записаному кроці global_step=19996 середня черга mean_n_queued дорівнює 163,25, сумарна черга sum_n_queued – 653, а mean_reward становить -163,850. Для PPO у файлі /ppo_scale_1/ppo_training_signals.csv на відповідному завершальному кроці global_step=19996 середня черга дорівнює 139,0, сумарна черга – 556, а mean_reward становить -389,221.

Ці значення не слід прямо порівнювати з duration_mean або waitingTime_mean, оскільки вони мають іншу природу: навчальний сигнал відображає стан мережі на кроці моделювання, а tripinfo – підсумок завершених маршрутів. Проте разом вони пояснюють, чому при високому навантаженні оцінювання лише за винагородою або лише за кількістю завершених поїздок може дати неповну картину.

Файли dqn_train_log.csv і ppo_train_log.csv використовуються для аналізу внутрішньої динаміки навчання. Для DQN у журналі зберігаються global_step, epsilon, loss, mean_episode_reward, mean_episode_length, mean_average_travel_time, mean_n_vehicles і buffer_size; для PPO – update, global_step, policy_loss, value_loss, entropy, total_loss, mean_episode_reward, mean_episode_length, mean_average_travel_time і mean_n_vehicles. Відмінність цих журналів відповідає різній природі алгоритмів: DQN контролює спадання epsilon і помилку апроксимації Q-значень, тоді як PPO зберігає складові actor-critic-оновлення: policy_loss, value_loss, entropy і total_loss [13], [14].

У прямих запусках DQN завершував навчання з epsilon=0,05, що означає перехід від активного дослідження до майже жадібної політики. У PPO останні значення value_loss істотно зростають зі збільшенням масштабу попиту, що вказує на складність апроксимації функції цінності в умовах більш насиченої транспортної мережі. Це не є самостійною транспортною метрикою, але пояснює, чому для PPO особливо важливі параметри rollout_steps, ppo_epochs, gae_lambda і стабільність policy update.

Візуальний аналіз результатів у системі винесено в меню Build plots, реалізоване в run_once_gui.py і пов’язане з модулем /training/plots.py. Це меню формує графіки українською мовою в директорії Output folder/plots_ua/.

Програмно підтримуються графіки середньої винагороди за крок, окремі графіки DQN/PPO loss, normalized loss comparison, фактичного середнього часу поїздки, середнього часу очікування, середніх втрат часу, кількості завершених поїздок, середньої довжини черги з rolling average, порівняння за Traffic Scale та графіки етапів Auto tune. Перелік сформованих графіків для auto-tuning зафіксовано в додатку Е. Рисунок 4.2 демонструє порівняння DQN і PPO за фактичними tripinfo-метриками.

Графіки auto-tuning доповнюють числові таблиці, оскільки показують не лише фінальний результат, а й логіку вибору параметрів. Зокрема, файл auto_tune_98_final_best_parameters.png узагальнює найкращі параметри, зафіксовані в Test_Sumo/auto_tuning/auto_tuning_summary.json: для DQN це hidden_dim=256, activation=leaky_relu, learning_rate=0.001, epsilon_decay_steps=10000, queue_weight=1.0, delay_weight=1.0, switch_penalty=0.1; для PPO – hidden_dim=256, activation=tanh, learning_rate=0.0003, rollout_steps=256, ppo_epochs=4.

У підрозділі 4.4 уже було показано, що підбір виконувався за інтегральним критерієм score, сформованим із duration_mean, waitingTime_mean, timeLoss_mean і n_completed.

