

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

Вычислительная техника

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Профиль «Вычислительные машины, комплексы, системы
и сети»

БАКАЛАВРСКАЯ РАБОТА

на тему

**Бизнес-проект «ПРОГРАММА МОДЕЛИРОВАНИЯ
ИНТЕЛЛЕКТУАЛЬНОГО УПРАВЛЕНИЯ
ДОРОЖНЫМ ТРАФИКОМ.
ИНТЕЛЛЕКТУАЛЬНЫЕ АЛГОРИТМЫ»**

Студент

30.05.2025 Аляев А.О.
(подпись, дата)

Руководитель

30.05.2025 Митрохин М.А.
(подпись, дата)

Нормоконтролёр

30.05.2025 Бычков А.С.
(подпись, дата)

Работа допущена к защите (протокол заседания кафедры от

28.05.2025 г. № 16)

Заведующий кафедрой

отметка (подпись) Митрохин М.А.

Работа защищена с отметкой (протокол заседания ГЭК от 02.06.25 № 1)

Секретарь ГЭК

Г (подпись) Гурин Е.И.

Пенза, 2025

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

«Утверждаю»

Заведующий кафедрой

М.А. Митрохин

«06» 05 2025 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Аляев Александр Олегович гр. 21BBB1 факультета вычислительной техники.
2. Тема работы: Бизнес-проект «Программа моделирования интеллектуального управления дорожным трафиком. Интеллектуальные алгоритмы»

Тема утверждена приказом ПГУ № 601/о от "06" 05 2025 г.

3. Руководитель бакалаврской работы д.т.н., профессор, зав. каф. ВТ ПГУ Митрохин М.А.
4. Задание на работу (назначение разработки, исходные данные и т.п.)

Необходимо создать программу моделирования интеллектуального управления дорожным трафиком, которая предназначена для анализа и оптимизации работы светофорных объектов. Программа должна реализовывать интеллектуальные алгоритмы управления светофорами, такие как Q-обучение, нейронная сеть.

Программа должна получать входные данные о дорожном трафике либо генерировать их на основании настроек пользователя, а также отображать результаты моделирования при помощи графиков зависимостей характеристик дорожного движения.

5. Перечень подлежащих разработке вопросов

Требуется произвести анализ следующих предметных областей: изучить существующие методы и алгоритмы управления светофорными объектами, изучить современные интеллектуальные алгоритмы, применимые в управлении дорожным движением, изучить существующие среды имитационного моделирования транспортных потоков и выбрать наиболее подходящую из них. Также необходимо изучить функционал фреймворков PyTorch и Gym. Необходимо рассмотреть вопросы создания архитектуры программы, реализации интеллектуальных алгоритмов в выбранном языке программирования, а также изучить возможности интеграции алгоритмов со средой имитационного моделирования.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Изучение предметной области	52	06.05.2025-09.05.2025	
2.	Анализ требований	13	10.05.2025	
3.	Выбор технологий, применяемых при проектировании и разработке	52	11.05.2025-14.05.2025	
4.	Описание алгоритмов	39	15.05.2025-17.05.2025	
5.	Разработка программы моделирования интеллектуального управления дорожным трафиком	104	18.05.2025 - 25.05.2025	
6.	Оформление пояснительной записки	64	26.05.2025 - 30.05.2025	

Дата выдачи задания " 06 " 05 2025 г.

Руководитель бакалаврской работы  Митрохин М.А.
(подпись, дата)

Задание к исполнению принял студент  06.05.2025 Аляев А.О.
(подпись, дата)

Работу к защите допустить
Декан факультета  27.05.2025 Фионова Л.Р.
(подпись, дата)

Реферат

Бакалаврская работа содержит 107 листов, 14 рисунков, 5 таблиц, 32 источника и 3 приложения.

ИММИТАЦИОННОЕ МОДЕЛИРОВАНИЕ ТРАНСПОРТНЫХ ПОТОКОВ, ТРАФИК, ИНТЕЛЛЕКТУАЛЬНЫЕ АЛГОРИТМЫ РАБОТЫ СВЕТОФОРОВ, АГЕНТНЫЙ ПОДХОД, МАШИННОЕ ОБУЧЕНИЕ

Цель работы заключается в разработке программы, которая реализует интеллектуальные методы управления светофорными объектами на перекрестке, а также визуализирует их эффективность при помощи графиков зависимостей различных характеристик дорожного движения.

Объектом исследования в работе являются интеллектуальные алгоритмы расчёта планов работы светофорного объекта на перекрестке.

В работе применены методы имитационного моделирования дорожного транспорта в среде SUMO, методы машинного обучения, агентный подход, а также программная реализация с использованием библиотек и фреймворков Python.

В работе рассмотрены существующие системы моделирования транспортных потоков, проведён обзор современных методов интеллектуального управления светофорами, протестированы интеллектуальные алгоритмы для расчета планов работы светофорного объекта, а также определен наилучший алгоритм для применения в системе управления светофорами на перекрестке.

					ПГУ 1.090301.02.001 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	Программа моделирования интеллектуального управления дорожным трафиком. Интеллектуальные алгоритмы Пояснительная записка	Лит.	Лист	Листов
Разраб.		Аляев А.О.		30.05.25				
Провер.		Митрохин М.А.		30.05.25			2	107
Ил. Контр		Бычков А.С.		30.05.25		ФВТ, каф. ВТ гпр. 21ВВВ1		
Утверд.		Митрохин М.А.		30.05.25				

Содержание

Перечень сокращений и обозначений	5
Введение.....	6
1 Программа моделирования интеллектуального управления	
дорожным движением	7
1.1 Постановка задачи.....	7
1.2 Анализ предметной области	8
1.2.1 Обзор аналогов	9
1.2.2 Выбор инструментов разработки	11
1.2.3 Выбор среды разработки.....	14
1.2.4 Выбор архитектуры программы	14
1.3 Проектирование и разработка.....	17
1.3.1 Механика моделирования в среде SUMO	17
1.3.2 Механизмы управления светофорными объектами в SUMO	18
1.3.3 Формат данных и интеграция с реальными источниками	20
1.3.4 Агентный подход в управлении светофорными объектами	20
1.3.5 Алгоритмы управления светофорными объектами:	
статические, адаптивные, интеллектуальные	22
1.3.6 Разработка модуля управления и настройки агента	30
1.3.7 Разработка алгоритма Q-обучения	32
1.3.8 Разработка нейронной сети.....	34
1.4 Тестирование и анализ результатов	36
1.4.1 Критерии оценки эффективности алгоритма.....	36
1.4.2 Анализ эффективности Q-обучения	37
1.4.3 Анализ эффективности нейронной сети	41
2 Охрана труда.....	48
2.1 Характеристика потенциальных рисков при работе с ПК	48
2.2 Оборудование рабочего места	48

					ПГУ 1.090301.02.001 ПЗ		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Аляев А.О.			Программа моделирования интеллектуального управления дорожным трафиком. Интеллектуальные алгоритмы Пояснительная записка	Лит.	Лист
Провер.		Митрохин М.А.					
И. Контр		Бычков А.С.				3	107
						ФВТ, каф. ВТ <i>гр. 21ВВВ1</i>	
Утверд.		Митрохин М.А.					

2.3 Освещённость рабочего места.....	49
2.4 Электробезопасность	50
2.5 Пожарная безопасность	50
2.6 Санитарно-гигиеническая характеристика производства.....	50
3 Стартап проект	52
3.1 Анализ рынка	52
3.2 Целевая аудитория и каналы взаимодействия.....	52
3.3 Источники дохода и расходов	53
3.4 Риски проекта.....	54
3.5 Ключевые ресурсы	55
Заключение.....	56
Список использованных источников	57
Приложение А - UML – диаграммы программы	61
Приложение Б - Листинг программы.....	67
Приложение В - Презентация.....	100

Перечень сокращений и обозначений

ИТС – интеллектуальная транспортная система

ПЭВМ – персональная электронно-вычислительная машина

ТС – транспортное средство

API (Application Programming Interface) – программный интерфейс приложения

ReLU (Rectified Linear Unit) – выпрямлённый линейный элемент

SUMO (Simulation of Urban Mobility) – моделирование городской мобильности

TraCI (Traffic Control Interface) – интерфейс управления дорожным движением

XML (eXtensible Markup Language) – расширяемый язык разметки

					ПГУ 1.090301.02.001 ПЗ	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		

Введение

Светофоры – важный элемент городского управления дорожным движением. Благодаря их использованию на дорогах организуется безопасность движения. Но со временем количество автомобилей растёт, что приводит к всё большим заторам. Из-за этого существующие статичные алгоритмы управления светофорными объектами теряют в своей эффективности, что ведёт к переполнению дорог и долгому ожиданию проезда перекрёстка. Решением этой проблемы может стать использование интеллектуальных алгоритмов управления светофорами, основанными на машинном обучении и нейронных сетях.

Эти алгоритмы представляют собой механизмы, которые могут адаптироваться к текущим условиям дорожного движения. Они позволяют светофорам реагировать на изменение дорожной ситуации и применять соответствующие меры для оптимизации движения. В результате их применения может достигаться более эффективное управление дорожным движением, что будет способствовать снижению заторов и других отрицательных эффектов.

Анализ эффективности интеллектуальных алгоритмов имеет важное значение, так как их использование может способствовать улучшению эффективности дорожного движения в целом, снижению вредных выбросов в атмосферу и др. Вместе с этим, можно предположить, что результаты управления дорожным трафиком в среде моделирования могут сильно отличаться от результатов реального использования в городской среде.

1 Программа моделирования интеллектуального управления дорожным движением

1.1 Постановка задачи

Задача состоит в разработке программы для исследования интеллектуальных алгоритмов управления фазами работы светофорных объектов и графического пользовательского интерфейса для неё.

Программа должна отвечать следующим требованиям:

- программа должна поддерживать использование различных алгоритмов машинного обучения: Q-обучения и нейронных сетей;
- блок анализа данных о моделировании должен автоматически фиксировать метрики характеристик дорожного движения: время ожидания, длина очереди, количество машин в модели;
- в процессе моделирования интеллектуальные алгоритмы должны самостоятельно обучаться;
- программа должна работать в ОС Microsoft Windows 10 и выше.

Ожидаемые результаты:

- программа, позволяющая исследовать интеллектуальные алгоритмы в среде имитационного моделирования;
- доказательство эффективности Q-обучения и нейронных сетей по сравнению с классическими методами управления светофорными объектами.

Технические требования:

- процессор: не менее 2 ядер;
- оперативная память: не менее 8 ГБ;
- дисковое пространство: не менее 8 ГБ;
- видеокарта: интегрированная (если не требуется использование тензорных ядер).

1.2 Анализ предметной области

Исследование интеллектуальных алгоритмов управления светофорными

					ПГУ 1.090301.02.001 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		

объектами неэффективно без применения современных инструментов анализа и моделирования дорожного движения. Одним из наиболее перспективных направлений является имитационное моделирование, которое позволяет определять эффективность исследуемых алгоритмов, до их апробации на реальных перекрестках.

Имитационное моделирование широко применяется для определения наилучшей схемы управления транспортными потоками в статье [1].

Отечественные [2] и зарубежные [3] исследования показали, что имитационное моделирование имеет потенциал в анализе работы транспортных систем.

Исследования [4] продемонстрировали эффективность различных алгоритмов, используемых при имитационном моделировании. Как упоминалось в статье [5], имитационные модели транспортных потоков являются перспективным инструментом для анализа и оценки систем управления городским дорожным движением.

Т. А. Кузнецов [6] в своей работе по проблемам моделирования дорожного движения рассмотрел задачу регулирования продолжительности зеленого света светофоров в зоне перекрестка. Авторы использовали методы имитационного моделирования для выбора оптимальных параметров работы светофоров, что позволило сократить время ожидания транспортных средств и уменьшить заторы на перекрестках.

В рамках изучения современных инструментов имитационного моделирования транспортных потоков авторы Захарова Ю. И. и Карнауха Е. С. [7] подтвердили значительное упрощение процесса изучения и контроля динамики транспортного потока средствами имитационного моделирования.

Наконец, в работах [8...10] доказана эффективность применения интеллектуальных алгоритмов управления светофорными объектами. Их использование приводит к снижению временных задержек и транспортных заторов, а также к общему улучшению характеристик транспортного потока.

Особенно стоит отметить программное обеспечение [11], которое функционирует в нашем городе.

1.2.1 Обзор аналогов

В России существует несколько программных продуктов, целью которых является использование систем интеллектуального управления светофорными объектами.

Программное обеспечение AVEDEX [12], разработанное компанией MALLENUM TECHNOLOGIES — это система анализа транспортных и пешеходных потоков, использующая технологии машинного зрения и нейронные сети. К преимуществам данного ПО относятся: точная видеоаналитика, основанная на применении нейронных сетей; настройка учёта типов участников дорожного движения; работа по расписанию; интеграция с другими системами. К недостаткам можно отнести: невозможность на самообучения модели в процессе работы; централизованное управление данными с перекрёстков.

Научно-производственное объединение НПО «РБС», разрабатывает систему управления дорожным движением Авангард [13], которая представляет собой решение, направленное на мониторинг, анализ и оптимизацию транспортных потоков. К достоинствам можно отнести: сбор и анализ данных с разных источников (камеры, индукционные датчики); анализ аварийности на дорогах; создание виртуальных моделей дорожной сети в реальном времени.

Недостатки: обязательное наличие нескольких типов датчиков для работы системы; не используются методы машинного обучения и нейронные сети.

Компания Complitech разрабатывает систему Sentinel Smart Traffic [14], представляющую собой программное обеспечение для автоматического сбора и анализа данных о транспортных потоках. Достоинствами системы является: автоматический мониторинг нарушений ПДД; оперативная реакция на ДТП; система распознавания номерных знаков автомобилей. К недостатку можно отнести: высокие финансовые затраты на интеграцию системы.

Компания СпецДорПроект занимается разработкой интеллектуальной транспортной системы “Умный перекрёсток” [15] в Москве и других городах.

					ПГУ 1.090301.02.001 ПЗ	Лист
						9
Изм.	Лист	№ докум.	Подпись	Дата		

Она использует индуктивные детекторы для адаптивного управления светофорами, что сокращает время ожидания и повышает пропускную способность дорог. Достоинствами данной системы являются: осуществление приоритета движения для общественного транспорта, интеграция с общей ИТС города. Главным недостатком является высокая цена для использования системы.

В таблице 1 представлена сводная информация о достоинствах и недостатках каждого программного продукта.

Таблица 1 – Сводная таблица достоинств и недостатков аналогов

Программное обеспечение	Разработчик	Достоинство	Недостаток
AVEDEX	MALLENOM TECHNOLOGIES	Высокоточная видеоаналитика на основе нейросетей; Учёт нескольких типов участников движения; Работа по расписанию; Интеграция с другими системами.	Невозможность самообучения модели; Централизованное управление данными с перекрёстков.
Авангард	НПО «РБС»	Сбор данных с разных источников (камеры, датчики); Анализ аварийности; Виртуальные модели дорожной сети в реальном времени.	Требует несколько типов датчиков; Отсутствие машинного обучения).
Sentinel Smart Traffic	Complitech	Автоматический мониторинг нарушений ПДД; Оперативная реакция на ДТП;	Невозможность самообучения модели; Высокая стоимость внедрения.
Умный перекрёсток	СпецДорПроект	Приоритет для общественного транспорта; Интеграция с городской ИТС; Снижение времени ожидания на перекрёстках;	Высокая цена внедрения

1.2.2 Выбор инструментов разработки

Приложение предназначено для исследования работы интеллектуальных алгоритмов управления светофорными объектами на перекрёстке при помощи имитационного моделирования.

В настоящее время существует ряд сред имитационного моделирования транспортных потоков с необходимым функционалом в управлении светофорными объектами. Рассмотрим некоторые из них.

Avenue 2.0 [16] — это онлайн сервис моделирования транспортных потоков, который позволяет создавать и анализировать транспортные потоки в городах и других населенных пунктах.

К достоинствам сервиса можно отнести: лёгкость использования, простой и понятный интерфейс; визуализацию транспортных потоков.

Недостаток сервиса: стоимость – Avenue 2.0 не является бесплатным сервисом, и его использование может быть достаточно дорогим для пользователей.

В целом, сервис Avenue 2.0 это хороший инструмент для моделирования транспортных потоков, который отличается понятным интерфейсом, позволяющим быстро и эффективно анализировать данные.

TrafficData [17] — это программное обеспечение для сбора, анализа и визуализации данных транспортных потоков.

Достоинства программного обеспечения TrafficData: удобный интерфейс; широкий функционал, визуализация результатов, анализ пробок и трафика; удобство использования.

Недостатки программного обеспечения TrafficData: не является бесплатным программным обеспечением; для работы с TrafficData необходимо иметь определенный опыт в области анализа данных и транспортных потоков.

Таким образом, TrafficData является универсальным программным обеспечением для анализа транспортных потоков. Но его использование может быть дорогим и требует опыта работы с таким программным обеспечением.

LISA+ [18] и INES+ [19] это инструменты для разработки режимов регулирования светофорных объектов, которые обеспечивают высокую точность благодаря использованию современных алгоритмов и методов.

Они позволяют управлять светофорным объектом и интегрировать его с другими системами для управления трафиком. Однако, их использование может потребовать определенного уровня знаний в области управления трафиком.

SUMO [20] используется для моделирования и анализа транспортного потока. Это позволяет исследовать эффективность дорожной сети, разрабатывать алгоритмы управления светофорами и оценивать влияние различных условий на транспортную систему.

К достоинствам можно отнести:

- реалистичное моделирование трафика, которое позволяет создавать детальные модели транспортного потока;
- гибкость и настраиваемость, которые реализуются в большом разнообразии настроек и параметров для моделирования трафика;
- интеграция с реальными данными позволяет использовать настоящие данные о транспортном потоке.

Недостатками программы SUMO являются:

- ресурсоемкость, так моделирование больших городских сетей с большим количеством транспортных средств может быть затратным процессом;
- ограниченность некоторыми сторонами моделирования, которая проявляется в ограничениях в моделировании некоторых параметров транспортной системы, таких как пешеходы, общественный транспорт.

В таблице 2 представлена сводная информация о достоинствах и недостатках сред имитационного моделирования транспортных потоков.

Таким образом, при выборе программы для моделирования трафика отдано предпочтение SUMO.

Таблица 2 – Сводная таблица достоинств и недостатков сред моделирования

Программное обеспечение	Достоинство	Недостаток
Avenue 2.0	Простой интерфейс; 3D-визуализация потоков.	Дорого для частных пользователей.
TrafficData	Широкий функционал анализа; Удобная визуализация.	Требует опыта работы; Высокая цена.

Программное обеспечение	Достоинство	Недостаток
LISA+ / INES+	Точное управление светофорами; - Интеграция с системами.	Сложность для новичков.
SUMO	Реалистичное моделирование трафика; Гибкость моделирования.	Ресурсоемкость; Ограниченная детализация некоторых аспектов моделирования.

Основой разработки стал язык программирования Python, который подошел для данной задачи благодаря уровню абстракции, большому выбору библиотек и многофункциональному API с используемой средой имитационного моделирования. Python позволил реализовывать и тестировать алгоритмы управления светофорным объектом, включая методы машинного обучения и нейронные сети, что было бы значительно сложнее сделать на таких языках, как C++ или Java.

Для реализации агентного подхода используется библиотека Gymnasium, которая предоставляет интерфейс для создания и тестирования различных алгоритмов. В сочетании с пакетом sumo-rl это позволило интегрировать SUMO в процесс обучения наших моделей. Для реализации алгоритма Q-обучения была использована библиотека numru.

В области нейронных сетей, был выбран фреймворк PyTorch, который обеспечил необходимую функциональность при создании архитектуры нейронной сети.

Библиотека TraCI стала основным интерфейсом между интеллектуальными алгоритмами и средой моделирования, позволяя в реальном времени изменять фазы работы светофорных объектов, получать данные о транспортных средствах и характеристиках среды. Дополнительно использовалась библиотека sumolib для работы с сетевыми файлами SUMO и генерации маршрутов движения.

Выбранная совокупность инструментов разработки соответствует требованиям проекта, обеспечивая необходимый функционал для создания пользовательского интерфейса программы для исследования различных интеллектуальных алгоритмов управления светофорными объектами.

					ПГУ 1.090301.02.001 ПЗ	Лист
						13
Изм.	Лист	№ докум.	Подпись	Дата		

1.2.3 Выбор среды разработки

Современные среды разработки предоставляют встроенные инструменты для проверки и отладки программ, управления библиотеками, интеграции с системами контроля версий и других функций, которые упрощают процесс разработки.