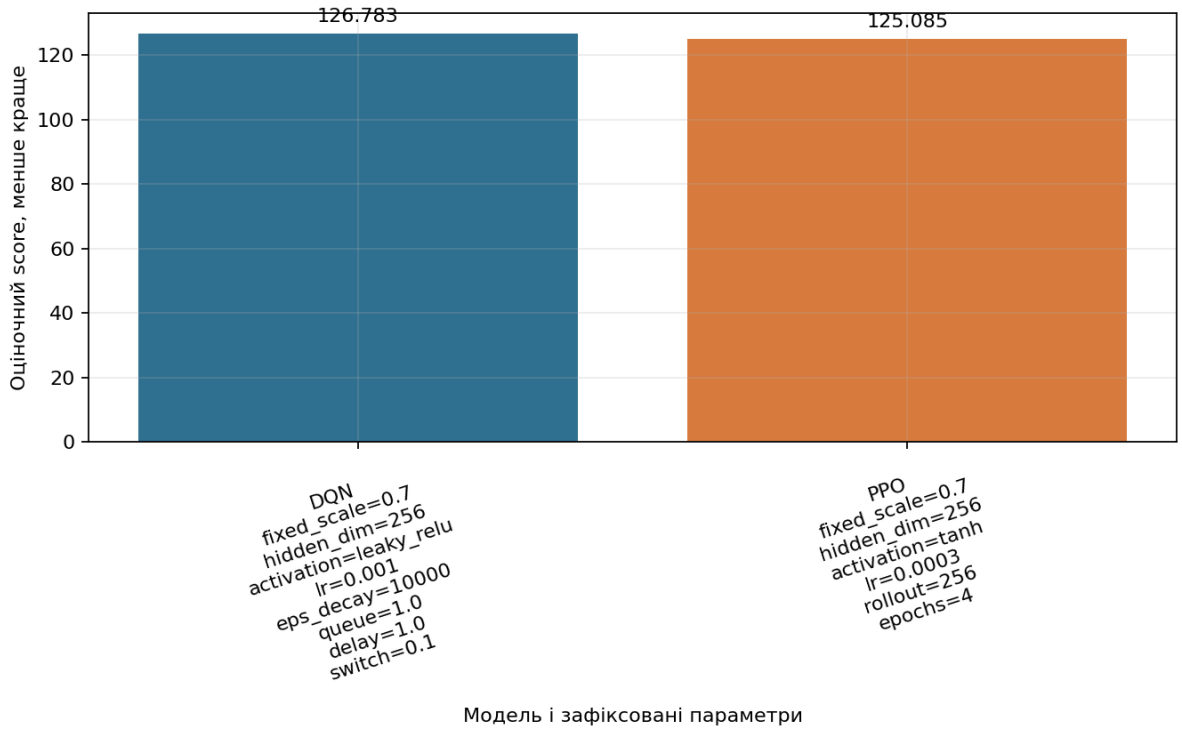


Рисунок 4.2 – Порівняння DQN і PPO за фактичними tripinfo-метриками

У цьому підрозділі важливо підкреслити інше: графіки потрібні не для декоративного подання результатів, а для виявлення режимів, у яких одна модель поводиться стабільніше за іншу. Наприклад, для Scale=0,75 у фінальних robustness-тестах auto-tuning DQN мав нижчий score і більшу кількість завершених поїздок, тоді як при Scale=1,0 інтегральні оцінки DQN і PPO стали майже однаковими.

Зведені результати автоматичного підбору параметрів, на основі яких побудовано ці графіки, подано в додатку Е. Вибір алгоритму залежить не тільки від середнього результату, а й від рівня транспортного навантаження.

Зведення найкращих параметрів DQN і PPO за даними:

– dan_activation: leaky_relu;
– dan_delay_weight: 1.0;
– dan_epsilon_decay_steps: 10000;
– dan_ hidden_dim: 256;
– dqn_learning_rate: 0.001;

– dan_queue_weight: 1.0;
– dan_switch_penalty: 0.1;
– dan_total_env_steps: 20000;
– ppo_activation: tanh;
– ppo_hidden_dim: 256;
– ppo_learning_rate: 0.0003;
– ppo_ppo_epochs: 4;
– ppo_rollout_steps: 256;
– ppo_target_global_steps: 20000;
– traffic_scale: 0.7.

4.6 Перевірка стійкості найкращих моделей DQN і PPO до зміни інтенсивності транспортного потоку

Після програмної реалізації, навчання моделей і поетапного підбору параметрів було виконано завершальну evaluation-only перевірку стійкості найкращих моделей DQN і PPO до зміни інтенсивності транспортного потоку.

На відміну від попередніх експериментів, на цьому етапі нове навчання не виконувалося. Система використовувала результати режиму Auto tune stages, завантажувала найкращі збережені ваги DQN і PPO та запускала їх у режимі оцінювання для кількох значень параметра Traffic Scale. Така перевірка потрібна для відповіді на прикладне питання: чи зберігає навчена модель перевагу не тільки в одному сценарії попиту, а й за інших рівнів транспортного навантаження.

Фінальну перевірку виконано в директорії /Users/kolii_oleksandr/Desktop/Test_Sumo/auto_tuning. Для DQN результати збережено в папці 05_final_dqn_scale_robustness, для PPO - у 05_final_ppo_scale_robustness, а для контрольного режиму фіксованого світлофорного циклу - у 05_final_fixed_time_scale_baseline.

У всіх трьох випадках використовувався сценарій 2x2_sumo_grid, generated-попит і тривалість моделювання 3600 с. Масштаб транспортного потоку змінювався за значеннями 0,25; 0,5; 0,75 і 1,0.

Для DQN у конфігураційних файлах зафіксовано hidden_dim=256, activation=leaky_relu, learning_rate=0.001, epsilon_decay_steps=10000; для PPO - hidden_dim=256, activation=tanh, learning_rate=0.0003, rollout_steps=256, ppo_epochs=4. Фрагменти конфігурацій DQN, PPO і fixed_time, які підтверджують однакові умови evaluation-only запусків, подано в додатку Б.

Отже, у цьому підрозділі порівнюються не довільні запуски, а найкращі конфігурації, отримані після автоматичного підбору параметрів.

Ключовою відмінністю підрозділу 4.6 від попереднього аналізу є обов’язкове включення baseline-контролера fixed_time. У цьому режимі світлофори працюють за фіксованим циклом без нейромережі, без навчання і без адаптації до стану транспортної мережі.

Тому fixed_time використовується як контрольна межа практичної доцільності: якщо DQN або PPO не покращують час поїздки, очікування, втрати часу або кількість завершених поїздок порівняно з цим режимом, то нейромережеве керування для відповідного рівня навантаження не має достатньої прикладної переваги.

Інтегральний показник score у системі формується за транспортними метриками SUMO: середнім фактичним часом поїздки, середнім часом очікування, середніми втратами часу та штрафним доданком, пов’язаним із кількістю завершених поїздок. Менше значення score, duration_mean, waitingTime_mean і timeLoss_mean відповідає кращому результату.

Таблиця 4.4 представляє зведені фрагменти узагальнених результатів, які взяті з файлу auto_tuning_summary.json і відповідних tripinfo_summary.csv.

Таблиця 4.4 – Перевірка стійкості найкращих DQN і PPO порівняно з fixed_time

Scale	Метод	score	duration_mean, с	waitingTime_mean, с	timeLoss_mean, с	n_completed
0,25	DQN	160,046	108,367	17,462	33,973	4116
0,25	PPO	124,209	96,303	5,757	21,908	4131
0,25	fixed_time	152,940	105,712	15,188	31,313	1375
0,50	DQN	157,149	108,814	14,570	33,644	8238
0,50	PPO	141,062	103,560	8,972	28,409	8253
0,50	fixed_time	167,292	112,047	18,001	36,880	2745
0,75	DQN	191,841	123,707	19,386	48,666	12210
0,75	PPO	236,846	142,972	25,521	68,271	12048
0,75	fixed_time	210,368	128,450	28,358	53,315	4079
1,00	DQN	359,588	192,820	44,252	122,417	9990
1,00	PPO	349,553	188,562	42,734	118,156	10011
1,00	fixed_time	537,848	250,248	111,322	176,069	4781

За низького навантаження, тобто при Scale=0,25, найкращий результат показала модель PPO. Вона зменшила середній час поїздки порівняно з fixed_time з 105,712 с до 96,303 с, середній час очікування – з 15,188 с до 5,757 с, а середні втрати часу – з 31,313 с до 21,908 с. Це відповідає зниженню часу очікування приблизно на 62,1% і втрат часу приблизно на 30,0%. DQN у цьому режимі, навпаки, мав гірші часові метрики за fixed_time: середній час поїздки зріс до 108,367 с, очікування – до 17,462 с, втрати часу – до 33,973 с. При малому попиті PPO поводиться стійкіше, тоді як DQN не дає переваги над простим фіксованим циклом.