Наиболее популярные из них: Visual Studio Code, PyCharm, Jupyter Notebook.

VS Code – это редактор кода от Microsoft, который благодаря расширению Python становится полноценной средой разработки. Он поддерживает автодополнение, отладку, виртуальные окружения, управление зависимостями и терминал. Также можно настроить интерфейс под себя.

PyCharm – это среда разработки от JetBrains, специально созданная для Python. Поддерживает множество функций на встроенном уровне: автодополнение и анализ кода, работу с базами данных и фреймворками.

Jupyter – это удобная среда для работы с кодом и анализа данных, визуализации и создания алгоритмов. Ячейки кода можно запускать по одной, вставлять графики, текст и формулы.

Таким образом, в качестве среды разработки выбор остановился на PyCharm Community.

1.2.4 Выбор архитектуры программы

Приложение состоит из нескольких компонентов, взаимодействующих друг с другом для моделирования транспортных потоков, интеллектуального управления светофорами и анализа данных.

Архитектура включает следующие основные блоки: пользовательский интерфейс; обработчик и генератор входных данных; ядро моделирования (SUMO); интеллектуальный контроллер светофоров; анализатор характеристик моделирования; визуализатор результатов.

Таким образом, структура программы представлена на рисунке 1.

Система моделирования дорожного трафика начинается с интерфейса управления, который обеспечивает взаимодействие пользователя с программой. Через графический интерфейс пользователь может задавать параметры моделирования, такие как выбор алгоритма управления светофорами, количество

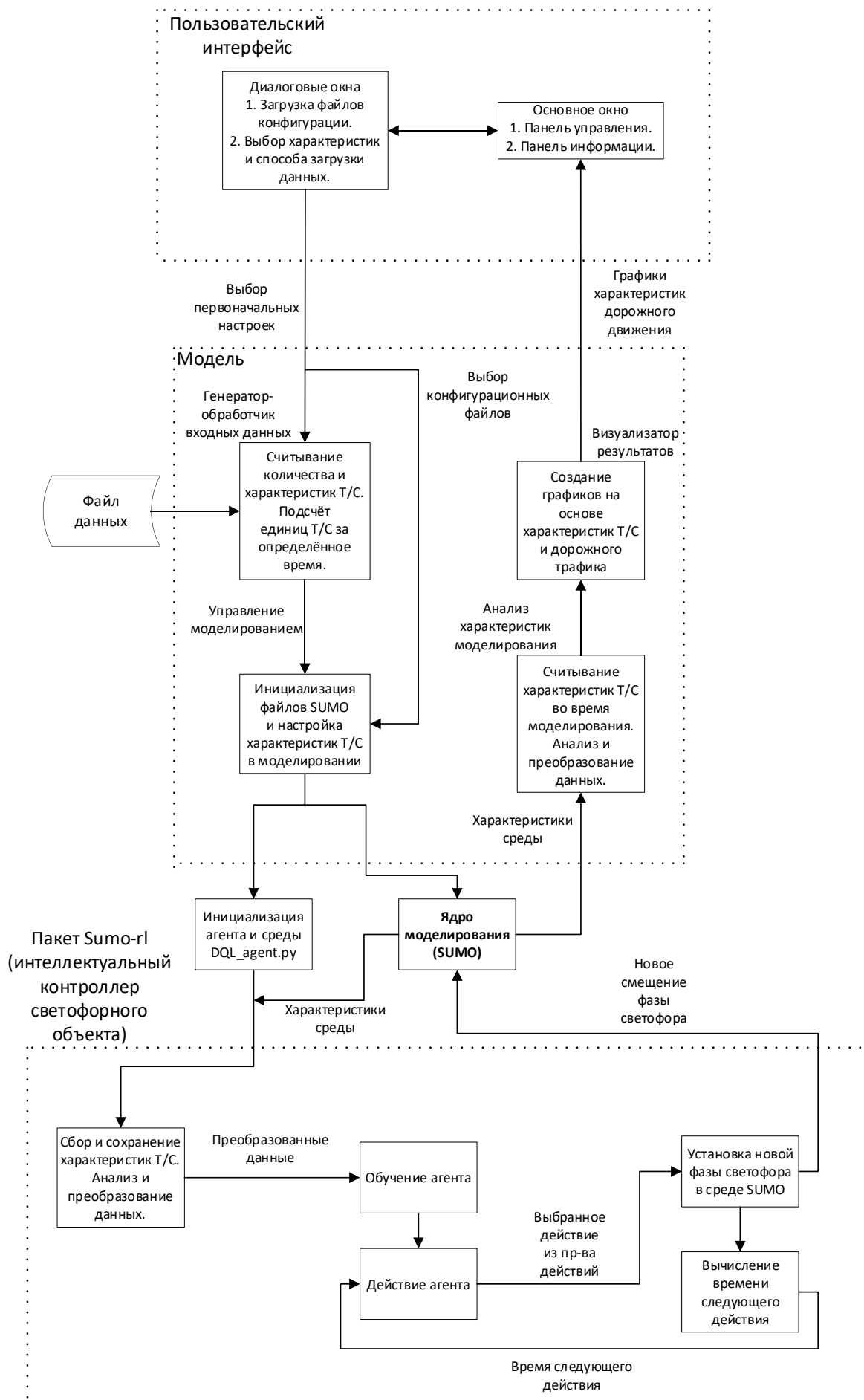


Рисунок 1 – Схема архитектуры программы

Изм.	Лист	№ докум.	Подпись	Дата

эпох обучения и настройки генерации входных данных. Также предусмотрена возможность загрузки уже готовых данных о транспортных потоках или автоматического их создания на основе заданных параметров.

Обработчик и генератор входных данных играют важную роль в подготовке информации для моделирования. В зависимости от настроек пользователя они принимают данные из XML файла либо самостоятельно их генерирует на основе полученных от пользователя данных, затем он преобразует их в формат, понятный SUMO. Так же в этом модуле считывается информация о количестве машин за интервал усреднения [21]. Далее обработанная информация о проездах машин загружается непосредственно в конфигурационные файлы SUMO.

Основу симуляции составляет ядро моделирования, представленное средой SUMO. Она выполняет все вычисления, связанные с движением транспорта по сети. Она инициализирует дорожную среду, управляет перемещением автомобилей и имитирует работу светофоров в соответствии с выбранным алгоритмом. Для динамического управления используется библиотека TraCI, которая обеспечивает связь между SUMO и интеллектуальными алгоритмами управления. Она позволяет в реальном времени изменять фазы работы светофоров.

Одновременно с этим интеллектуальный контроллер управления светофорами анализирует дорожную ситуацию, принимает решения о смене фаз работы светофорного объекта и адаптируется к изменяющимся условиям трафика, постоянно обучаясь на основе поступающих данных.

В частности, используется алгоритм Q-обучения, который принимает решения, основываясь на Q-таблице. Также может применяться подход с использованием нейронной сети, которая обучается предсказывать награды за выбор фазы работы светофорного объекта.

Для оценки эффективности алгоритмов применяется модуль анализа характеристик дорожного движения. Он собирает метрики, такие как среднее время ожидания, длина очередей и количество автомобилей на перекрёстке. Это позволяет проводить сравнение различных алгоритмов управления и определять наиболее эффективный из них. Все результаты сохраняются для последующего анализа.

Последним элементом является визуализатор результатов, который выполняет функции графического отображения полученных данных. Он позволяет строить графики зависимости различных параметров от времени моделирования. Таким образом, система не только моделирует транспортные потоки, но и представляет полученные результаты в виде графиков.

Взаимодействие между компонентами организовано по следующей схеме: пользователь через интерфейс управления инициирует моделирование, обработчик данных подготавливает необходимую информацию, ядро SUMO выполняет моделирование, интеллектуальный контроллер оптимизирует работу светофоров, а визуализатор предоставляет результаты в форме графиков зависимостей характеристик дорожного движения от времени моделирования.

Таким образом, выбранная архитектура и используемые инструменты разработки позволяют связать между собой анализ данных транспортных потоков, управление симуляцией и обучение интеллектуального алгоритма.

1.3 Проектирование и разработка

1.3.1 Механика моделирования в среде SUMO

SUMO предоставляет разнообразные методы моделирования, такие как: командная строка для запуска моделирования, конфигурационные файлы для настройки среды, API для программного управления и сбора данных, а также графический интерфейс sumo-gui для визуализации моделирования. Благодаря этим инструментам SUMO обеспечивает возможность выбора наиболее удобного способа решения задач моделирования транспортных систем.

Ниже представлены основные конфигурационные файлы, необходимые для запуска модели.

Файл сценария .sumocfg – основной файл конфигурации, который определяет параметры моделирования и ссылки на другие конфигурационные файлы.

Файл сети дорог .net.xml – определяет геометрию дорог, перекрестки, полосы движения и другие элементы сети. Он описывает структуру дорожной сети и связи между различными участками дорог.

					ПГУ 1.090301.02.001 ПЗ	Лист
						17
Изм.	Лист	№ докум.	Подпись	Дата		

Файл маршрутов flows.xml определяет потоки транспортных средств в симуляции. Он содержит информацию о типах транспортных средств, их количестве и маршрутах.

Файл маршрутов .row.xml, определяющий правила генерации транспортных средств в моделировании. Он определяет интервал, начальные, промежуточные и конечные пункты движения транспортных средств.

Файл результатов .output.xml создаётся в процессе моделирования в SUMO. Он содержит информацию о характеристиках симуляции по результатам моделирования.

При использовании графического интерфейса sumo-gui, необходимо загрузить соответствующие файлы сценария и конфигурации, а затем визуализировать моделирование.

1.3.2 Механизмы управления светофорными объектами в SUMO

В SUMO светофорные объекты описаны в файле конфигурации сети .net.xml:

```
<tlLogic id="B1" type="static" programID="0" offset="0">
    <phase duration="42" state="GGGggrrrrrGGGggrrrrr"/>
    <phase duration="3" state="yyyyyrrrrrryyyyyrrrrr"/>
    <phase duration="42" state="rrrrrGGGggrrrrrGGGgg"/>
    <phase duration="3" state="rrrrryyyyyrrrrryyyyy"/>
</tlLogic>
```

Этот фрагмент кода описывает логику работы светофорного объекта в SUMO.

Файл содержит теги и атрибуты:

- <tlLogic> – тег обозначает начало описания логики светофора;
- id="B1" – уникальный идентификатор светофора. В данном случае, светофор имеет идентификатор "B1";
- type="static" – тип логики светофора. "static" означает, что светофор работает в статическом режиме, т.е., фазы и их продолжительность заранее заданы и не изменяются динамически;
- programID="0" – идентификатор программы светофора. В данном случае

используется программа с идентификатором "0";

- offset="0" – смещение начала цикла светофора (в секундах);
- <phase> – этот тег определяет одну из фаз светофора;
- duration="42" – продолжительность фазы в секундах;
- state="GGGggrrrrrGGGggrrrrr" – состояние светофора в данной фазе.

Таким образом, приведенный код представляет собой описание четырехфазного статического светофора с заранее заданными продолжительностями фаз и состояниями каждой фазы.

Всего существует два основных способа изменения фаз светофора в SUMO.

Непосредственное изменение фазы светофора через конфигурационный файл .net.xml. Данный способ не позволяет изменять фазу светофора в процессе моделирования. Так же прямое изменение фаз светофоров в файле .net.xml может привести к нежелательным результатам и некорректному моделированию, так как не учитывает внутренние механизмы управления светофорами, которые предоставляет SUMO.

Управление светофорами через программный интерфейс, такой как TraCI, является рекомендуемым способом в SUMO. TraCI предоставляет программный интерфейс для взаимодействия с SUMO в режиме реального времени, что позволяет динамически управлять светофорами, учитывая текущее состояние трафика.

Преимущества использования TraCI для управления светофорами в SUMO:

- динамическое управление, так как TraCI позволяет управлять светофорами в реальном времени в зависимости от текущих условий дорожного движения;
- TraCI позволяет при помощи программы определять логику управления светофорами, к примеру, временные интервалы и другие параметры;
- интеграция с внешними системами, реализуемая в TraCI, позволяет использовать для управления светофорами внешние системы или алгоритмы, что эффективно для решения задач управления светофорными объектами.

1.3.3 Формат данных и интеграция с реальными источниками

Данные для моделирования из реальных источников должны быть представлены в формате XML файла flows.xml.

В нём должна содержаться информация о id транспортного средства, его точке начала (from), конца (to) движения на перекрёстке, а также о времени отправления (depart).

Пример исходных данных представлен ниже.

```
<trip id="0" depart="0.0" from="A1B1" to="B1A1" />
```

```
<trip id="1" depart="0.0" from="A1B1" to="B1B0" />
```

Этот фрагмент кода описывает два проезда перекрёстка автомобилем. Файл содержит теги и атрибуты:

id – уникальный идентификатор автомобиля;

depart – время выезда из начальной точки (в секундах от начала моделирования);

from – идентификатор начального участка дороги;

to – идентификатор конечного участка.

Такой формат позволяет показать движение автомобилей в процессе моделирования. Использование реальных маршрутов и времени выезда даёт возможность проводить моделирование, которое приближено к действительности, а также анализировать конкретные участки дорог и оценивать эффективность интеллектуальных алгоритмов управления трафиком.

1.3.4 Агентный подход в управлении светофорными объектами

Агентный подход в управлении светофорами представляет собой концепцию, в рамках которой каждый светофор рассматривается как автономный агент, способный принимать решения на основе локальной информации о трафике.

Агентный подход в управлении светофорами также может использовать методы машинного обучения и нейронные сети для возможного создания эффективных систем управления дорожным движением.

Статья [22] предлагает агентную систему управления светофорами, где агент использует алгоритм адаптивного управления для оптимизации времени работы светофорного объекта. По результатам исследования агентный подход позволяет реализовать динамическое управление и повысить эффективность системы к изменяющимся условиям.

Таким образом, статья подтверждает пользу агентных систем для создания эффективных систем управления трафиком.

Для функционирования агенту нужна среда, с которой он взаимодействует. Она представляет собой перекрёсток и связанную с ним дорожную сеть.

Агент воспринимает окружающую среду с помощью различных датчиков, которые передают ему информацию о текущем состоянии трафика.

На основе полученной информации агент принимает решения о настройке сигналов светофоров. Эти решения могут включать в себя изменение длительности зеленого сигнала, установку определенной последовательности сигналов и т.д. После принятия решений агент воздействует на среду, изменяя параметры работы светофоров. Эти изменения могут влиять на поведение транспортного потока.

В процессе моделирования агент непрерывно получает обратную связь от среды о результате своих действий. На основе этой обратной связи агент может изменять свои будущие решения для оптимизации работы светофоров. Таким образом, агент взаимодействует со средой с целью достижения оптимальной работы светофоров, возможно улучшая тем самым эффективность и безопасность дорожного движения.

Общая концепция агентного подхода показана на рисунке 2.

Для связи агента с средой SUMO может использоваться библиотека OpenAI-Gym [23]. Gym является набором инструментов, разработанным для реализации агентного подхода.

Использование Gym позволяет создавать среды моделирования, в которых возможно проводить управление светофорными объектами.

В библиотеке Gym можно задать пространства действий и наблюдений для агента. Также определить правила взаимодействия с средой, которые включают получение вознаграждений и обновление состояния среды.

Пространство действий (Action space).

Это множество всех возможных действий, которые агент может совершать в среде. Оно определяет действия, которые агент может выполнить в ответ на свое текущее состояние.

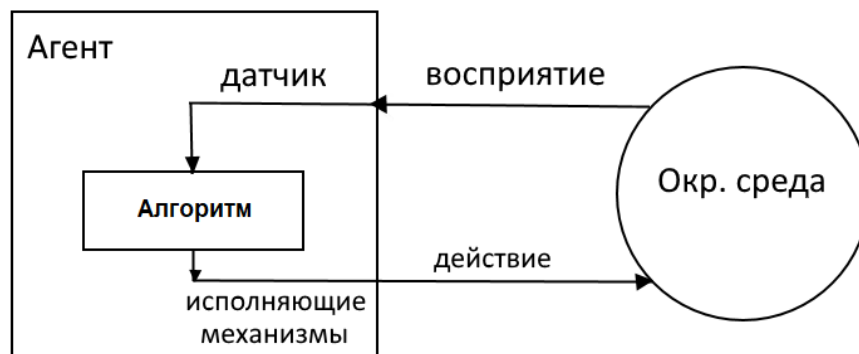


Рисунок 2 – Схема взаимодействия агента и среды

Для агента, управляющего светофорным объектом, пространство действий может представлять из себя набор длительностей фаз светофора или набор смещений фаз светофора.

Пространство наблюдений (Observation space).

Это множество всех возможных состояний, которые агент может наблюдать в среде. Пространство наблюдений содержит информацию, которую агент может получать из среды моделирования.

Ниже проанализированы распространённые алгоритмы управления светофорными объектами, которые возможно реализовать при помощи агентного подхода.

1.3.5 Алгоритмы управления светофорными объектами: статические, адаптивные, интеллектуальные

Рассмотрение алгоритмов начинается с классических способов управления светофорными объектами, а именно со статических алгоритмов.

В классическом управлении светофорами применяются программы с фиксированным циклом, где набор временных интервалов для каждого направления движения устанавливается статически.

Программы с фиксированным циклом обладают несколькими преимуществами. Простота реализации делает эти программы легко внедряемыми, что подходит небольшим городам. Предсказуемость работы обеспечивается статичным циклом. Такие алгоритмы остаются стабильными в условиях постоянного транспортного потока.

Но, программы с фиксированным циклом имеют ряд серьёзных недостатков, которые связаны с неэффективностью при переменной нагрузке. Так как такие программы не могут адаптироваться к изменениям в интенсивности движения.

Таким образом, программы с фиксированным циклом применяются на перекрёстках, где не требуется адаптация к изменениям в транспортном потоке. Ещё она может быть полезна в районах, где доступ к современным технологиям и сложным системам управления ограничен или не оправдан.

Адаптивные алгоритмы в управлении светофорными объектами основаны на подстройке к текущим характеристикам дорожного трафика. Различные адаптивные алгоритмы [24] могут собирать информацию о таких характеристиках дорожного движения как: очереди на подъездах к перекрёстку, среднее время ожидания проезда перекрёстка на каждом направлении, интенсивность движения и др.

Адаптивные алгоритмы работы светофорных объектов обладают рядом преимуществ по сравнению со статическими. Они позволяют учитывать текущую ситуацию на дороге в реальном времени и изменять продолжительность фаз.

Но у адаптивных алгоритмов есть и недостатки. Для их работы необходимо использование более сложной логики, а также постоянного сбора и анализа данных, что требует дополнительных вычислительных мощностей. Также недостатком является чувствительность к качеству поступающих данных, так как при ошибках в измерениях или запаздывании информации система может принимать ошибочные решения, ухудшая дорожную ситуацию.

Далее рассматриваются интеллектуальные алгоритмы управления светофорными объектами, которые в отличие от статических и адаптивных имеют память.

1.3.5.1 Алгоритм Q-обучения

Q-обучение (Q-learning) является одним из распространённых методов обучения с подкреплением. Он используется для обучения агентов принятию решений в среде, где есть вознаграждение может быть получено только после выполнения действий.

Таблица 3 – Формат Q-таблицы

Состояние (observation space) представляет собой описание ситуации среды, в которой находится агент. В данном случае состояние содержит 12 значений (0, 45, 9, 9, 0, 0, 1, 1, 0, 0, 2, 2), представляющие собой характеристики дорожного движения и светофорного объекта: продолжительность фазы, минимальная продолжительность зелёной фазы, очередь и плотность движения и т.д.

Строка награды за действие (action space) длиной в количество возможных действий. В примере есть 40 действий, представленных в виде списка. Например, в 1 строке таблицы 1 все награды равны 0 за все действия, кроме одного, для которого награда составляет минус 3. Это означает, что для большинства действий награда еще не просчитана, а для одного действия награда отрицательна, это говорит о том, что агент уже когда-то был в этом состоянии и выбрал это действие.

Каждая ячейка Q-таблицы обновляется в процессе обучения с подкреплением на основе формулы обновления Q-значения, которая зависит от полученных наград и действий агента. Цель обучения с подкреплением состоит в том, чтобы обновлять значения в Q-таблице таким образом, чтобы агент мог принимать оптимальные решения для максимизации награды в среде.

Когда агент взаимодействует со средой и обновляет значения Q-таблицы на основе полученного опыта, то он постепенно улучшает свою стратегию, выбирая оптимальные действия для каждого состояния и максимизируя награду.

Формула обновления Q-функции (формула 1) представлена ниже.