При Scale=0,5 обидві нейромережеві моделі вже перевищують baseline за основними часовими показниками. DQN зменшує score на 6,1% відносно fixed_time, а PPO – на 15,7%. Для PPO середній час очікування становить 8,972 с проти 18,001 с у fixed_time, тобто зменшується приблизно на 50,2%. Середні втрати часу також нижчі: 28,409 с у PPO проти 36,880 с у baseline. Це свідчить, що за помірного транспортного потоку навчена політика PPO краще адаптує фази світлофорів до фактичного стану черг, ніж фіксований цикл.

За Scale=0,75 ситуація змінюється. Найкращим стає DQN: його score дорівнює 191,841 проти 210,368 у fixed_time і 236,846 у PPO. DQN також має нижчий середній час поїздки, час очікування і втрати часу порівняно з baseline. Зокрема, середній час очікування зменшується з 28,358 с у fixed_time до 19,386 с у DQN, тобто приблизно на 31,6%. PPO у цьому режимі зберігає менший час очікування, ніж fixed_time, але програє за середнім часом поїздки, втратами часу та інтегральним score. Це показує, що PPO виявився чутливішим до переходу від помірного до підвищеного навантаження, тоді як DQN краще витримав цей рівень попиту.

Найважчим режимом є Scale=1,0. Тут усі методи демонструють зростання затримок, що очікувано для насиченого транспортного потоку. Проте обидві нейромережеві моделі суттєво переважають fixed_time за часовими метриками. Для DQN середній час поїздки становить 192,820 с, для PPO – 188,562 с, тоді як у fixed_time він зростає до 250,248 с. Середній час очікування зменшується з 111,322 с у baseline до 44,252 с у DQN і 42,734 с у PPO. За інтегральним показником score PPO має невелику перевагу над DQN: 349,553 проти 359,588. При максимальному з перевірених навантажень обидві навчені моделі залишаються практично корисними, а PPO показує трохи кращий баланс часових метрик.

Окремо слід врахувати кількість завершених поїздок. У tripinfo_summary.csv для DQN і PPO результати сформовано за 5 XML-файлами, а для fixed_time – за 2 XML-файлами, тому абсолютні значення n_completed не можна механічно порівнювати без урахування кількості файлів. Якщо нормувати кількість завершених поїздок на один файл, то при Scale 0,25-0,75 DQN і PPO дають приблизно на (18–20)% більше завершених поїздок за один запуск, ніж fixed_time. При Scale=1,0 ситуація інша: fixed_time має більше завершених поїздок на файл, але ці поїздки супроводжуються значно більшим очікуванням і втратами часу. Тому для насиченого режиму важливо не трактувати пропуск лише за n_completed, а розглядати його разом із duration_mean, waitingTime_mean і timeLoss_mean.

Для візуального аналізу в меню Build plots використовується прапорець Compare best weights vs fixed-time baseline. Після його увімкнення система читає Output folder/auto_tuning/auto_tuning_summary.json і будує графіки в папці Output folder/plots_ua/. Фрагмент програмного інструментарію побудови цих графіків подано в додатку Д. На кожному графіку по осі X відкладається значення Traffic Scale, а окремими лініями показано DQN з найкращими вагами, PPO з найкращими вагами та fixed_time baseline. Перелік відповідних графічних файлів і підсумкових числових результатів наведено в додатку Е.

4.7 Висновки до розділу

Реалізована система збору результатів забезпечує повний цикл експериментального аналізу: збереження конфігурацій, журналів навчання, evaluation-метрик, агрегованих CSV-таблиць і графіків. Модель навчається за локальними кроковими показниками черг, затримок, тиску, зайнятості та винагороди, а її транспортна якість оцінюється за duration_mean, waitingTime_mean, timeLoss_mean і n_completed, отриманими з tripinfo_summary.csv. Це забезпечує коректність висновків: ефективність мультиагентної системи світлофорного регулювання визначається не лише внутрішньою динамікою DQN або PPO, а й фактичними наслідками їхніх рішень для завершених поїздок у SUMO-моделі міського транспортного потоку. Загалом фінальна перевірка підтвердила, що нейромережеве керування має практичну перевагу над фіксованим світлофорним циклом не в усіх режимах однаково. PPO є найкращим при низькому та помірному навантаженні, особливо за показником часу очікування. DQN виявився стійкішим при Scale=0,75, де PPO погіршив середній час поїздки та втрати часу порівняно з fixed_time. При Scale=1,0 обидві моделі суттєво зменшують часові втрати порівняно з baseline, а PPO має найменший інтегральний score.