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha * (r_t + \gamma * \max Q(S_{t+1}, A_t) - Q(S_t, A_t)), \quad (1)$$

где $Q_{st,at}$ – текущее оценочное значение Q-функции;

S_t – множество действий;

A_t – множество состояний;

α – коэффициент обучения, определяющий величину шага обновления Q-функции;

r_t – награда, полученная после выполнения действия A_t в состоянии S_t и перехода в следующее состояние;

γ – коэффициент дисконтирования, который определяет важность будущих наград по сравнению с немедленными;

$\max Q(S_{t+1}, a)$ – максимальное значение Q-функции для всех действий a в следующем состоянии S_{t+1} .

Таким образом, в Q-обучении агент обновляет Q-функцию, используя максимальное значение функции для всех возможных действий в следующем состоянии S_{t+1} .

Это позволяет агенту выбирать действие с максимальной ожидаемой наградой в следующем состоянии. Это делает его стратегию жадной.

Обновление Q-значения происходит с целью минимизации ошибки (разница между полученной наградой и предсказанным значением Q-функции) с учетом коэффициента обучения α . Это позволяет агенту корректировать свои оценки исходя из полученного опыта.

В обучении с подкреплением и нейронных сетях часто используются различные коэффициенты (называемые гиперпараметрами), такие как alpha, beta, epsilon и decay, которые играют важную роль в обучении и управлении поведением алгоритма. Значения всех коэффициентов находятся в диапазоне от 0 до 1.

Alpha (α) представляет собой коэффициент скорости обучения,

который определяет, насколько агент будет учитывать новую информацию при обновлении своей стратегии.

Beta (β) является коэффициентом дисконтирования, который определяет, как агент учитывает будущие вознаграждения при принятии решений в текущем состоянии.

Epsilon (ϵ) – это коэффициент исследования, который определяет вероятность случайного выбора действия вместо выбора оптимального действия.

Decay является коэффициентом затухания, который определяет скорость уменьшения значения epsilon с течением времени.

Научные исследования позволяют оценить эффективность применения Q-обучения в управлении светофорными объектами.

К примеру, как показано в работах [8,9] исследователей, использование Q-обучения демонстрирует перспективность при оптимизации транспортных систем.

Наиболее рейтинговые репозитории [25...27] на GitHub, посвящённые управлению светофорными объектами, демонстрируют эффективность применения агентного подхода и Q-обучения в управлении светофорами.

Важно, что большинство репозиторий на GitHub с проектами, которые используют SUMO и направлены на оптимизацию фаз работы светофора используют также пакет sumo-rl [28].

Он является инструментом, разработанным для соединения SUMO с интеллектуальными алгоритмами на языке программирования Python.

Sumo-rl также обеспечивает интерфейс для взаимодействия со средой SUMO и реализацией алгоритмов обучения с подкреплением. Благодаря наличию API возможно собирать и обрабатывать различные характеристики движения в среде, что облегчает задачу тестирования и обучения алгоритмов.

Таким образом, на основе анализа Q-обучения можно выделить некоторые плюсы и минусы использования обучения с подкреплением в управлении светофорным объектом.

Достоинства обучения с подкреплением:

– Q-обучение адаптивно, то есть оно позволяет улучшить эффективность алгоритма управления светофорным объектом

в зависимости от изменяющихся характеристик дорожного движения;

– алгоритм Q-обучения основывается на основе накопленного ранее опыта. Это позволяет учитывать прошлые результаты и принимать правильные решения в будущем;

– этот метод относительно прост в реализации и может быть добавлен в систему управления светофорами.

Недостатки:

– Q-обучение требует времени для накопления опыта и обучения, прежде чем может показать оптимальные результаты;

– при увеличении размерности пространства состояний и действий Q-обучение может столкнуться с проблемами масштабирования и обучения;

– качество обучения может зависеть от выбора гиперпараметров, таких как коэффициент обучения и параметр жадности, что требует внимательного подбора для достижения оптимальных результатов;

– алгоритм обучения неэффективен при наличии непредсказуемых или аномальных ситуаций, к примеру авария на перекрёстке;

– полный учет всех факторов, которые влияют на транспортное движение (например, пешеходы, общественный транспорт), может быть сложной задачей при применении Q-обучения на перекрестках.

1.3.5.2 Нейронные сети

Нейронные сети представляют собой инструмент в области машинного обучения, который имитирует работу мозга, используя математические функции. Сети состоят из нескольких слоев нейронов, которые организованы в структуру. Эта структура передает информацию между слоями. Основным элементом нейронной сети - нейрон, который принимает входные сигналы, взвешивает их и передает результат через функцию активации следующим нейронам.

Ниже подробно рассмотрена полносвязная нейронная сеть, также известная как многослойный персептрон.

Она является классической моделью нейронной сети, состоящей из нескольких слоев нейронов,

каждый из которых полностью связан с нейронами предыдущего и последующего слоев. Сеть состоит из одного входного слоя, одного или нескольких скрытых слоев и одного выходного слоя. Пример структуры такой нейронной сети показан на рисунке 3.

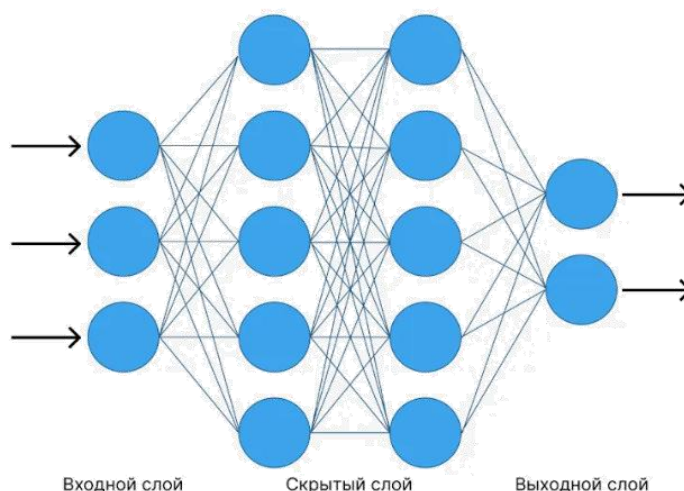


Рисунок 3 – Схема полносвязной нейронной сети

Далее рассмотрены принципы работы нейронной сети на примере полносвязной нейронной сети.

На нейроны входного слоя поступает информация из пространства наблюдений, а на выходном слое информация записывается в пространство действий агента.

Сеть может быть произвольного размера по количеству слоёв, входной слой должен быть размерностью равном пространству наблюдений. Выходной слой должен быть размерностью, равной пространству действий.

Каждый нейрон в скрытых и выходных слоях активируется с помощью функции активации, которая позволяет нейронной сети выучивать сложные зависимости в данных.

Функция активации в нейронной сети принимает входные данные и решает, будет ли нейрон "включен" или "выключен" при работе сети. Используемая функция ReLU (формула 2) обнуляет все отрицательные значения входа, оставляя положительные значения без изменений.

$$f(x) = \max(0, x), \quad (2)$$

где x – аргумент функции.

Изм.	Лист	№ докум.	Подпись	Дата

Q-функция анализирует информацию о состоянии перекрестка и оценивает, насколько выгодно каждое возможное действие из пространства действий.

После этого сеть сравнивает свои текущие оценки с прошлыми и определяет точность своих предыдущих прогнозов. На основе этого сравнения сеть корректирует свои веса нейронов, чтобы уменьшить разницу между оценками и фактическими результатами, тем самым улучшая будущие прогнозы. Этот процесс повторяется многократно, позволяя сети учиться на своих ошибках и становиться все более эффективной в управлении трафиком.

Преимущества применения нейронной сети заключаются в предсказании возможных наград или штрафов за каждое возможное действие.

Но для эффективного обучения требуется большое количество данных о трафике, а настройка параметров сети может быть сложной задачей, требующей специальных знаний и вычислительных ресурсов.

Анализ научных публикаций позволяет оценить эффективность использования нейронных сетей в системах управления светофорами.

В управлении светофорными объектами нейронные сети могут быть эффективны. К примеру, результаты исследования в статье [10] показывают, что их использование снижает плотность потока на перекрёстке.

В статье авторов [22] также показана эффективность использования нейронных сетей в управлении светофорными объектами.

Репозиторий [29] включает проект по управлению светофорами через нейросеть с глубоким обучением. В результате моделирования нейронная сеть сокращает суммарное время ожидания на перекрёстке по сравнению со статическим планом работы светофорного объекта.

На основании представленных данных возможно реализовать алгоритмы с использованием sumo-gui и провести исследование их эффективности при различных сценариях транспортных потоков. Эффективность разработанных нами алгоритмов предстоит проверить путем проведения серии экспериментов и тестирования на синтезированных данных.

1.3.6 Разработка модуля управления и настройки агента

Модуль управления и настройки агента разработан для реализации агентного подхода в программе. Ниже представлен код, который реализует агентный подход для управления светофорами.

```
while not done["__all__"]:  
    actions = {ts: ql_agents[ts].act() for ts in ql_agents.keys()}  
    s, r, done, _ = env.step(action=actions)  
    for agent_id in ql_agents.keys():  
        ql_agents[agent_id].learn(next_state=env.encode(s[agent_id], agent_id),  
reward=r[agent_id])
```

Обучение и взаимодействие со средой происходит в цикле `while not done["__all__"]`; то есть пока моделирование не будет завершено. Агент получает действия `actions` и вызывает функцию `env.step`, которая возвращает информацию о среде: состояние `s`, награду `r`, переменную выполнения моделирования `done`.

Затем через цикл происходит выбор агента и у него вызывается метод `learn`, в который передаётся информация о текущем состоянии агента `s[agent_id]`, а также о его текущей награде `r[agent_id]`. Для связи агента с средой SUMO используется библиотека OpenAI-Gym.

Ниже представлен код, который реализует агента для управления светофорами с помощью SUMO.

```
ql_agents = {  
    ts: QLAgent(  
        starting_state=env.encode(initial_states[ts], ts),  
        state_space=env.observation_space,  
        action_space=env.action_space,  
        alpha=alpha,  
        gamma=gamma,  
        exploration_strategy=EpsilonGreedy(  
            initial_epsilon=0.3, min_epsilon=0.09, decay=0.3  
        ),  
        for ts in env.ts_ids  
    )  
}
```

В начале создаётся словарь агентов `ql_agents` с агентом `ts`. Ему передаются следующие параметры: начальное состояние агента `starting_state`;

пространство состояний среды `state_space`; пространство действий среды `action_space`; параметр скорости обучения `alpha`; параметр дисконтирования `gamma` и стратегия исследования `exploration_strategy`.

Используется стратегия `EpsilonGreedy`, которая выбирает случайное действие с вероятностью `epsilon` и наилучшее известное действие с вероятностью `1 - epsilon`. `Initial_epsilon`, `min_epsilon` и `decay` являются параметрами этой стратегии.

Пространство действий дискретно, оно содержит 40 возможных действий – изменение фазы светофора в диапазоне от минус 20 до плюс 20 секунд. К примеру, действие плюс 5 увеличит зелёный сигнал на 5 секунд, а другое выбранное действие минус 10 сократит его на 10 секунд.

Ниже представлена часть кода, которая задаёт пространство действий.

```
self.action_space = spaces.Discrete(40, start=-20)
```

Пространство наблюдений включает в себя данные о текущей длине фазы светофора, средней скорости, плотности потока, очереди. Ниже представлен пример кода, задающего пространство наблюдений.

```
def __call__(self) -> np.ndarray:
    """Return the default observation."""
    phase_id = [1 if self.ts.green_phase == i else 0 for i in
range(self.ts.num_green_phases)] # one-hot encoding
    phase_duration = [x.duration for x in self.ts.green_phases]
    min_green = [0 if self.ts.time_since_last_phase_change < self.ts.min_green +
self.ts.yellow_time else 1]
    queue = [round(x, 1) for x in self.ts.get_lanes_queue()]
    observation = np.array(phase_id + phase_duration + min_green + queue +
density, dtype=np.float32)
    return observation
```

В пространство наблюдения `observation` входит номер фазы `phase_id`, продолжительность фазы `phase_duration`, минимальная продолжительность зелёной фазы `min_green`, очередь `queue` и плотность движения `density`. Исходя из этих данных формируется `observation`, который потом используется для оценки ситуации в среде и управлении продолжительностью фазы светофора.

1.3.7 Разработка алгоритма Q-обучения

Для использования агента с алгоритмом Q-обучения необходимо реализовать несколько функций.

Функция вознаграждения определяет, какое вознаграждение агент получает за действие в состоянии. Выбор действия будет осуществляться с помощью функции выбора действия, которая выбирает действие агента на основе случайного числа.

Функция обучения будет использовать формулу Q-обучения для обновления значений на основе полученного опыта.

В коде функция обучения выглядит следующим образом:

```
def learn (self, next_state, reward, done=False):
    if next_state not in self.q_table:
self.q_table[next_state] = [0 for _ in range(self.action_space.n)]
        s = self.state
        s1 = next_state
        a = self.action

self.q_table[s][a] = self.q_table[s][a] + self.alpha * (
            reward + self.gamma * max(self.q_table[s1]) - self.q_table[s]
        [a]
self.state = s1
self.acc_reward += reward
```

Сначала функция проверяет, есть ли состояние next_state в Q-таблице. Если его нет, то создается новая запись в Q-таблице для этого состояния.

Затем функция сохраняет текущее состояние s, следующее состояние s1 и действие a. Это необходимо для обновления значений в Q-таблице.

Далее происходит обновление значения Q-функции для пары состояния и действия, используя формулу обучения.

Затем, текущее состояние агента self.state приравнивается к следующему состоянию s1, так как агент перешел в новое состояние.

Накопленная награда увеличивается на значение полученной награды self.acc_reward += reward, чтобы отслеживать общую сумму наград за обучение.

Агент принимает решения на основе функции вознаграждения, которая определяет награду или штраф, который получает агент за выполнение действия.

Функцией вознаграждения может быть длина очереди за момент симуляции на каждой из полос.

```
def get_total_queued(self):  
    return sum(self.sumo.lane.getLastStepHaltingNumber(lane) for lane  
in self.lanes)
```

Либо функцией вознаграждения может быть величина среднего времени ожидания.

```
def _diff_waiting_time_reward(self):  
    ts_wait = sum(self.get_accumulated_waiting_time_per_lane()) / 100.0  
    reward = self.last_measure - ts_wait  
    self.last_measure = ts_wait  
    return reward
```

В целом функцией вознаграждения может быть любая функция, которая отражает пропускную способность перекрёстка.

Во время обучения агенту важно находить баланс между исследованием новых стратегий и использованием уже известных. Функция choose определяет, какое действие агент будет предпринимать.

```
def choose(self, q_table, state, action_space):  
    """Choose action based on epsilon greedy strategy."""  
    if np.random.rand() < self.epsilon:  
        action = int(action_space.sample())  
    else:  
        action = np.argmax(q_table[state])  
    self.epsilon = max(self.epsilon * self.decay, self.min_epsilon)  
    # print(self.epsilon)  
    return action
```

Функция проверяет возникновение случайного события. Если это число меньше значения параметра epsilon, то агент принимает случайное действие из всего множества доступных действий. После выбора действия параметр epsilon уменьшается на коэффициент затухания decay.

Взаимодействие этих функций формирует цикл обучения с подкреплением, который позволяет агенту постепенно обучаться.

1.3.8 Разработка нейронной сети

Используется полносвязная трёхслойная нейронная сеть. На нейроны

входного слоя поступает информация из пространства наблюдений, а на выходном слое информация записывается в пространство действий агента. Ниже приведён пример создания слоёв нейронной сети.

```
def __init__(self, state_size, action_size):
    super(QNetwork, self).__init__()
    self.fc1 = nn.Linear(state_size, 64)
    self.fc2 = nn.Linear(64, 64)
    self.fc3 = nn.Linear(64, action_size)
```

Сеть имеет 3 слоя. Входной слой fc1 имеет размерность state_size (размер пространства наблюдений в среде) и выходных нейронов – 64. Скрытый слой fc2 имеет одинаковую размер для входных и выходных нейронов равным 64. Выходной слой fc3 имеет размер входных нейронов равным 64 и выходных нейронов равным action_size (размер пространства действий).

Каждый нейрон в скрытых и выходных слоях активируется с помощью функции активации, которая вводит позволяет нейронной сети выявлять зависимости в данных. Это происходит при помощи функции forward.

```
def forward(self, x):
    x = torch.relu(self.fc1(x))
    x = torch.relu(self.fc2(x))
    x = self.fc3(x)
    return x
```

Функция активации в нейронной сети принимает входные данные и решает, будет ли нейрон активен или нет при работе сети. В данном случае тоже используется функция ReLU.

Ниже приведён анализ действий нейронной сети на примере функции learn.

```
def learn(self, state, action, next_state, reward, done=False):
    state = torch.tensor(state, dtype=torch.float32).to(self.device)
    next_state = torch.tensor(next_state, dtype=torch.float32).to(
        self.device)
    q_values = self.q_network(state)
    q_values_next = self.q_network(next_state)
    q_value = q_values[action]
    if done:
        target = reward
    else:
```

```

        target = reward + self.gamma * torch.max(q_values_next)
        loss = nn.MSELoss()(q_value, target)
        self.optimizer.zero_grad()
        self.lossList.append(int(loss))
        loss.backward()
        self.optimizer.step()
        self.acc_reward += reward

```

Сначала входные состояния `state` и `next_state` преобразуются в тензоры PyTorch типа и перемещаются на устройство (GPU или CPU), указанное в `self.device`.

Затем текущее состояние `state` передается через сеть `self.q_network`, которая возвращает Q-значения для всех действий. Затем вычисляются Q-значения для следующего состояния `next_state`.

Из Q-значений выбирается значение, соответствующее действию `action`, выполненному в текущем состоянии.

Целевое значение `target` вычисляется с использованием формулы Q-обновления. Если моделирование завершено `done == True`, целевым значением является только награда `reward`. Иначе к награде добавляется максимальное Q-значение для следующего состояния.

Функция потерь между текущим Q-значением и целевым значением вычисляется с использованием метода `nn.MSELoss`. Он измеряет среднеквадратичное отклонение между предсказанным Q-значением и целевым значением. Минимизация функции потерь является целью обучения модели.

Градиент потери вычисляется с помощью функции обратного распространения ошибки `loss.backward`, а затем параметры сети обновляются с использованием оптимизатора `self.optimizer.step`.

Во время работы этих функций происходит сохранение результатов обучения нейронной сети на этом шаге симуляции и обновление весов нейронов.

Веса являются одним ключевым компонентом нейронной сети. Они определяют, насколько каждый входной сигнал важен для вычисления выхода нейрона. Они настраиваются в процессе обучения сети с использованием алгоритмов оптимизации Adam.

Это позволяет модели минимизировать ошибку прогнозирования на данных модели и улучшить ее способность к обобщению на новые данные.

1.4 Тестирование и анализ результатов

1.4.1 Критерии оценки эффективности алгоритма

Оценка эффективности интеллектуальных алгоритмов управления зависит от нескольких ключевых критериев. Вот несколько из них.

Одним из основных критериев эффективности является способность системы справляться со скоплением машин на направлениях перекрёстка, что обеспечивает беспрепятственное движение по дороге. Такой критерий используется в исследовании [2].

Эффективные алгоритмы стремятся минимизировать среднее время ожидания для водителей. Данный критерий используется в исследовании А.А. Агафонова, А.С. Юмаганова и В. В. Мясникова [31].

В нашей работе использовано два основных критерия для оценки эффективности адаптивных алгоритмов управления светофорами.

Средняя очередь на перекрёстке позволяет оценить, насколько хорошо алгоритм справляется с задачей предотвращения образования заторов и равномерного распределения потоков.

Среднее время ожидания позволит оценить, насколько эффективно алгоритмы сокращают время, которое участники движения проводят в ожидании перед светофорами.

Выбор этих критериев основан на стремлении к оптимизации двух параметров: максимизации интенсивность движения и минимизации времени ожидания, что в сумме даёт эффективное управление дорожным движением.

Кроме этих критериев, в некоторых исследованиях рассматриваются такие параметры, как количество остановок транспорта, уровень выбросов и средняя скорость движения. Стоит отметить, что все характеристики транспортного потока находятся в прямой или обратной зависимости друг от друга.

1.4.2 Анализ эффективности Q-обучения

Обучение агента происходило в течение 1000 эпох, были взяты следующие гиперпараметры: $\alpha = 0.5$, $\beta = 0.6$, $\text{decay} = 0.001$. Коэффициент ϵ в начале моделирования составлял 1 и каждую эпоху обучения уменьшался на 0.001 пункта.

На рисунке 4 представлены результаты обучения агента на разных эпохах.

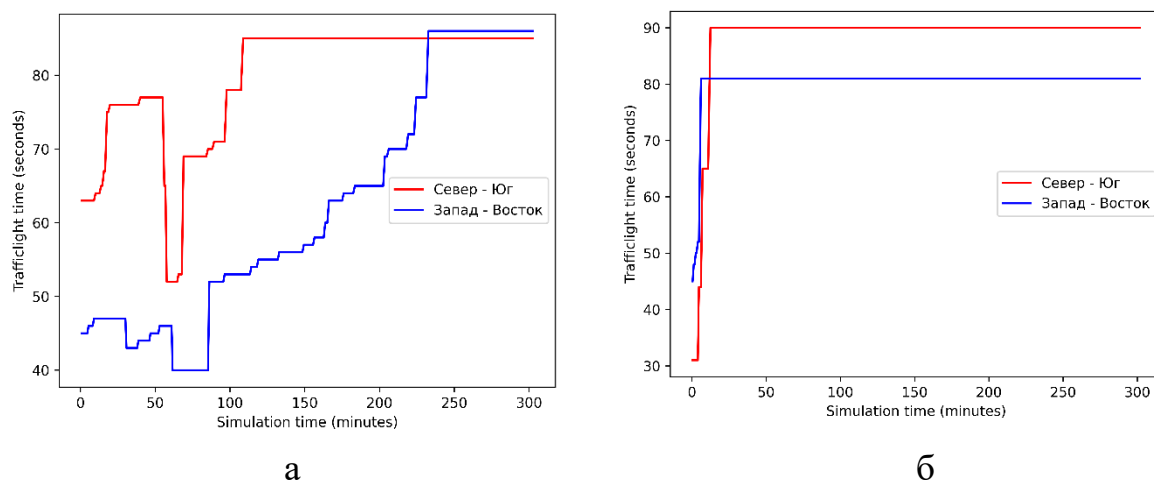


Рисунок 4 – Графики продолжительности фазы светофорного объекта
(а – после 1 эпохи обучения, б – после 1000 эпохи обучения)

Видно, что агент на рисунке 4 (а) изменяет фазу работы светофора достаточно часто, это связано с тем, что коэффициент ϵ , определяющий вероятность случайного выбора действия, равен 1 и агент выбирает новое действие, исследуя пространство действий.

На рисунке 4 (б) видно, что агент практически не изменяет фазу работы светофора так как коэффициент исследования ϵ равен 0. В данном случае агент ориентируется только на данные, полученные в результате предыдущих эпох моделирования.

Ниже представлены графики сравнения характеристик дорожного движения с участием и без участия агента с алгоритмом Q-обучения.

Синий цвет показывает среднее характеристики без участия агента, а красный - с участием агента. Среднее время ожидания измеряется в секундах. Средняя очередь и среднее количество машин измеряется в единицах ТС. Их различие состоит в том, что средняя очередь считается как сумма ТС, чья скорость меньше 0.1 метра в секунду.

На рисунке 5 отражено среднее время ожидания автомобилей на каждом из направлений перекрёстка после обучения агента с Q-обучением.

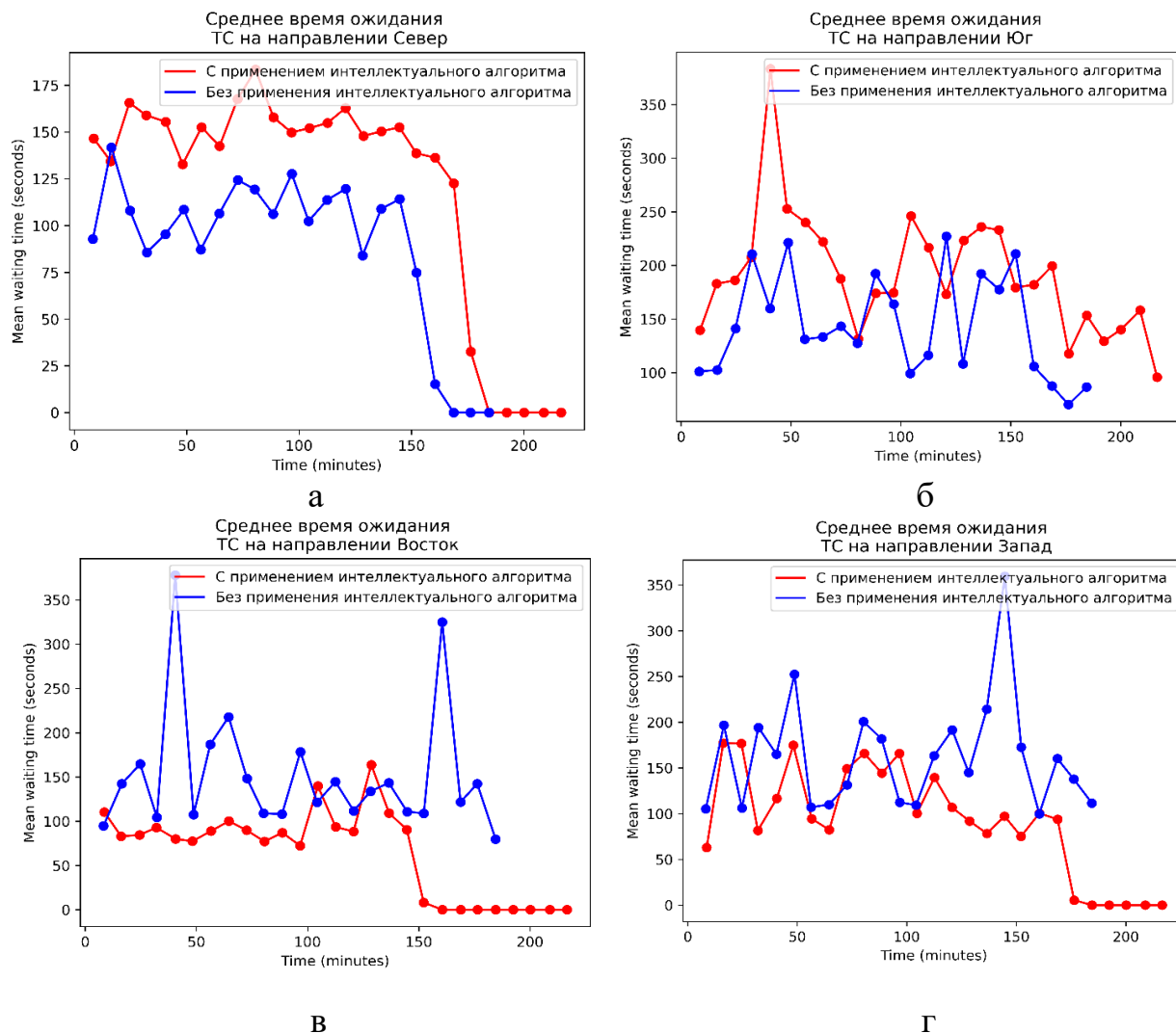
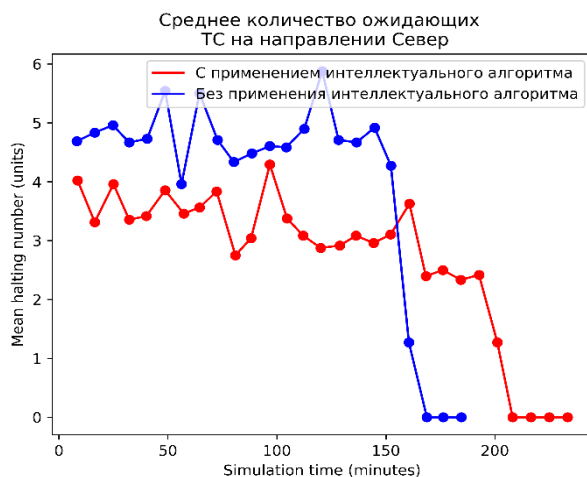


Рисунок 5 – Графики среднего времени ожидания для каждого направления после обучения Q – модели (а – направление север, б – направление юг, в – направление восток, г – направление запад)

Агент уже обладает достаточными знаниями для управления светофорными объектами. На графиках видно, что агент снижает время ожидания транспортных средств на направлениях восток и запад, но повышает его на направлениях север и юг.

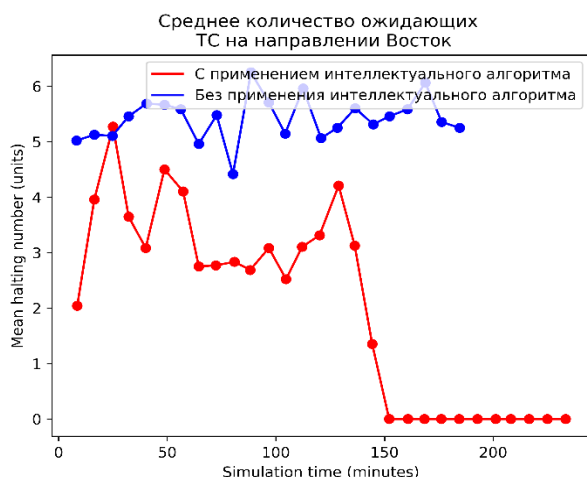
На рисунке 6 показаны графики средней длины очереди для каждого направления перекрёстка за период после обучения агента с Q-обучением.



а



б



в

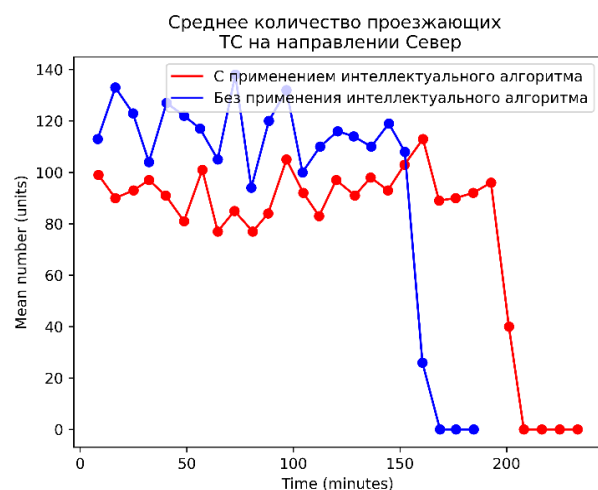


г

Рисунок 6 – Графики средней длины очереди по направлениям после обучения Q – модели (а – направление север, б – направление юг, в – направление восток, г – направление запад)

По графикам видно, что агент стремится успешно сокращает очереди на всех направлениях, особенно успешно на направлении восток и запад.

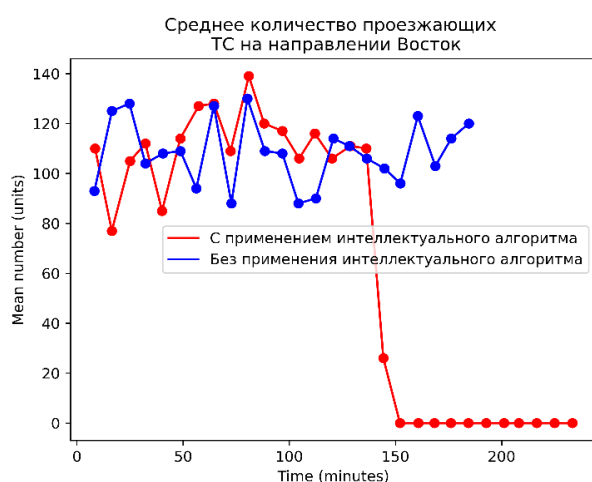
На рисунке 7 отражено среднее количество машин для каждого направления перекрёстка от времени моделирования после обучения агента с Q-обучением.



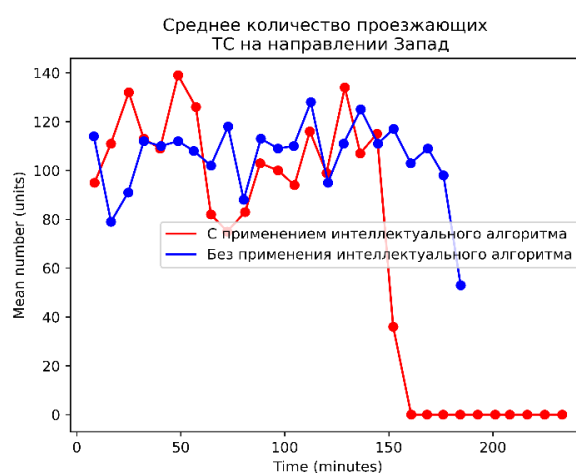
а



б



в



г

Рисунок 7 – Графики среднего количества машин по направлениям после обучения Q – модели (а – направление север, б – направление юг, в – направление восток, г – направление запад)

По графикам видно, что агент не допускает накопления машин на перекрёстке, оставляя их количество примерно таким же, как и на кривой, отображающей количество машин без участия агента.

Были проведены расчёты эффективности применения Q-обучения для задачи управления светофорными объектами. Они представлены в таблице 4.

Таблица 4 – Средние значения времени ожидания по направлениям для алгоритма Q-обучения

Направление	Среднее время ожидания при использовании обучения с подкреплением (секунд)	Среднее время ожидания без использования обучения с подкреплением (секунд)
Север	136.4	88.5
Восток	48.8	145.4
Юг	193.7	137.7
Запад	55.45	131.5

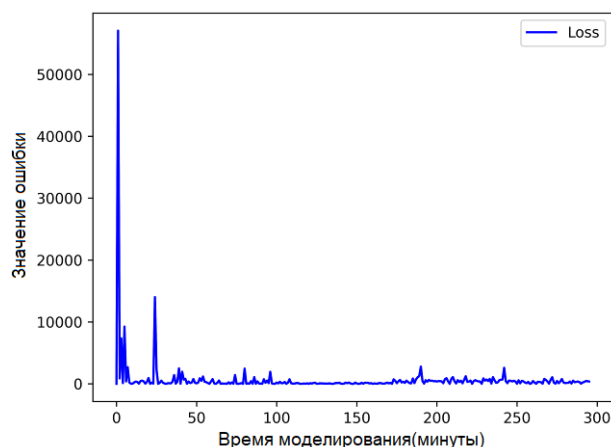
Таким образом, в процентном соотношении для направления север обучение с подкреплением хуже в эффективности на 54 %. Для направления восток Q-обучение оказалось эффективней на 66 %. На направлении юг обучение с подкреплением ухудшило результат управления на 40 %. Но на направлении запад алгоритм показал эффективность 57 %. В общем, по всем направлениям обучение с подкреплением оказалось эффективней на 29.5 %.

В итоге, использование Q-обучения в управлении светофорами является инструментом, который может оптимизировать управление дорожным трафиком на отдельных направлениях движения на перекрёстке.

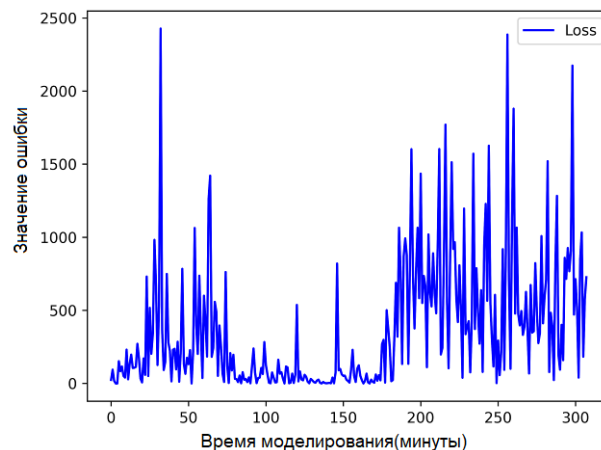
1.4.3 Анализ эффективности нейронной сети

Агент обучался в течение 1000 эпох, были взяты следующие коэффициенты $\alpha = 0.5$, $\beta = 0.6$, $\text{decay} = 0.001$. Коэффициент ϵ в начале моделирования составлял 1 и каждую эпоху обучения уменьшался на 0.003 пункта. Это так же, как и в обучении с подкреплением было сделано с целью обеспечить агенту достаточный уровень исследования среды и избежать преждевременной остановки на определённой стратегии действий.

На рисунке 8 представлены два графика зависимости ошибки от времени моделирования для начала обучения (рисунок 8 (а)) и для конца обучения (рисунок 8 (б)).



а



б

Рисунок 8 – Графики зависимости ошибки от времени моделирования
(а – после 1 эпохи обучения, б – после 1000 эпохи обучения)

Видно, что на начале обучения ошибка достигает внушительных значений, так как агент ещё не имеет достаточных знаний о среде. После некоторого обучения ошибка существенно снижается, что говорит о том, что агент успешно обучается в среде.

На рисунке 9 показана зависимость продолжительности фазы работы светофорного объекта от времени моделирования.

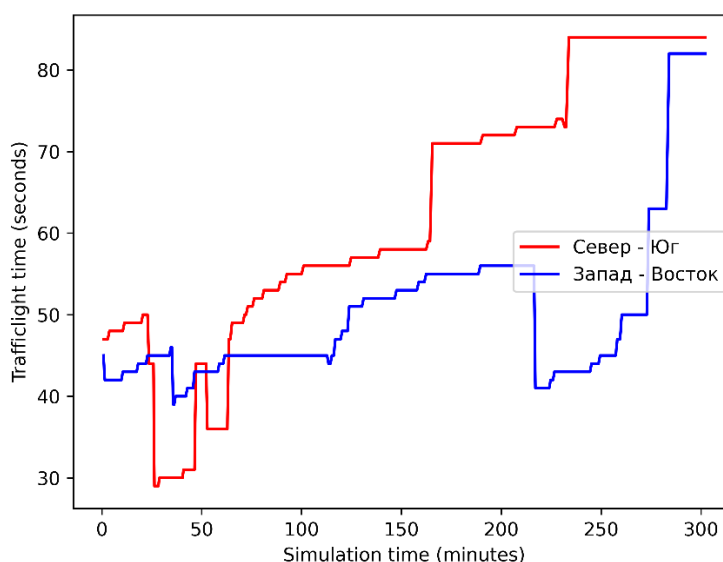


Рисунок 9 – График продолжительности фазы работы светофорного
объекта

По сравнению с аналогичным графиком для обучения с подкреплением (рисунок 4 (б)) видно, что нейросеть даже после обучения продолжает изменять продолжительности фаз, в отличие от обучения с подкреплением, где установились стабильные продолжительности фаз.

На рисунке 10 отражено среднее время ожидания автомобилей на каждом из направлений перекрёстка после обучения агента с нейронной сетью.

Синий цвет показывает среднее характеристики без участия агента, а красный - с участием агента. Среднее время ожидания измеряется в секундах. Средняя очередь и среднее количество машин измеряется в единицах ТС. Их различие состоит в том, что средняя очередь считается как сумма ТС, чья скорость меньше 0.1 метров в секунду.



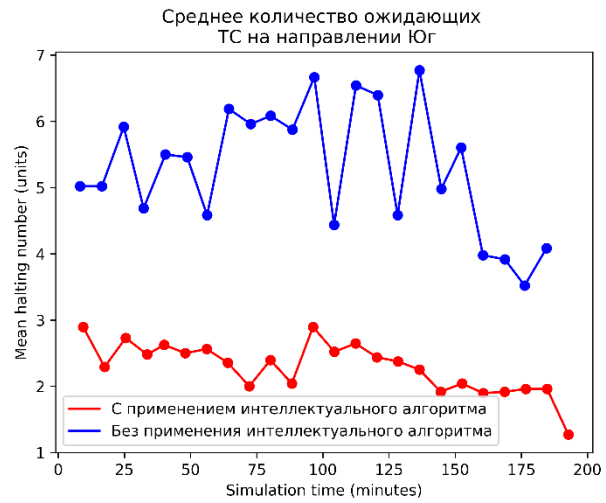
Рисунок 10 – Графики среднего времени ожидания машин по направлениям после обучения нейронной сети (а – направление север, б – направление юг, в – направление восток, г – направление запад)

Видно, что агент, использующий нейросеть, лишь немного снижает время ожидания на перекрёстке.

На рисунке 11 отражена длина очереди на светофоре на каждом из направлений перекрёстка после обучения агента с нейронной сетью.