ВИСНОВКИ

Отримані результати підтверджують, що мультиагентне світлофорне керування на основі DQN і PPO є придатним для кількісної оптимізації транспортних потоків у SUMO-сценарії 2x2_sumo_grid з тривалістю епізоду 3600 с. Експериментальна оцінка охопила чотири рівні транспортного попиту: 0,25, 0,5, 0,75 і 1,0, а якість керування визначалася за фактичними `tripinfo`-метриками завершених поїздок: середнім часом поїздки, середнім часом очікування, середніми втратами часу та кількістю завершених маршрутів.

За малого навантаження PPO мав незначну перевагу над DQN. При Scale=0,25 середній час поїздки становив 96,689 с для PPO проти 97,144 с для DQN, середній час очікування – 6,130 с проти 6,605 с, а середні втрати часу – 22,301 с проти 22,763 с. При Scale=0,5 різниця залишалася мінімальною: PPO забезпечив 104,189 с середнього часу поїздки та 9,528 с очікування, тоді як DQN – 104,344 с і 9,717 с відповідно. Це свідчить, що за помірного попиту обидва алгоритми формують близькі за якістю режими керування.

За вищого навантаження перевага змістилася в бік DQN. При Scale=0,75 DQN забезпечив 138,112 с середнього часу поїздки проти 142,088 с у PPO, 24,460 с очікування проти 25,686 с і 63,371 с втрат часу проти 67,379 с. Отже, у цьому режимі DQN зменшив середній час поїздки приблизно на 2,8%, час очікування – на 4,8%, а втрати часу – майже на 6,0% порівняно з PPO.

Найбільш показовим став режим Scale=1,0. DQN мав 189,568 с середнього часу поїздки, 45,160 с очікування і 119,186 с втрат часу, тоді як PPO показав відповідно 248,235 с, 118,821 с і 174,487 с. За цими часовими метриками DQN виявився кращим: середній час поїздки був нижчим приблизно на 23,6%, час очікування – на 62,0%, а втрати часу – на 31,7%. Водночас PPO завершив більше поїздок: 171580 проти 123273 у DQN, тобто

на 48307 маршрутів більше. Це показує, що в умовах повного навантаження кількість пропущених транспортних засобів не може бути єдиним критерієм якості, оскільки вона супроводжувалася істотним зростанням очікування та часових втрат.

Результати вказують на різну практичну придатність алгоритмів залежно від інтенсивності руху. PPO є конкурентним за малого й середнього попиту, де різниця з DQN не перевищує кількох десятих секунди за середнім часом поїздки. DQN виявляється стійкішим за підвищеного навантаження, особливо коли головним критерієм є мінімізація очікування та втрат часу. Найкраща конфігурація DQN використовує hidden_dim=256, leaky_relu, learning_rate=0,001 та epsilon_decay_steps=10000; для PPO найкращою стала конфігурація hidden_dim=256, tanh, learning_rate=0,0003, rollout_steps=256 і ppo_epochs=4.

Найкращі DQN і PPO моделі не слід оцінювати лише за результатом на одному рівні транспортного попиту. Проведена evaluation-only перевірка показала, що стійкість моделі залежить від інтенсивності потоку: PPO краще працює на малому та середньому навантаженні, DQN є надійнішим у проміжному насиченому режимі, а при максимальному навантаженні обидва нейромережеві підходи мають істотну перевагу над fixed_time за середнім часом поїздки, очікуванням і втратами часу. Це підтверджує прикладну доцільність використання мультиагентного навчання з підкріпленням для світлофорного регулювання, але також показує необхідність перевіряти навчені політики на кількох рівнях транспортного попиту перед практичним використанням.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Signal timing manual - second edition / T. Urbanik et al. Washington, D.C. : Transportation Research Board, 2015. URL: https://doi.org/10.17226/22097 (date of access: 24.05.2026).