а



б



в

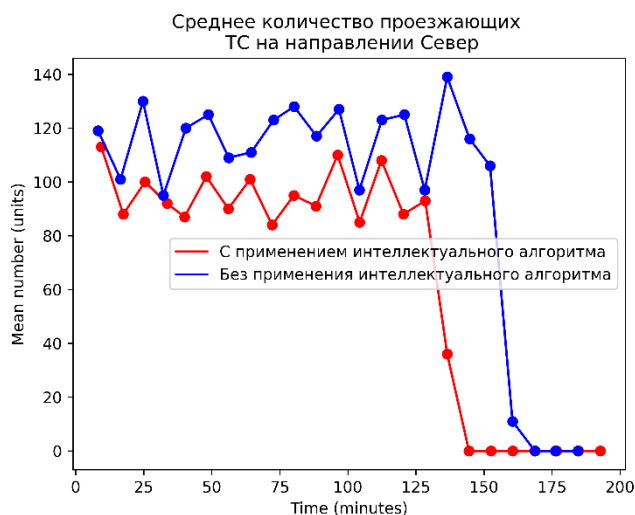


г

Рисунок 11– Графики средней длины очереди по направлениям после обучения нейронной сети (а – направление север, б – направление юг, в – направление восток, г – направление запад)

Агент на всех направлениях уменьшил среднюю длину очереди на перекрёстке. Особенно это заметно на направлении юг (рисунок 11 (б)).

На рисунке 12 отражено среднее количество машин на каждом из направлений перекрёстка после обучения агента с нейронной сетью.



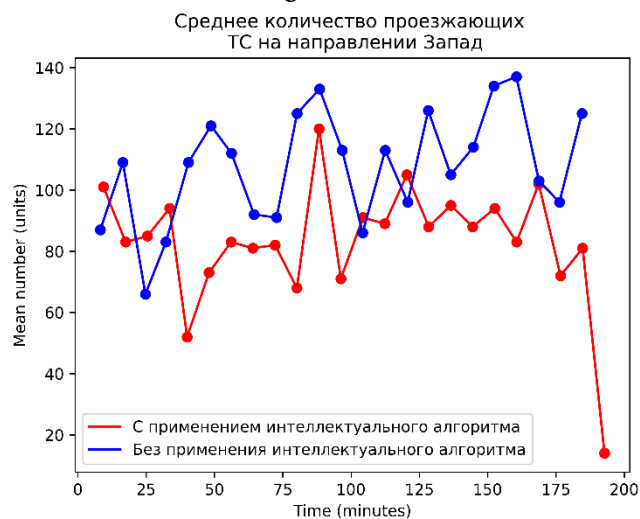
а



б



в



г

Рисунок 12 – Графики среднего количества машин по направлениям после обучения нейронной сети (а – направление север, б – направление юг, в – направление восток, г – направление запад)

Видно, что агент оставляет количество машин по направлению примерно одинаковым, разница видна лишь на некоторых участках, к примеру на рисунке 12 (б) агент уменьшил среднее количество машин в районе 150 – 200 минут моделирования.

Анализ эффективности применения нейронной сети в процентном соотношении приведён в таблице 5.

Изм.	Лист	№ докум.	Подпись	Дата

Таблица 5 – Средние значения времени ожидания по направлениям для нейронной сети

Направление	Среднее время ожидания при использовании нейронной сети (секунд)	Среднее время ожидания без использования нейронной сети (секунд)
Север	84.7	86.7
Восток	143.7	142.2
Юг	131.1	155.6
Запад	145.6	150.8

Таким образом, в процентном соотношении для направления север нейронная сеть лучше в эффективности на 2 %. Для направления восток применение нейронной сети оказалось хуже на 1 %. На направлении юг нейронная сеть улучшила результат управления на 15 %. На направлении запад нейронная сеть показала эффективность 3 %. В общем, по всем направлениям обучение с подкреплением оказалось эффективней на 19 %.

На рисунке 13 проведено сравнение эффективности нейронной сети и Q-обучения. Сравнение будет происходить по критерию среднего времени ожидания.

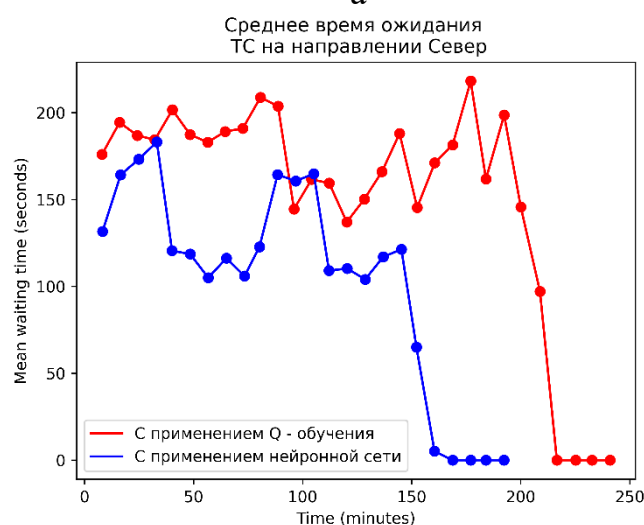
На рисунке видно, что Q-обучение эффективней справляется на направлениях восток и запад на 60 % и 53 % соответственно. В свою очередь нейронная сеть эффективней, чем Q-обучение на направлениях север и юг на 47.2 % и 47.3 % соответственно. В среднем Q-обучение эффективней нейронной сети на 18 %.



а



б



в



г

Рисунок 13 (а, б, в, г) – Сравнительные графики среднего времени ожидания для каждого направления (а – направление север, б – направление юг, в – направление восток, г – направление запад)

2 Охрана труда

В данном разделе рассматриваются требования по обеспечению безопасности жизнедеятельности и охраны труда пользователей ПЭВМ, участвующих в разработке и эксплуатации программного обеспечения, а также способы минимизации вредных факторов на здоровье.

2.1 Характеристика потенциальных рисков при работе с ПК

В соответствии с ГОСТ 12.0.003-2015 «Система стандартов безопасности труда. Электронно-вычислительная техника. Общие требования безопасности», при эксплуатации ПК выделяются следующие основные риски:

- риски поражения электрическим током из-за: неисправной проводки; отсутствия заземления; попадания жидкости на технику; короткого замыкания при перегрузке сети;
- риски заболеваний опорно-двигательной системы из-за: неправильного положения тела; неправильной высоты стула и стола;
- риски заболеваний зрения из-за: использования ПК без перерывов для глаз; неправильной яркости экрана и расположения монитора.

2.2 Оборудование рабочего места

Основные требования, которые должны соблюдаться при пользовании рабочем местом:

- расстояние от монитора до глаз должно быть не менее 500 – 700 мм. Монитор должен располагаться ниже уровня глаз под углом 15-20°;
- высота стола должна быть 680 – 800 мм (в зависимости от роста пользователя);
- кресло должно иметь регулируемую высоту сиденья и спинку, а также глубину сиденья 400 - 450 мм.

На рисунке 14 показана правильная организация рабочего пространства для ПК.

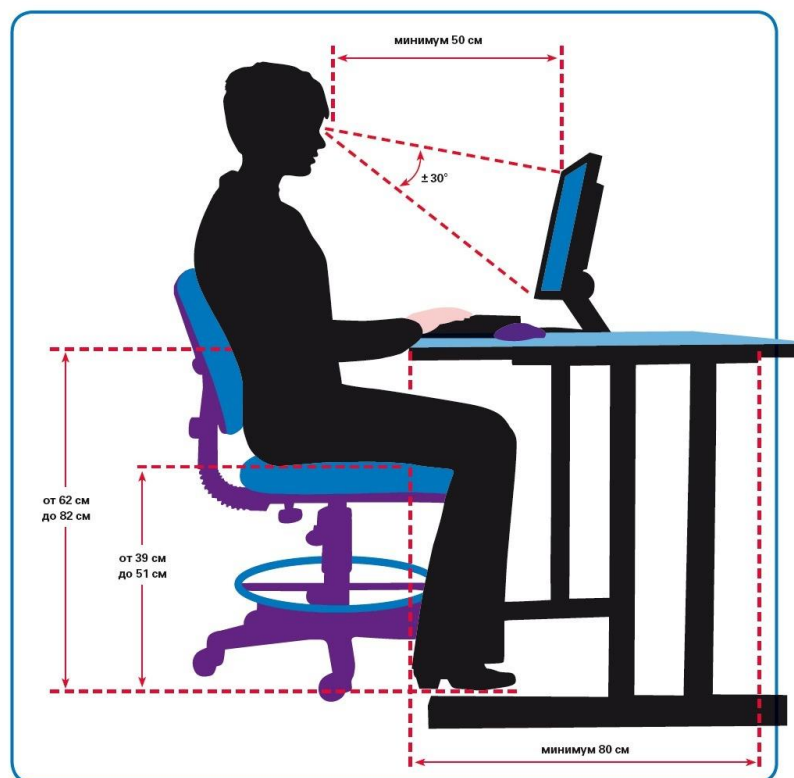


Рисунок 14 – Правильная организация рабочего места

Рабочее место с монитором должно гарантировать пользователю удобную работу с ПК в положении сидя. При нахождении за ПК должны выполняться следующие условия: прямая спина с опорой на спинку кресла; расслабленные плечи; кисти и запястья в прямом положении; ноги должны устойчиво стоять на полу.

2.3 Освещённость рабочего места

Для обеспечения здоровья органов зрения на рабочем месте должны соблюдаться стандарты освещения. Согласно санитарным правилам, общая освещённость рабочего места должна составлять не менее 400 люкс. Источники освещения должны обеспечивать равномерный рассеянный свет без бликов на мониторе. Цветовая температура источников освещения должна составлять 4000 – 5000К, что соответствует белому цвету. Недопустимо применение мерцающих источников света и расположение в поле зрения пользователя ярких источников света.

2.4 Электробезопасность

Электробезопасность при работе с ПК регламентируется ГОСТ 12.1.019-2017 «Электробезопасность». Исходя из стандарта, всё электрооборудование должно иметь защитное заземление. ГОСТ предусматривает использование только исправного оборудования с целой изоляцией и компонентами.

Кабели должны быть проложены в местах, где они не будут подвергаться механическим воздействиям и повреждениям. Стандарт требует, чтобы все электрические компоненты были защищены от случайного прикосновения, а корпуса электрооборудования были заземлены.

Пользователи оборудования должны быть обучены правилам электробезопасности и оказании первой помощи при поражении электрическим током.

2.5 Пожарная безопасность

Основные риски пожарной безопасности связаны с возможным перегревом оборудования, короткими замыканиями и возгоранием компонентов. Для предотвращения опасных ситуаций необходимо поддерживать исправное состояние электрооборудования с средствами защиты. Особое внимание необходимо обратить на правильную организацию рабочего пространства - компьютеры должны располагаться на расстоянии не менее 0.5 метра от отопительных приборов и горючих материалов. В помещениях с компьютерами необходимо наличие средств пожаротушения.

Для повышения противопожарной безопасности необходимо соблюдать следующие правила: проходы и запасные выходы не должны быть перекрыты посторонними предметами; в здании должна быть установлена автоматическая пожарная сигнализация и планы эвакуации;

2.6 Санитарно-гигиеническая характеристика производства

Санитарно-гигиеническая характеристика производства регламентируется в соответствии с СанПиН 1.2.3685-21 и другими нормативными документами.

Основными критериями являются: влажность и температура воздуха в помещении, количество света, скорость движения воздуха.

Рабочие столы должны располагаться не более, чем в 6 метрах от оконных проемов. Окна и рабочие места должны регулярно очищаться от пыли и загрязнений. Температурный режим в помещении должен поддерживаться на уровне 22 – 25 °С при влажности воздуха около 60 %. Необходимо наличие вентиляции или системы кондиционирования. Уровень шума не должен превышать 65 дБ.

В помещении необходимо обеспечить: доступ к питьевой воде; доступ к огнетушителям (1 единицу на 50 м³ помещения); аптечки первой помощи.

Все сотрудники должны проходить инструктажи по технике безопасности не реже 1 раза в год.

					ПГУ 1.090301.02.001 ПЗ	Лист
						51
Изм.	Лист	№ докум.	Подпись	Дата		

2 Стартап проект

Стартап проект направлен на разработку и исследование применения интеллектуальных алгоритмов управления светофорными объектами. Система будет адаптироваться к условиям дорожного трафика, тем самым оптимизируя движение на перекрёстке.

Целью деятельности стартапа является получение прибыли для улучшения благосостояния его участников и улучшение эффективности движения на перекрёстках за счёт применения современных технологий. Основной аудиторией являются государственные структуры и частные компании, занимающиеся разработкой решений в области ИТС.

2.1 Анализ рынка

Рынок ИТС в России за последние годы демонстрирует устойчивый рост. Этому способствуют исследования в этой области и федеральные программы.

По данным CNews, инвестиции в решения для транспорта с 2019 г. превысили 100 млрд ₽. Объём рынка цифровых транспортных решений к концу 2024 г. оценивается в 40–50 млрд ₽. В ходе национального проекта «Безопасные качественные дороги» в 2024 г. было выделено 3,15 млрд ₽ на внедрение ИТС в 62 городах в 56 субъектах России.

Факторы роста рынка ИТС и умных городов:

- увеличение числа транспортных средств требует внедрения интеллектуальных решений для управления дорожным движением;
- государственные программы, которые представляют собой инвестиции в транспортную инфраструктуру для повышения эффективности дорожного движения.

По прогнозам, к 2027–2029 гг. рынок российских ИТС будет расти ежегодно на 8–10 % за счёт инвестиции в ИТС проекты, развития инфраструктуры и внедрения интеллектуальных для прогнозирования трафика в реальном времени.

2.2 Целевая аудитория и каналы взаимодействия

Целевой аудиторией стартапа являются муниципальные органы власти, заинтересованные в улучшении транспортной инфраструктуры города; коммерческие организации, которые занимаются разработкой, внедрением или исследованием ИТС.

Ценовые предложения для каждой группы будут подбираться индивидуально, в зависимости от масштаба внедряемой системы и потребностей заказчика.

Для коммуникации с целевой аудиторией планируется использовать следующие каналы: участие в профильных выставках и конференциях; сотрудничество с органами власти; публикации в журналах и профессиональных сообществах; демонстрация возможностей системы.

К каналам взаимодействия также можно отнести рассылку коммерческих предложений, прямые встречи с заинтересованными лицами, на которых будет продемонстрированы возможности системы.

После заключения договора обязательным будет сопровождение внедрения системы. Команда будет заниматься настройкой и развёртыванием системы на перекрёстке.

Наличие плана внедрения и сопровождения продукта позволит обеспечить удовлетворённость клиентов и эффективное взаимодействие в будущем.

2.3 Источники доходов и расходов

Стартовый капитал стартапа составляет 300 000 Р, финансируемых ООО «ЦИФРАЛЕКТ». Первый контракт в 300 000 Р покрывает размер стартового капитала. По прогнозам, в первый год доход достигнет 1 000 000 Р при расходах на уровне 570 000 Р. Это приведёт к прибыли 430 000 Р.

Точка безубыточности достигается после заключения 2 – 3 контрактов. При средней выручке 250 000 – 300 000 с одного контракта критический уровень расходов будет покрыт. При 2 – 3 контрактах в год выручка составит около 900 000 Р, что даёт запас для дальнейшего развития.

					ПГУ 1.090301.02.001 ПЗ	Лист
						53
Изм.	Лист	№ докум.	Подпись	Дата		

Рентабельность собственного капитала в стартапе равна 75,4 %, а чистый оборотный капитал составляет около 100 000 Р. Потребность в дополнительном финансировании при текущих темпах реализации проектов отсутствует, но для ускорения масштабирования возможен привлечение до 500 000 Р.

Распределение расходов выглядит следующим образом: на разработку программного обеспечения планируется направить 250 000 Р, маркетинг — 100 000 Р, прочие операционные затраты — 220 000 Р. В результате, при 2 – 3 исполненных контрактах чистая прибыль составит порядка 330 000 Р.

Таким образом, при текущем темпе роста стартап выйдет на прибыль уже на 11-м месяце работы, обеспечивая 75,4 % годовой рентабельности без привлечения внешних инвестиций. При дополнительном финансировании развитие проекта может ускориться в два раза, что позволит быстрее расширить клиентскую базу и увеличить объёмы выручки.

2.4 Риски проекта

Стартап может столкнуться с некоторыми рисками, которые могут повлиять на его разработку и доходы. Ниже представлены некоторые из них.

Изменение ситуации на рынке. Возможны экономические спады, изменения приоритетов в финансировании программ развития ИТС и как следствия снижение спроса на ИСТ решения.

Меры реагирования: заключение долгосрочных контрактов с заказчиками, гибкость в ценообразовании и корректирование решений под разных клиентов.

Появление новых технологий. Быстрое развитие технологий может сделать продукт устаревшим или менее эффективным по сравнению с конкурентами.

Меры реагирования: постоянное обновление функционала программы, отслеживание развития технологий, проведение тестовых внедрений новых функций.

Деятельность конкурентов. Крупные компании могут предлагать более привлекательные по цене и функционалу решения.

Меры реагирования: создания долгосрочных B2G контрактов и подстраивание решения под нужды каждого клиента.

					ПГУ 1.090301.02.001 ПЗ	Лист
						54
Изм.	Лист	№ докум.	Подпись	Дата		

Недостаток финансирования. Ограниченные ресурсы могут замедлить развитию проекта.

Меры реагирования: привлечение инвестиций, оптимизация расходов и участие в конкурсах и программах.

2.5 Ключевые ресурсы

Для успешной реализации стартапа необходимо наличие нескольких ресурсов, которые обеспечивают разработку, сопровождение и масштабирование продукта. Ниже представлены некоторые из них.

Команда специалистов. Основой стартапа является профессиональная команда, которая включает в себя разработчиков ПО, специалистов в области машинного обучения и нейронных сетей, а также специалистов для работы с клиентами.

Вычислительные мощности. Для обучения моделей и обработки данных требуются вычислительные мощности с поддержкой тензорных вычислений. Эти ресурсы важны как при разработке, так и при сопровождении продукта.

Связь с партнёрами. Связи с государственными учреждениями, коммерческими и исследовательскими организациями упрощает внедрение системы и затраты на масштабирование.

Каналы коммуникаций. Продвижение стартапа возможно через профессиональные сообщества, выставки, соцсети, а также прямое взаимодействие с заинтересованными лицами.

Финансовые ресурсы. Базовое финансирование необходимо для покрытия первичных расходов и создания резерва. Возможность дополнительного финансирования в виде инвестиций и грантов повысит надёжность проекта.

Заключение

В ходе выполнения работы изучены способы имитационного моделирования в программе SUMO, интеллектуальные алгоритмы управления светофорами, а также интерфейсы связи алгоритмов со средой моделирования. Получены навыки разработки алгоритмов машинного обучения при помощи библиотеки PyTorch и numpy.

В процессе работы были разработаны два интеллектуальных алгоритма управления светофорным объектом: Q-обучение и нейронная сеть. Затем был проведён анализ их эффективности на основе синтезированных данных.

В результате было выяснено, что использование интеллектуальных методов управления светофорами может улучшить эффективность работы перекрёстка на некоторых направлениях. По результатам применения Q –обучения, сделан вывод, что такой интеллектуальный алгоритм может динамически адаптироваться к изменяющимся условиям дорожного движения.

Также были исследованы современные способы интеллектуального управления светофорами. Это позволило определить наилучший подход к их применению на перекрестках.

В настоящее время данная программа используется в [32] для интеллектуального управления дорожным движением на перекрёстках города Пензы.

В дальнейшем программу можно развить, путём усовершенствования конфигурации нейронной сети. Также можно провести моделирование поведения агента на различных перекрёстках для оценки его эффективности в непривычной среде.

Список использованных источников

1. А. Д. Сальникова, А. В. Баните, Д. Г. Плотников, А. С. Кашталинский. Формирование системы транспортно-пересадочных узлов в городской агломерации // Автоматика на транспорте. – 2023. – Т. 9, № 1. – С. 87–98.
2. И. В. Макаров, Р. Г. Хабибуллин, К. А. Шубенкова. Совершенствование управления транспортными потоками города с использованием имитационного моделирования // Имитационное моделирование. Теория и практика (ИММОД-2009): материалы 4-й Всероссийская научно-практическая конференция, Санкт-Петербург, 21–23 окт. 2009 г. – Санкт-Петербург, 2009.
3. R. A. Shaharuddin, M. Y. Misro. Controlling Traffic Congestion in Urbanised City: A Framework Using Agent-Based Modelling and Simulation Approach // ISPRS International Journal of Geo-Information. – 2023. – Vol. 12, № 6. – P. 226.
4. T. Alghamdi, S. Mostafi, S. Abdelkader, K. Elgazzar. Comparative Study on Traffic Modeling Techniques for Predicting and Simulating Traffic Behavior // Future Internet. – 2021. – Vol. 14. – P. 294.
5. P. Yedavalli, K. Kumar, P. Waddell. Microsimulation analysis for network traffic assignment (MANTA) at metropolitan-scale for agile transportation planning // Transportmetrica A: Transport Science. – 2021. – P. 1278–1299.
6. Т. А. Кузнецов. Оптимизация движения транспортного потока на регулируемом перекрестке с использованием имитационного моделирования // Политехнический молодежный журнал. – 2022. – № 7 (72).
7. Ю. И. Захаров, Е. С. Карнаух. Основные современные инструменты имитационного моделирования транспортных потоков // Вісник Придніпровської державної академії будівництва та архітектури. - 2023. - С. 87-98.
8. И. А. Саленек, Я. А. Селиверстов, С. А. Селиверстов, Н. И. Носкова. Оптимизация циклов светофорного регулирования методом обучения с подкреплением // Сборник научных трудов XXVI Международной научно-практической конференции. - Санкт-Петербург: Санкт-Петербургский политехнический университет Петра Великого, 2023. - 13-14 окт.

9. M. N. Abdullah, H. A. Jeiad, S. S. Hamid. Intelligent Traffic Lights Control Using Q-learning // International Journal of Science and Research (IJSR). 2018. – Vol. 7, № 9. – P. 365-369.

10. О. П. Тимофеева, С. С. Палицына. Обучение нейронной сети интеллектуальной системы управления городскими светофорами // Труды Нижегородского государственного технического университета им. Р.Е. Алексеева. - Нижний Новгород, 2016. - № 1(112).

11. Сетевое адаптивное управление светофорами: успешный кейс Пензы [Электронный ресурс] – URL: <https://www.itsjournal.ru/articles/digital-region/setevoe-adaptivnoe-upravlenie-svetoforami-uspeshnyy-keys-penzy> (дата обращения: 13.05.2025).

12. AVEDEX [Электронный ресурс] – URL: <https://mallenom-tech.ru/> (дата обращения: 25.03.2025).

13. Авангард [Электронный ресурс] – URL: <https://fvf-rbs.ru/> (дата обращения: 25.03.2025).

14. Sentinel Smart Traffic [Электронный ресурс] – URL: <https://complitech.org/> (дата обращения: 25.03.2025).

15. Умный перекрёсток [Электронный ресурс] – URL: <https://www.spetsdor.ru/> (дата обращения: 25.03.2025).

16. Avenue 2.0 [Электронный ресурс] – URL: <http://avenue-app.com/> (дата обращения 26.03.2025).

17. TrafficData [Электронный ресурс] – URL: <https://trafficdata.ru/> (дата обращения 26.03.2025).

18. LISA+ [Электронный ресурс] – URL: <https://www.asudd.com/lisa/> (дата обращения 26.03.2025).

19. INES+ [Электронный ресурс] – URL: <https://www.asudd.com/inesplus/> (дата обращения 26.03.2025).

20. SUMO [Электронный ресурс] – URL: <https://sumo.dlr.de/docs/index.html> (дата обращения 26.03.2025).

21. М. А. Митрохин, А. О. Аляев, Р. И. Лобанов, В. Е. Кирьянов. Моделирование локальных транспортных потоков в условиях неполноты исходных данных // Автоматика на транспорте. – 2023. – № 9(4).

22. P. Rani, D. K. Shaw. Multi-Agent Based Traffic Signal Optimization // International Journal of Innovative Technology and Exploring Engineering (IJITEE). – 2019. – Vol. 8, Issue 11. – September.

23. Gymnasium. [Электронный ресурс] – URL: <https://gymnasium.farama.org/> (дата обращения 27.03.2025).

24. А. Ю. Кретов. Обзор некоторых адаптивных алгоритмов светофорного регулирования перекрестков // Известия Тульского государственного университета. Технические науки. – 2013.

25. James Han. reinforcement-learning-traffic-control [Электронный ресурс] – URL: <https://github.com/ACM-Research/reinforcement-learning-traffic-control> (дата обращения: 27.03.2025).

26. JDGlick. sumo_reinforcement_learning [Электронный ресурс] – URL: https://github.com/JDGlick/sumo_reinforcement_learning (дата обращения: 28.03.2025).

27. mschrader15. reinforcement-learning-sumo. [Электронный ресурс] – URL: <https://github.com/mschrader15/reinforcement-learning-sumo>. (дата обращения 28.03.2025).

28. LucasAlegre. sumo-rl. [Электронный ресурс] – URL: <https://github.com/LucasAlegre/sumo-rl>. (дата обращения 27.03.2025).

29. hammertime89. sumolights. [Электронный ресурс] – URL: <https://github.com/docwza/sumolights> (дата обращения 29.03.2025).

30. В. Д. Шепелев, З. В. Альметова, А. Д. Моор, В. И. Берстенева. Оптимизация работы адаптивных светофоров на основе использования машинного зрения // Вестник Южно-Уральского государственного университета. - 2020. – Т. 14, № 1. – С. 189 – 196.

31. А. А. Агафонов, А. С. Юмаганов, В. В. Мясников. Эффективность адаптивного управления дорожными сигналами в условиях смешанного потока транспортных средств // Материалы IX Международной конференции и молодежной школы. - Самара: Самарский национальный исследовательский университет им. академика С.П. Королёва, 2023. – 17-23 апр.

32. Нейросетевая видеоаналитика автомобильного трафика [Электронный ресурс] – URL: <https://traffic.codeinside.ru> (дата обращения 13.05.2025).

					<i>ПГУ 1.090301.02.001 ПЗ</i>	Лист
						60
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А

(обязательное)

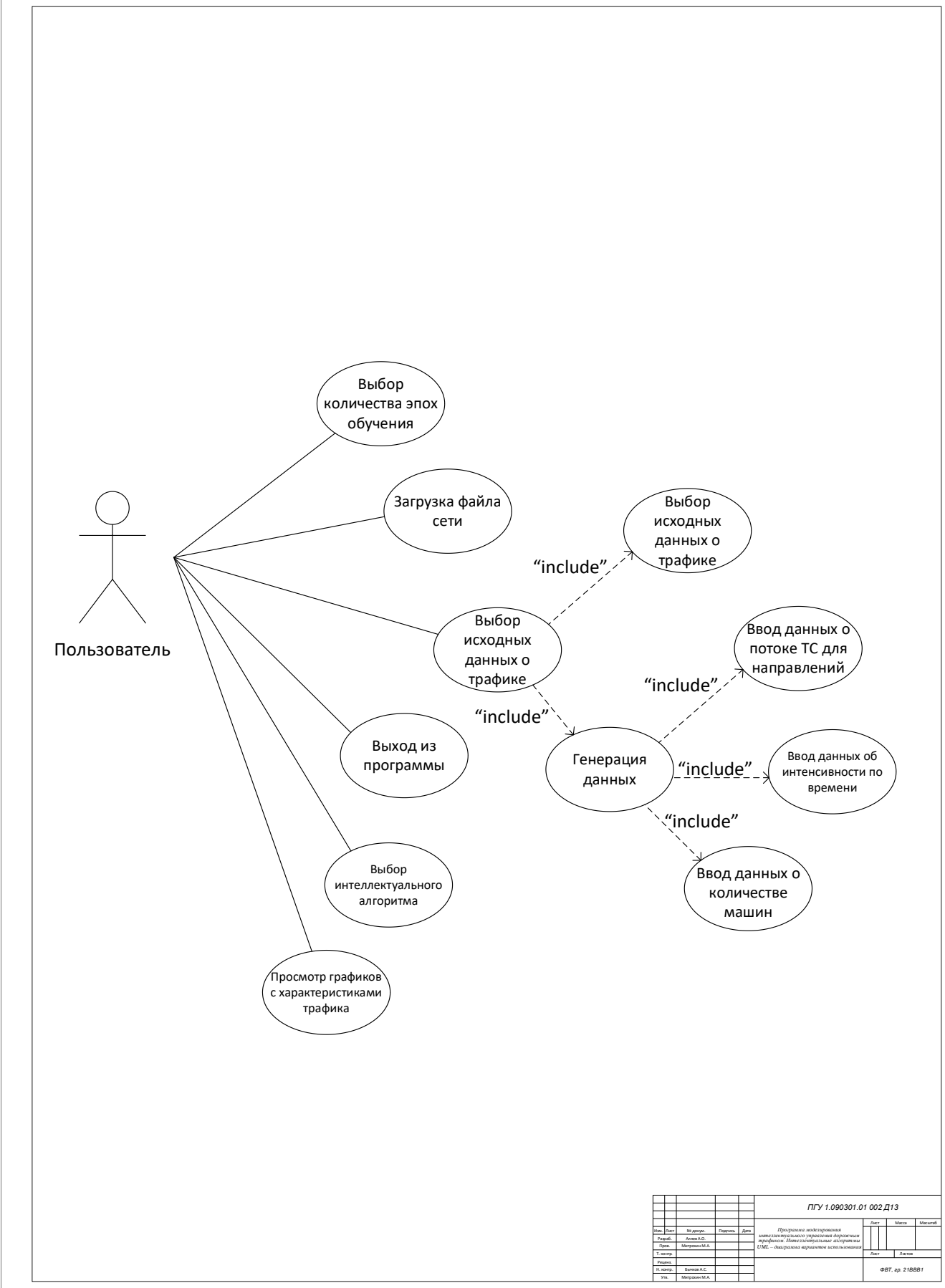
Программа моделирования интеллектуального управления дорожным

трафиком

UML – диаграммы

					ПГУ 1.090301.02.001 ПЗ	Лист
						61
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А.1 - Диаграмма вариантов использования



ПГУ 1.090301.01 002 Д13									
Имя	Лист	№ докум.	Подпись	Дата	Лист	Место	Масштаб		
Имя	Лист	№ докум.	Подпись	Дата	Лист	Место	Масштаб		
Имя	Лист	№ докум.	Подпись	Дата	Лист	Место	Масштаб		
Имя	Лист	№ докум.	Подпись	Дата	Лист	Место	Масштаб		
Имя	Лист	№ докум.	Подпись	Дата	Лист	Место	Масштаб		
Имя	Лист	№ докум.	Подпись	Дата	Лист	Место	Масштаб		
Имя	Лист	№ докум.	Подпись	Дата	Лист	Место	Масштаб		
Имя	Лист	№ докум.	Подпись	Дата	Лист	Место	Масштаб		
Имя	Лист	№ докум.	Подпись	Дата	Лист	Место	Масштаб		

Изм.	Лист	№ докум.	Подпись	Дата

Приложение А.2 - Диаграмма развёртывания

[illegible]

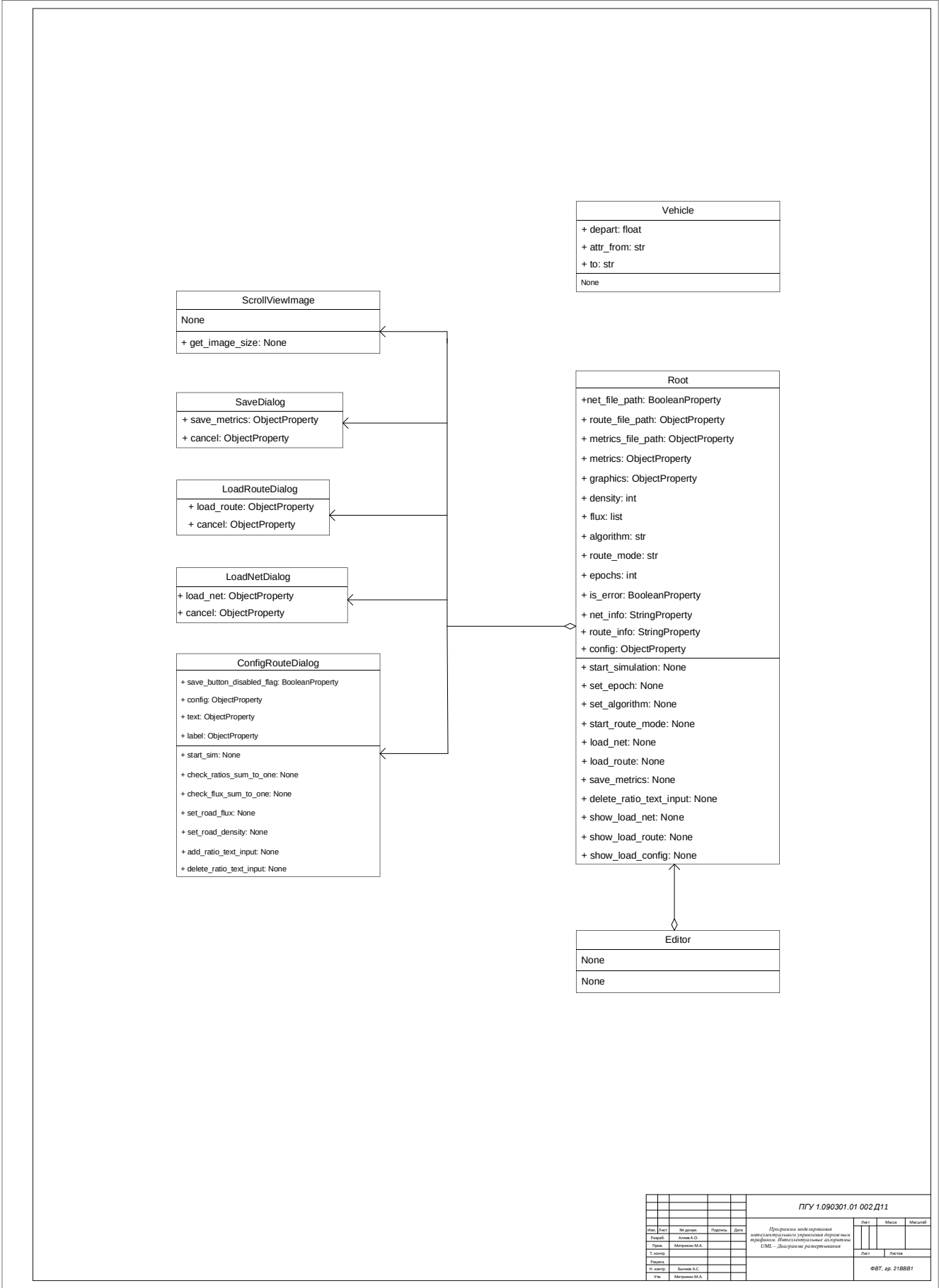
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.02.001 ПЗ

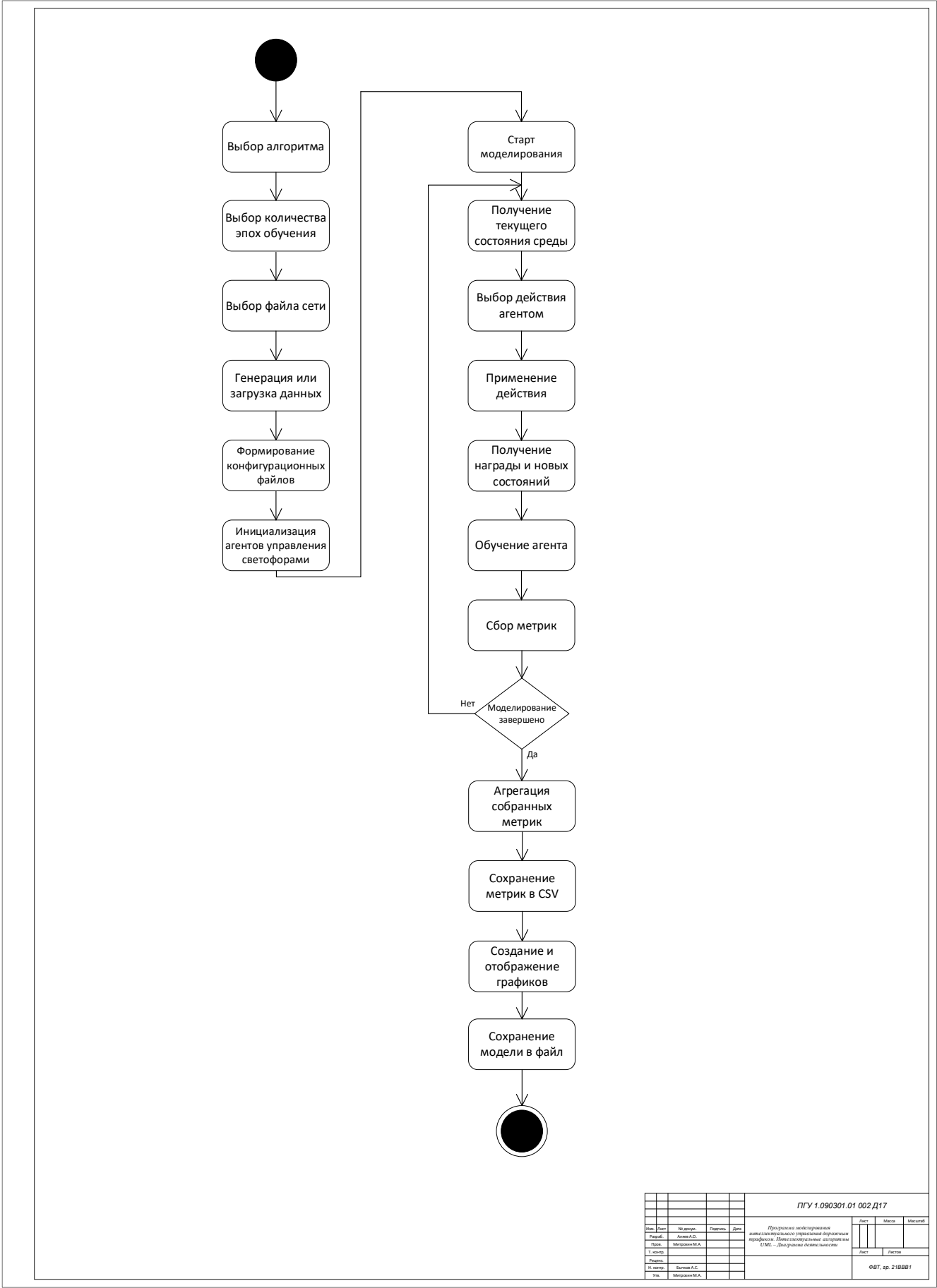
Лист

63

Приложение А.3 - Диаграмма классов



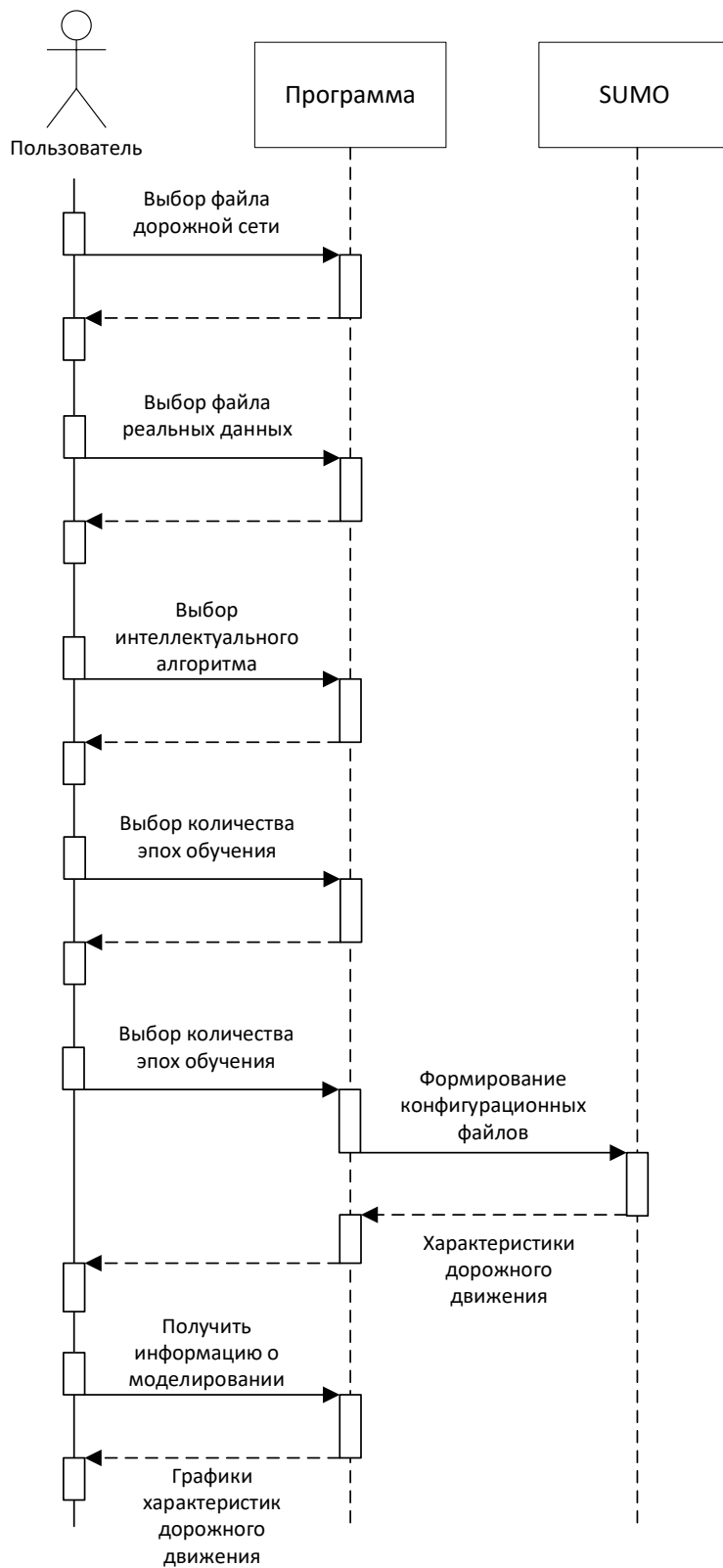
Приложение А.4 - Диаграмма деятельности



						ПГУ 1.090301.01 002 Д17		
Имя	Лист	№ докум.	Подпись	Дата	Программа моделирования	Лист	Масш.	Масштаб
Разраб.	Алекс А.О.				информационно-управляющие системы			
Проф.	Михайлов М.А.				Программное обеспечение			
Уч. центр					УИМ - информационные системы	Лист		Листов
Разраб.								
И. центр	Богачев А.С.					ФБТ, стр. 218881		
Уч. центр	Михайлов М.А.							