2. Webster F. V. Traffic signal settings. London : H.M.S.O, 1958. 44 p.

3. Highway capacity manual 7th edition. Washington, D.C. : National Academies Press, 2022. 1285 p. URL: https://doi.org/10.17226/26432 (date of access: 24.05.2026).

4. Balancing safety and capacity in an adaptive signal control system – phase 1 - FHWA-HRT-10-038. *Access Denied*. URL: https://www.fhwa.dot.gov/publications/research/safety/10038/index.cfm (date of access: 23.05.2026).

5. Fuzzy logic in traffic engineering: a review on signal control / M. Koukol et al. *Mathematical problems in engineering*. 2015. Vol. 2015. P. 1–14. URL: https://doi.org/10.1155/2015/979160 (date of access: 24.05.2026).

6. Jiang W., Luo J. Graph neural network for traffic forecasting: a survey. *Expert systems with applications*. 2022. P. 117921. URL: https://doi.org/10.1016/j.eswa.2022.117921 (date of access: 24.05.2026).

7. Haydari A., Yilmaz Y. Deep reinforcement learning for intelligent transportation systems: a survey. *IEEE transactions on intelligent transportation systems*. 2020. P. 1–22. URL: https://doi.org/10.1109/tits.2020.3008612 (date of access: 24.05.2026).

8. Microscopic traffic simulation using SUMO / P. A. Lopez et al. *2018 21st international conference on intelligent transportation systems (ITSC)*, Maui, HI, 4–7 November 2018. 2018. URL: https://doi.org/10.1109/itsc.2018.8569938 (date of access: 24.05.2026).

9. PTV Vissim. *Empowering mobility for a cleaner, smarter & safer future | PTV Group*. URL: https://www.ptvgroup.com/en/products/ptv-vissim (date of access: 23.05.2026).

10. Axhausen K. W. The multi-agent transport simulation MATSim / ed. by A. Horni, K. Nagel. Ubiquity Press, 2016. URL: https://doi.org/10.5334/baw (date of access: 23.05.2026).

11. Sutton R. S., Barto A. G. Reinforcement learning: an introduction. Second edition. Cambridge, MA : The MIT Press, 2018. 552 p.

12. Watkins C. J. C. H., Dayan P. Q-learning. *Machine learning*. 1992. Vol. 8, no. 3-4. P. 279–292. URL: https://doi.org/10.1007/bf00992698 (date of access: 24.05.2026).

13. Human-level control through deep reinforcement learning / V. Mnih et al. *Nature*. 2015. Vol. 518, no. 7540. P. 529–533. URL: https://doi.org/10.1038/nature14236 (date of access: 23.05.2026).

14. Proximal policy optimization algorithms / J. Schulman et al. *arXiv* preprint arXiv:1707.06347. 2017. URL: https://arxiv.org/abs/1707.06347 (date of access: 23.05.2026).

15. Busoniu L., Babuska R., De Schutter B. A comprehensive survey of multiagent reinforcement learning. *IEEE transactions on systems, man, and cybernetics, part C (applications and reviews)*. 2008. Vol. 38, no. 2. P. 156–172. URL: https://doi.org/10.1109/tsmcc.2007.913919 (date of access: 24.05.2026).

16. Multi-agent deep reinforcement learning for large-scale traffic signal control / T. Chu et al. *IEEE transactions on intelligent transportation systems*. 2020. Vol. 21, no. 3. P. 1086–1095. URL: https://doi.org/10.1109/tits.2019.2901791 (date of access: 24.05.2026).

17. Multi-agent actor-critic for mixed cooperative-competitive environments / R. Lowe et al. Advances in neural information processing systems. 2017. Vol. 30. P. 6380–6391. URL: https://papers.nips.cc/paper/7217-multi-agent-actor-critic-for-mixed-cooperative-competitive-environments (date of access: 24.05.2026).

18. Інтелектуальні системи автоматизації : монографія / О. Г. Аврунін та ін. Кременчук : Видавництво «НОВАБУК», 2021. 322 с. URL: https://openarchive.nure.ua/handle/document/20504 (дата звернення: 21.05.2026).