Изм.	Лист	№ докум.	Подпись	Дата

Приложение А.3 - Диаграмма последовательности



						ПГУ 1.090301.01 002 Д13		
						Программа модернизации интеллектуального управления дорожным движением. Тематическое направление УГМЭ – обеспечение безопасности использования		
Имя, Фамилия	№ документа	Подпись	Дата			Лист	Места	Местами
Рязанов, Алексей А.О.								
Попов, Митрофан М.А.								
Т. номер						Лист		Листов
Рязанов, Алексей А.О.								
И. номер	Булганов А.С.							
	Мухометов М.А.							
						ФБП, стр. 218ББ1		

Приложение Б

(обязательное)

Программа моделирования интеллектуального управления дорожным

трафиком

Листинг программы

					ПГУ 1.090301.02.001 ПЗ	Лист
						67
Изм.	Лист	№ докум.	Подпись	Дата		

Файл «main.py»

```

from kivy.app import App
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.floatlayout import FloatLayout
from kivy.factory import Factory
from kivy.properties import ObjectProperty
from kivy.properties import BooleanProperty
from kivy.properties import StringProperty
from kivy.uix.popup import Popup
from threading import Thread
from kivy.uix.image import Image
import sim_main
import os
import shutil
from kivy.core.window import Window
from kivy.uix.textinput import TextInput
from kivy.config import Config

Window.fullscreen = False

class LoadNetDialog(FloatLayout):
    load_net = ObjectProperty(None)
    cancel = ObjectProperty(None)

class ConfigRouteDialog(BoxLayout):
    save_button_disabled_flag = BooleanProperty(True)
    config = ObjectProperty(None)
    text = ObjectProperty(None)
    label = ObjectProperty(None)

    start_simulation = ObjectProperty(None)
    load_density = ObjectProperty(None)
    load_road_ratio = ObjectProperty(None)
    load_road_flux = ObjectProperty(None)

    cancel = ObjectProperty(None)

    def start_sim(self):
        self.start_simulation()
        self.cancel()

    def check_ratios_sum_to_one(self):
        return sum(self.config['ratio']) == 1

    def check_flux_sum_to_one(self):
        try:
            values = [float(x) if x.strip() else 0.0 for x in
self.config['flux']] # Конвертация во float
            return sum(values) == 1
        except Exception:
            return False # Если он ловит неконвертируемое

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.02.001 ПЗ

Лист

68

```

def set_road_flux(self):
    self.load_road_flux((self.ids.text_input_north_flux.text,
                        self.ids.text_input_east_flux.text,
                        self.ids.text_input_south_flux.text,
                        self.ids.text_input_west_flux.text,
                        ))

    self.save_button_disabled_flag = not
(self.check_ratios_sum_to_one() and self.check_flux_sum_to_one())

def set_density(self):
    self.load_density(self.ids.text_input_density.text)

def set_road_ratio(self, _, __):
    ratio = []
    for text_input in self.ids.ratio_box_layout.children:
        text = text_input.text
        if text.replace('.', '', 1).isdigit():
            ratio.append(float(text))
        else:
            ratio.append(0.0)

    ratio.reverse()
    self.config['ratio'] = ratio
    self.load_road_ratio(ratio)

    self.save_button_disabled_flag = not
(self.check_ratios_sum_to_one() and self.check_flux_sum_to_one())

def get_ratio(self, box_layout, text_input):
    for i in range(len(box_layout.children)):
        if text_input == box_layout.children[i]:
            return self.config['ratio'][i]

def add_ratio_text_input(self, box_layout):
    self.config['ratio'].append(0)
    text_input = TextInput(
        text=str(self.config['ratio'][len(self.config['ratio'])
1]),
        multiline="False",
    )
    text_input.bind(text=self.set_road_ratio)
    box_layout.add_widget(text_input)

def set_config_ratio(self, config):
    self.ids.ratio_box_layout.clear_widgets()
    for i in range(len(self.config['ratio'])):
        text_input = TextInput(
            text=str(config['ratio'][i]),
            multiline="False"
        )

```

```

text_input.bind(text=self.set_road_ratio)
                self.ids.ratio_box_layout.add_widget(text_input)
                self.save_button_disabled_flag = not
(self.check_ratios_sum_to_one() and self.check_flux_sum_to_one())
def delete_ratio_text_input(self, ratio_box_layout):
    ratio_box_layout.remove_widget(
        ratio_box_layout.children[0]
    )
    del self.config['ratio'][-1]
    self.load_road_ratio(self.config['ratio'])

    self.save_button_disabled_flag = not
(self.check_ratios_sum_to_one() and self.check_flux_sum_to_one())

class LoadRouteDialog(FloatLayout):
    load_route = ObjectProperty(None)
    cancel = ObjectProperty(None)

class SaveDialog(FloatLayout):
    save_metrics = ObjectProperty(None)
    cancel = ObjectProperty(None)
    file_name = ObjectProperty(None)
    algorithm = StringProperty(None)

    def save(self, save_metrics, file_name):
        if self.algorithm == "Q-learning":
            full_path = str(save_metrics + "/" + file_name + ".pkl")
            shutil.copy2('database/dataDWT.pkl', full_path)
        elif self.algorithm == "Neural Net":
            full_path = str(save_metrics + "/" + file_name + ".pth")
            shutil.copy2('database/dataQ.pth', full_path)
        self.cancel()

class Root(FloatLayout):
    net_file_path = ObjectProperty(None)
    route_file_path = ObjectProperty(None)
    metrics_file_path = ObjectProperty(None)
    metrics = ObjectProperty(None)
    graphics = ObjectProperty(None)
    density = 0 # Обращаться по self.density
    ratio = [1]
    flux = [0, 0, 0, 0]
    algorithm = "Q-learning"
    route_mode = "Flows"
    epochs = 1
    is_modeling = BooleanProperty(False) # для статусов
    is_error = BooleanProperty(False) # для ошибок
    net_info = StringProperty("Файл сети: ---") # для загрузки
информации о выбранном файле сети
    route_info = StringProperty("Файл маршрутов: ---") # для загрузки
информации о выбранном файле маршрутов

```

```

config = ObjectProperty({'density': density,
                        'ratio': ratio,
                        'flux': flux}) # Обращаться по
self.density`
sim_inf = ObjectProperty(None)

def set_epochs(self, text):
    try:
        value = int(text)
        if value > 0:
            self.epochs = value
            print(f"[INFO] Количество эпох установлено:
{self.epochs}")
        else:
            value = 1
    except Exception:
        return False

def set_route_mode(self, mode):
    self.route_mode = mode

def start_simulation(self):
    if self.route_mode == 'Flows':
        self.sim_inf = {
            'mode': self.route_mode,
            'algorithm': self.algorithm,
            'route_path': self.route_file_path,
            'net_path': self.net_file_path,
            'epochs': self.epochs
        }
    elif self.route_mode == 'generate':
        self.sim_inf = {
            'mode': self.route_mode,
            'algorithm': self.algorithm,
            'config': self.config,
            'net_path': self.net_file_path,
            'epochs': self.epochs
        }

def run():
    self.is_modeling = True
    self.graphics = []

    try:
        self.graphics = sim_main.sim_main(self.sim_inf)
    except Exception as e:
        print(f"[Ошибка]: {e}")
        self.is_error = True
        self.status_text = "Статус: Ошибка моделирования!"
        self.is_modeling = False
        raise

    if self.is_error:
        return

```

```

# только если всё прошло успешно:
    for img in os.listdir("graph/"):
        self.graphics.append("graph/" + str(img))

    self.is_modeling = False

    Thread(target=run, daemon=True).start()

def set_algorithm(self, value):
    self.algorithm = value
    print(f"Выбран алгоритм: {self.algorithm}")

def load_road_ratio(self, ratio):
    self.ratio = ratio
    self.config['ratio'] = ratio

def load_road_flux(self, flux):
    self.flux = flux
    self.config['flux'] = flux

def load_density(self, density):
    if str.isdecimal(density):
        self.density = density
        self.config['density'] = density
    else:
        self.density = 'Введите число'
        self.config['density'] = density

def dismiss_popup(self):
    self._popup.dismiss()

def show_load_net(self):
    content = LoadNetDialog(load_net=self.load_net,
cancel=self.dismiss_popup)
    self._popup = Popup(title="Load net file", content=content,
        size_hint=(0.9, 0.9))
    self._popup.open()

def show_load_route(self):
    content = LoadRouteDialog(load_route=self.load_route,
cancel=self.dismiss_popup)
    self._popup = Popup(title="Load route file", content=content,
        size_hint=(0.9, 0.9))
    self._popup.open()

def show_route_config(self):
    content = ConfigRouteDialog(config=self.config,
                                load_density=self.load_density,
                                load_road_ratio=self.load_road_ratio,
                                load_road_flux=self.load_road_flux,

start_simulation=self.start_simulation,
                                cancel=self.dismiss_popup, )
    content.set_config_ratio(self.config)

```

					ПГУ 1.090301.02.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		72

```

self._popup = Popup(title="Load route file", content=content,
                    size_hint=(0.9, 0.9))
self._popup.open()

def show_save(self):
    content = SaveDialog(save_metrics=self.save_metrics,
cancel=self.dismiss_popup, algorithm=self.algorithm)
    self._popup = Popup(title="Save file", content=content,
                    size_hint=(0.9, 0.9))
    self._popup.open()

def show_metrics(self, scrollViewBox):
    if self.graphics is not None:
        scrollViewBox.clear_widgets()
        scrollViewBox.height = 0
        for graph in self.graphics:
            img = ScrollViewImage(source=graph, height=500)
            scrollViewBox.add_widget(img)
            scrollViewBox.height += img.height - 20 # Space before
scroll WHY????
            # content = MetricsDialog(metrics = self.metrics, graphics =
self.graphics, cancel = self.dismiss_popup)
            # self._popup = Popup(title="Metrics and graphics",
content=content,
            # size_hint=(0.9, 0.9))
            # self._popup.open()

def change_label(self, root, text):
    root.ids.label.text = text

def load_net(self, path, filename):
    self.net_file_path = filename
    self.net_info = f"Файл сети: {os.path.basename(filename[0])}"
    self.dismiss_popup()

def load_route(self, path, filename):
    self.route_file_path = filename
    self.route_info = f"Файл маршрутов:
{os.path.basename(filename[0])}"
    self.dismiss_popup()

def save_metrics(self, path, filename):
    with open(os.path.join(path, filename), 'w') as stream:
        stream.write(self.text_input.text)

    self.dismiss_popup()

class ScrollViewImage(Image):
    def get_image_size(self):
        return 300, 300

class Editor(App):
    pass

```

```

Factory.register('Root', cls=Root)
Factory.register('LoadNetDialog', cls=LoadNetDialog)
Factory.register('LoadRouteDialog', cls=LoadRouteDialog)
Factory.register('ConfigRouteDialog', cls=ConfigRouteDialog)
Factory.register('SaveDialog', cls=SaveDialog)
# Factory.register('MetricsDialog', cls=MetricsDialog)

if __name__ == '__main__':
    Editor().run()

```

Файл «editor.kv»

```

Root:
    id: main_root
    orientation: 'vertical'
    padding: 10
    spacing: 10
    canvas.before:
        Color:
            rgba: 0.96, 0.96, 0.96, 1
        Rectangle:
            pos: self.pos
            size: self.size

    BoxLayout:
        id: left_menu
        orientation: 'horizontal'
        spacing: 15

        BoxLayout:
            orientation: 'vertical'
            size_hint_x: 0.4
            padding: 10
            spacing: 10
            canvas.before:
                Color:
                    rgba: 1, 1, 1, 1
                RoundedRectangle:
                    pos: self.pos
                    size: self.size
                    radius: [10]

            Label:
                text: "Действия"
                font_size: '20sp'
                color: 0.15, 0.15, 0.2, 1
                bold: True
                size_hint_y: None
                height: 40

            Button:
                text: 'Загрузить сеть'
                on_release: root.show_load_net()

```

```

Button:
    text: 'Загрузить маршрут'
    on_release: root.show_load_route()
Button:
    text: 'Начать моделирование \n с маршрутами из файла'
    halign: 'center'
    valign: 'center'
    on_release:
        root.set_route_mode('Flows')
        root.start_simulation()
Button:
    text: 'Начать моделирование \n с синтезированными \n данными'
    halign: 'center'
    valign: 'center'
    on_release:
        root.set_route_mode('generate')
        root.show_route_config()
Button:
    text: 'Показать графики'
    on_release: root.show_metrics(scrollViewBox)
Button:
    text: 'Сохранить результаты \n обучения'
    halign: 'center'
    valign: 'center'
    on_release: root.show_save()
BoxLayout:
    id: labels_layout
    orientation: 'vertical'
    padding: 10
    spacing: 10
    canvas.before:
        Color:
            rgba: 1, 1, 1, 1
        RoundedRectangle:
            pos: self.pos
            size: self.size
            radius: [10]

    Label:
        text: "Traffic Light Control System"
        font_size: '20sp'
        bold: True
        color: 0.15, 0.15, 0.2, 1
        size_hint_y: None
        height: "40dp"

    Label:
        id: status_label
        text: "Статус: Моделирование" if root.is_modeling else
("Статус: Ошибка моделирования!" if root.is_error else "Статус: Готов
к работе")
        color: (0.9, 0.54, 0.15, 1) if root.is_modeling else (0.93,
0.25, 0.13, 1) if root.is_error else (0.2, 0.8, 0.2, 1)

```

```
font_size: '16sp'
size_hint_y: None
height: "30dp"
```

```
AnchorLayout:
    anchor_x: 'center'
    anchor_y: 'center'
    size_hint_y: None
    height: "20dp"
```

```
Spinner:
    text: root.algorithm
    values: ["Q-learning", "Neural Net"]
    on_text: root.set_algorithm(self.text)
    size_hint: None, None
    size: "100dp", "20dp"
    background_normal: ''
    background_color: (0.95, 0.95, 0.95, 1)
    color: 0.1, 0.1, 0.1, 1
```

```
AnchorLayout:
    anchor_x: 'center'
    anchor_y: 'center'
    size_hint_y: None
    height: "30dp"
```

```
BoxLayout:
    orientation: 'horizontal'
    size_hint: None, None
    size: "150dp", "26dp"
    spacing: 10
```

```
Label:
    text: "Эпохи:"
    size_hint_x: None
    width: "60dp"
    font_size: '14sp'
    color: 0.2, 0.2, 0.2, 1
```

```
TextInput:
    id: epoch_input
    text: str(root.epochs)
    size_hint: None, None
    size: "60dp", "26dp"
    multiline: False
    input_filter: 'int'
    font_size: '14sp'
    on_text: root.set_epochs(self.text)
```

```
Label:
    id: info_label
    text: root.net_info
    font_size: '14sp'
    color: 0.2, 0.2, 0.2, 1
    size_hint_y: None
    height: "10dp"
```

```

Label:
    id: info_label
    text: root.route_info
    font_size: '14sp'
    color: 0.2, 0.2, 0.2, 1
    size_hint_y: None
    height: "10dp"

ScrollView:
    id: scrollView
    do_scroll: (True, True) #up and down
    size_hint: (1, 1)
    BoxLayout:
        orientation : 'vertical'
        size_hint_y : None
        id: scrollViewBox
        height: self.minimum_height
<ScrollViewImage>:
    height : 500
    source: 'images/notebook.png'
    size_hint_y: None

<PopupHeader@Label>:
    size_hint_y: None
    height: 35
    font_size: '16sp'
    bold: True
    color: 0.1, 0.1, 0.1, 1
    canvas.before:
        Color:
            rgba: 0.95, 0.95, 0.95, 1
        Rectangle:
            pos: self.pos
            size: self.size
<LoadNetDialog>:
    BoxLayout:
        orientation: "vertical"

        size: root.size
        pos: root.pos

        PopupHeader:
            text: "Загрузка файла сети"

        FileChooserIconView:
            id: filechooser
BoxLayout:
    size_hint_y: None
    height: 50
    spacing: 12
    Button:

```

```

text: "Отмена"
    on_release: root.cancel()
    background_color: 0.9, 0.3, 0.3, 1
    Button:
        text: "Загрузить"
        on_release: root.load_net(filechooser.path,
filechooser.selection)
        background_color: 0.3, 0.7, 0.3, 1

<LoadRouteDialog>:
    BoxLayout:
        orientation: "vertical"

        size: root.size
        pos: root.pos

        PopupHeader:
            text: "Загрузка файла маршрутов"
FileChooserIconView:
    id: filechooser

BoxLayout:
    size_hint_y: None
    height: 50
    spacing: 12
    Button:
        text: "Отмена"
        on_release: root.cancel()
        background_color: 0.9, 0.3, 0.3, 1
    Button:
        text: "Загрузить"
        on_release: root.load_route(filechooser.path,
filechooser.selection)
        background_color: 0.3, 0.7, 0.3, 1
<ConfigRouteDialog>:
    BoxLayout:
        orientation: 'vertical'
        size_hint: (0.8, 0.8)
        pos_hint: {'center_x': 0.5, 'center_y': 0.5}
        spacing: 10
        padding: 10

        GridLayout:
            cols: 2
            rows: 3
            spacing: 5
            size_hint_y: 0.8
Label:
    text: "Общее количество машин:"
    size_hint_x: 0.5
    halign: 'left'
    valign: 'middle'
    text_size: self.width, None

```

```

TextInput:
    id: text_input_density
    size_hint_x: 0.3
    height: 30
    text: str(root.config['density'])
    multiline: False
    input_filter: 'int'
    on_text: root.set_density()

Label:
    text: "Распределение по периодам:"
    size_hint_x: 0.5
    halign: 'left'
    valign: 'middle'
    text_size: self.width, None
BoxLayout:
    orientation: 'vertical'
    size_hint_x: 0.5
    spacing: 5

    BoxLayout:
        id: ratio_box_layout
        orientation: 'horizontal'
        spacing: 5
        size_hint_y: 0.7

    BoxLayout:
        orientation: 'horizontal'
        spacing: 5
        size_hint_y: 0.3
        Button:
            text: '+'
            size_hint_x: 0.5
            on_release: root.add_ratio_text_input(ratio_box_layout)

Button:
    id: remove_period_btn
    text: '-'
    size_hint_x: 0.5
    disabled: len(ratio_box_layout.children) <= 1
    on_release: root.delete_ratio_text_input(ratio_box_layout)

Label:
    text: "Соотношение направлений:"
    size_hint_x: 0.5
    halign: 'left'
    valign: 'middle'
    text_size: self.width, None
GridLayout:
    cols: 4
    rows: 2
    spacing: 5
    size_hint_x: 0.5
Label:
    text: "C"

```

```

Label:
    text: "B"
Label:
    text: "Ю"
Label:
    text: "3"

TextInput:
    id: text_input_north_flux
    size_hint_x: 0.25
    height: 30
    text: str(root.config['flux'][0])
    multiline: False
    input_filter: 'float'
    on_text: root.set_road_flux()
TextInput:
    id: text_input_east_flux
    size_hint_x: 0.25
    height: 30
    text: str(root.config['flux'][1])
    multiline: False
    input_filter: 'float'
    on_text: root.set_road_flux()
TextInput:
    id: text_input_south_flux
    size_hint_x: 0.25
    height: 30
    text: str(root.config['flux'][2])
    multiline: False
    input_filter: 'float'
    on_text: root.set_road_flux()
TextInput:
    id: text_input_west_flux
    size_hint_x: 0.25
    height: 30
    text: str(root.config['flux'][3])
    multiline: False
    input_filter: 'float'
    on_text: root.set_road_flux()

BoxLayout:
    orientation: 'horizontal'
    size_hint_y: 0.1
    spacing: 20
    padding: [0, 10, 0, 0]
Button:
    text: "Отмена"
    size_hint_x: 0.4
    on_release: root.cancel()
Button:
    id: save_button
    text: "Начать моделирование"

```

```

        size_hint_x: 0.4
        disabled: root.save_button_disabled_flag
        on_release: root.start_sim()

<SaveDialog>:
    text_input: text_input
    BoxLayout:
        size: root.size
        pos: root.pos
        orientation: "vertical"
        padding: 20
        spacing: 12
    PopupHeader:
        text: "Сохранение данных"

FileChooserIconView:
    id: filechooser
    on_selection: text_input.text = self.selection and
self.selection[0] or ''

TextInput:
    id: text_input
    size_hint_y: None
    height: 35
    multiline: False
    hint_text: "Имя файла для сохранения (без расширения)"
BoxLayout:
    size_hint_y: None
    height: 50
    spacing: 12
    Button:
        text: "Отмена"
        on_release: root.cancel()
        background_color: 0.9, 0.3, 0.3, 1
    Button:
        text: "Сохранить"
        on_release: root.save(filechooser.path, text_input.text)
        background_color: 0.3, 0.7, 0.3, 1

```

Файл «sim_main.py»

```

import time
import Q_Agent
import NN_Agent
import graph
import routeChange
import junction_config
from datetime import datetime, timedelta
import os
import traci as traci
import numpy as np
import csv

```

```

class Vehicle:
    vehicle_type = ""
    start = ""
    finish = ""
    event_time = 0
    start_road = ""
    finish_road = ""
    accumulating_time = 0.0

    def __init__(self, vehicle_type, start, finish, event_time,
start_road, finish_road, accumulating_time):
        self.vehicle_type = vehicle_type
        self.start = start
        self.finish = finish
        self.event_time = event_time
        self.start_road = start_road
        self.finish_road = finish_road
        self.accumulating_time = accumulating_time

def calculate_waiting_time():
    edges = ["A1B1", "B2B1", "C1B1", "B0B1"]
    waiting_list = []

    for edge in edges:
        vehicle_ids = traci.edge.getLastStepVehicleIDs(edge)
        if vehicle_ids:
            waiting_time = traci.edge.getWaitingTime(edge) /
len(vehicle_ids)
        else:
            waiting_time = 0
        waiting_list.append(waiting_time)

    return waiting_list

def count_cars_from_to_direct(dict_vehicle_list,
from_to_directions_dict, period, period_count):
    for junction in dict_vehicle_list:
        for junction1 in from_to_directions_dict.keys():
            if junction1 == junction:
                vehicle_list = dict_vehicle_list.get(junction)
                from_to_direction =
from_to_directions_dict.get(junction1)
                for vehicle in vehicle_list:
                    to_direction =
from_to_direction.get(vehicle.start)
                    time = int(vehicle.event_time // period)
                    if not to_direction.get(vehicle.finish):
                        to_direction[vehicle.finish] = [0 for _ in
range(period_count)]
                    to_direction[vehicle.finish][time] += 1

```

```

def run_rl_mode(period, epoch, initial_green, net_path, algorithm):
    branches_passed = 0
    filenames = []
    for attempt in range(epoch * 2):
        agent_is_on = (attempt % 2 == 0)
        if agent_is_on:
            if algorithm == "Q-learning":
                env_agent_on, initial_green = Q_Agent.sumoRl(period,
0.5, 0.6, initial_green, agent_is_on, net_path)
            elif algorithm == "Neural Net":
                env_agent_on, initial_green = NN_Agent.sumoRl(period,
0.5, 0.6, initial_green, agent_is_on, net_path)

            else:
                #env_agent_off, initial_green = NN_Agent.sumoRl(period,
0.5, 0.6, initial_green, agent_is_on, net_path)
                initial_green = {"B1": [45, 45]}
                env_agent_off, initial_green = Q_Agent.sumoRl(period, 0.5,
0.6, initial_green, agent_is_on, net_path)

        traci.close()
        branches_passed += 1

        if branches_passed == 2:
            linear_filenames = []
            filenames = graph.graph(env_agent_on, env_agent_off,
attempt // 2)
            for i in range(len(filenames)):
                for j in filenames[i]:
                    linear_filenames.append(j)
            env_agent_on = None
            env_agent_off = None
            branches_passed = 0

    return linear_filenames

def sim_main(sim_inf):
    period = 480
    mode = '1Junction'
    direct_count = 4
    data_type = "dl"

    try:
        names_list, directions_dict, edges_dict, junction_count =
junction_config.get_junctions_config(
            "junction_config.ini")
        if sim_inf['mode'] == "Flows":
            os.system(
                f"duarouter --flow-files={sim_inf['route_path'][0]} --
net-file=1Junction.net.xml "
                "--output-file=grid.rou.xml --begin 0 --end 100000000"
            )
        elif sim_inf['mode'] == 'generate':
            cars = int(sim_inf['config']['density'])

```

```

ratio = [float(x) for x in sim_inf['config']['flux']] # !!!
flux_list = sim_inf['config']['ratio'] # !!!
ways = routeChange.generate_ways()
routeChange.flux_to_route(ratio, flux_list, period, ways, cars)
os.system(
    "duarouter --flow-files=flows.xml --net-file=1Junction.net.xml "
    "--output-file=grid.rou.xml --begin 0 --end 100000000"
)
start = time.time()
epoch = 1
initial_green = {"B1": [45, 45]}
linear_filenames = run_rl_mode(period, sim_inf["epochs"],
initial_green, sim_inf['net_path'], sim_inf['algorithm'])

    end = time.time()
    print(str(end - start))
except Exception as e:
    print(f"[Ошибка]: {e}")
    raise

return linear_filenames

```

Файл «graph.py»

```

import matplotlib
from matplotlib import pyplot as plt
matplotlib.use('Agg') # Отключаем GUI для нормальной работы в другом
потоке

def graph(envOn, envOff, attempt):
    graph_filenames = []
    trafficLightAllTime(envOn.metrics, attempt)
    graph_filenames.append(eachDirectionsCars(envOn.metricsPeriod,
envOff.metricsPeriod, attempt))
    graph_filenames.append(carsWaitTimeInPeriod(envOn.metricsPeriod,
envOff.metricsPeriod, attempt))
    graph_filenames.append(carsHaltingInPeriod(envOn.metricsPeriod,
envOff.metricsPeriod, attempt))
    return graph_filenames
def trafficLightAllTime(metrics_on, attempt):
    colors = ['r', 'b']
    edges_labels = ['Север - Юг', 'Запад - Восток']
    for tl, color, edge in zip([tl for tl in metrics_on[0] if
"ts_time" in tl], colors, edges_labels):
        plt.plot([step["step"] / 60 for step in metrics_on[:]],
[tl["ts_time"][0] for tl in metrics_on[:]], color='r',
label=edges_labels[0])
        plt.plot([step["step"] / 60 for step in metrics_on[:]],
[tl["ts_time"][1] for tl in metrics_on[:]], color='b',
label=edges_labels[1])
    plt.legend(loc='center right')
    plt.ylabel('Trafficlight time (seconds)')
    plt.xlabel('Simulation time (minutes)')

```

```

filename = f'graph//TrafficlightAllTime_' + str(attempt) + '.png'
plt.savefig(filename, dpi=600)
plt.close()

def carsHaltingInPeriod(metricsPeriodOn, metricsPeriodOff, attempt):
    filenames = []
    edges_labels = ['Север', 'Восток', 'Юг', 'Запад']
    for i, (laneOn, laneOff, edge) in enumerate(
        zip([laneOn for laneOn in metricsPeriodOn[0] if
            "vehicleTotalHalting_lane " in laneOn], [laneOff for laneOff in
            metricsPeriodOff[0] if "vehicleTotalHalting_lane " in laneOff],
            edges_labels)):
        plt.plot([step["step"] / 60 for step in metricsPeriodOn[:]],
            [l[laneOn] for l in metricsPeriodOn[:]],
            label="С применением интеллектуального алгоритма",
            color="red")
        plt.plot([step["step"] / 60 for step in metricsPeriodOn[:]],
            [l[laneOn] for l in metricsPeriodOn[:]], 'o', color="red")

        plt.plot([step["step"] / 60 for step in metricsPeriodOff[:]],
            [l[laneOff] for l in metricsPeriodOff[:]],
            label="Без применения интеллектуального алгоритма",
            color="blue")
        plt.plot([step["step"] / 60 for step in metricsPeriodOff[:]],
            [l[laneOff] for l in metricsPeriodOff[:]], 'o',
            color="blue")
        plt.legend(loc='best')
        plt.title(f"Среднее количество ожидающих \n ТС на направлении
{edges_labels[i]}")
        plt.ylabel('Mean halting number (units)')
        plt.xlabel('Simulation time (minutes)')
        filename = f'graph//vehicleMeanHalting_' + str(attempt) +
str(edges_labels[i]) + '.png'
        plt.savefig(filename, dpi=600)
        plt.close()
        filenames.append(filename)
    return filenames

def eachDirectionsCars(metricsPeriodOn, metricsPeriodOff, attempt):
    filenames = []
    edges_labels = ['Север', 'Восток', 'Юг', 'Запад']
    for i, (laneOn, laneOff, edge) in enumerate(
        zip([laneOn for laneOn in metricsPeriodOn[0] if
            "vehiclesIDson_lane " in laneOn], [laneOff for laneOff in
            metricsPeriodOff[0] if "vehiclesIDson_lane " in laneOff],
            edges_labels)):
        plt.plot([step["step"] / 60 for step in metricsPeriodOn[:]],
            [l[laneOn] for l in metricsPeriodOn[:]],
            label="С применением интеллектуального алгоритма",
            color="red")
        plt.plot([step["step"] / 60 for step in metricsPeriodOn[:]],
            [l[laneOn] for l in metricsPeriodOn[:]], 'o', color="red")

```

```

plt.plot([step["step"] / 60 for step in metricsPeriodOff[:]],
[l[laneOff] for l in metricsPeriodOff[:]],
        label="Без применения интеллектуального алгоритма",
color="blue")
    plt.plot([step["step"] / 60 for step in metricsPeriodOff[:]],
[l[laneOff] for l in metricsPeriodOff[:]], 'o',
        color="blue")
    plt.title(f"Среднее количество проезжающих \n ТС на направлении
{edges_labels[i]}")
    plt.legend(loc='best')
    plt.ylabel('Mean number (units)')
    plt.xlabel('Time (minutes)')
    filename = f'graph//carsFromDirection_' + str(attempt) +
str(edges_labels[i]) + '.png'
    plt.savefig(filename, dpi=600)
    plt.close()
    filenames.append(filename)
return filenames

def carsWaitTimeInPeriod(metricsPeriodOn, metricsPeriodOff, attempt):
    filenames = []
    edges_labels = ['Север', 'Восток', 'Юг', 'Запад']

    for i, (laneOn, laneOff, edge) in enumerate(
        zip([laneOn for laneOn in metricsPeriodOn[0] if
"vehicleMeanWaitingTime_lane " in laneOn], [laneOff for laneOff in
metricsPeriodOff[0] if "vehicleMeanWaitingTime_lane " in laneOff],
edges_labels)):
        all_values_with_alg = []
        all_values_without_alg = []
        values_with_alg = [l[laneOn] for l in metricsPeriodOn[:]]
        values_without_alg = [l[laneOff] for l in metricsPeriodOff[:]]
        # Добавляем значения в общие списки
        all_values_with_alg.extend(values_with_alg)
        all_values_without_alg.extend(values_without_alg)
        plt.plot([step["step"] / 60 for step in metricsPeriodOn[:]],
[l[laneOn] for l in metricsPeriodOn[:]],
                label="С применением интеллектуального алгоритма",
color="red")
        plt.plot([step["step"] / 60 for step in metricsPeriodOn[:]],
[l[laneOn] for l in metricsPeriodOn[:]], 'o', color="red")
        plt.plot([step["step"] / 60 for step in metricsPeriodOff[:]],
[l[laneOff] for l in metricsPeriodOff[:]],
                label="Без применения интеллектуального алгоритма",
color="blue")
        plt.plot([step["step"] / 60 for step in metricsPeriodOff[:]],
[l[laneOff] for l in metricsPeriodOff[:]], 'o',
                color="blue")
    plt.legend(loc='best')
    plt.title(f"Среднее время ожидания \n ТС на направлении
{edges_labels[i]}")
    plt.ylabel('Mean waiting time (seconds)')
    plt.xlabel('Time (minutes)')

```

```

filename = f'graph//vehicleMeanWaitingTime_' + str(attempt) +
str(edges_labels[i]) + '.png'
plt.savefig(filename, dpi=600)
plt.close()
filenames.append(filename)
avg_with_alg = sum(all_values_with_alg) / len(all_values_with_alg)
if all_values_with_alg else 0
avg_without_alg = sum(all_values_without_alg) /
len(all_values_without_alg) if all_values_without_alg else 0

print(f"\nОбщее среднее время ожидания ТС для
{str(edges_labels[i])}:")
print(f"С алгоритмом: {avg_with_alg:.2f} секунд")
print(f"Без алгоритма: {avg_without_alg:.2f} секунд")
if avg_without_alg > 0:
improvement = ((avg_without_alg - avg_with_alg) /
avg_without_alg) * 100
print(f"Улучшение: {improvement:.2f}%")
return filenames

```

Файл «Q_agent.py»

```

from sumo_rl.agents import QAgent
from sumo_rl.environment.env import TLSCEnvironment
from sumo_rl.exploration.epsilon_greedy_q import EpsilonGreedyQ
import pickle
import os

def sumoRl(period, alpha, gamma, initial_green, agentIsOn, net_path):
    env = TLSCEnvironment(
        net_file=net_path[0],
        route_file="grid.rou.xml",
        out_csv_name="dataFiles/output.csv",
        use_gui=False,
        num_seconds=21000,
        min_green=10,
        max_green=80,
        initial_green=initial_green,
        period=period,
        reward_fn="queue",
        agentIsOn=agentIsOn
    )
    initial_states = env.reset()
    ql_agents = {
        ts: QAgent(
            starting_state=env.encode(initial_states[ts], ts),
            state_space=env.observation_space,
            action_space=env.action_space,
            alpha=alpha,
            gamma=gamma,
            exploration_strategy=EpsilonGreedyQ(
                initial_epsilon=0.3, min_epsilon=0.001, decay=0.001
            ),
        ),
    }

```

```

for ts in env.ts_ids
}
done = {"__all__": False}
out_csv = f"dataFiles/output.csv"
if os.path.exists('database//dataDWT.pkl'):
    with open('database//dataDWT.pkl', 'rb') as f:
        for agent_id in ql_agents.keys():
            ql_agents[agent_id].q_table |= pickle.load(f)
while not done["__all__"]:
    actions = {ts: ql_agents[ts].act() for ts in ql_agents.keys()}
    s, r, done, _ = env.step(action=actions)
    for agent_id in ql_agents.keys():
        ql_agents[agent_id].learn(next_state=env.encode(s[agent_id],
agent_id), reward=r[agent_id])
initial_green = env.save_tl_phases()
env.save_csv(out_csv, 1)
q_saving = ql_agents[agent_id].q_table
with open('database//dataDWT.pkl', 'wb') as f:
    pickle.dump(q_saving, f)
return env, initial_green

```

Файл «NN_agent.py»

```

from sumo_rl.agents import NNAgent
from sumo_rl.environment.env import TLSCEnvironment
from sumo_rl.exploration import EpsilonGreedyNN
import pickle
import torch
import os

def sumoRl(period, alpha, gamma, initial_green, agentIsOn, net_path):
    env = TLSCEnvironment(
        net_file=net_path[0],
        route_file="grid.rou.xml",
        out_csv_name="dataFiles//output.csv",
        use_gui=False,
        num_seconds=21000,
        min_green=10,
        max_green=80,
        initial_green=initial_green,
        period=period,
        reward_fn="diff-waiting-time",
        agentIsOn=agentIsOn,
    )
    initial_states = env.reset()
    lossList = []
    ql_agents = {
        ts: NNAgent(
            starting_state=env.encode(initial_states[ts], ts),
            action_space=env.action_space,
            alpha=alpha,
            gamma=gamma,

```

					ПГУ 1.090301.02.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		88

```

exploration_strategy=EpsilonGreedyNN(
    initial_epsilon=0.3, min_epsilon=0.01, decay=0.01
),
    lossList=lossList
)
    for ts in env.ts_ids
}
done = {"__all__": False}
out_csv = f"dataFiles/output.csv"
if os.path.exists('database//dataQ.pth'):
    for agent_id in ql_agents.keys():

ql_agents[agent_id].q_network.load_state_dict(torch.load('database//da
taQ.pth'))
while not done["__all__"]:
    actions = {ts: ql_agents[ts].act() for ts in ql_agents.keys()}
    s, r, done, _ = env.step(action=actions)
    for agent_id in ql_agents.keys():
        ql_agents[agent_id].learn(ql_agents[agent_id].state,
actions[agent_id],
                                next_state=env.encode(s[agent_id],
agent_id), reward=r[agent_id])
initial_green = env.save_tl_phases()
env.save_csv(out_csv, 1)
for agent_id in ql_agents.keys():
    torch.save(ql_agents[agent_id].q_network.state_dict(),
'database//dataQ.pth')
    params = ql_agents[agent_id].q_network.state_dict()
weights = []
for param_tensor in ql_agents[agent_id].q_network.state_dict():

weights.append(ql_agents[agent_id].q_network.state_dict()[param_tensor
].detach().numpy())
return env, initial_green

```

Файл «route_change.py»

```

import xml.etree.cElementTree as ET
import random
import numpy as np
import sumolib.route
import math
import os

class V:
    depart = 0
    attr_from = ''
    to = ''

    def __init__(self, depart, attr_from, to):
        self.depart = depart
        self.attr_from = attr_from
        self.to = to

```

					ПГУ 1.090301.02.001 ПЗ	Лист
						89
Изм.	Лист	№ докум.	Подпись	Дата		

```

def change_time_in_period(car_count_list, period, dict_vehicle_list):
    total_cars = sum(car_count_list)
    for vehicle_list in dict_vehicle_list.values():
        time = 0
        j = 0
        for num_cars in car_count_list:
            for y in range(num_cars):
                a = np.random.exponential(scale=1 / 2.1859)
                time_interval = period / num_cars * a
                vehicle_list[j].event_time = time + y + time_interval
                if vehicle_list[j].event_time < 0:
                    vehicle_list[j].event_time = 0
                j += 1
            time += period

def remove_cars_in_junction(dict_vehicle_list):
    incoming_routes = []
    exiting_routes = []
    exiting_routes_set = set()
    temp = sumolib.net.readNet('3Junction.net.xml')
    routes = temp.getEdges()

    for route in routes:
        if len(route.getIncoming()) <= 1 and len(route.getOutgoing())
> 0:
            incoming_routes.append(route.getID()[2])

    exiting_routes = list(exiting_routes_set)
    new_dict_vehicle_list = dict(dict_vehicle_list) # Create a new
dictionary for the result

    for key, vehicles in dict_vehicle_list.items():
        new_vehicles = []
        for car in vehicles:
            if car.start in incoming_routes:
                new_vehicles.append(car)
        new_dict_vehicle_list[key] = new_vehicles

    return new_dict_vehicle_list

def reroute_cars(dict_vehicle_list, flux_function):

    def continue_routes(car, junction_list, flux_function,
incoming_junction, outgoing_junction):
        if incoming_junction in junction_list:
            flux_subdict =
flux_function.get(incoming_junction).get(outgoing_junction)
            total_sum = 0
            random_number = round(random.uniform(0, 1), 8)
            outgoing_junction = incoming_junction
            for incoming_junction, value in flux_subdict.items():
                if random_number > total_sum + value:
                    total_sum += value

```

```

else:
    return continue_routes(car, junction_list,
flux_function, incoming_junction, outgoing_junction)
    else:
        return outgoing_junction + incoming_junction

junction_list = list(dict_vehicle_list.keys())
for junction in junction_list:
    for key1, vehicles in dict_vehicle_list.items():
        for car in vehicles:
            car.start_road = car.start + key1
            car.finish_road = continue_routes(car, junction_list,
flux_function, key1, car.start)

def set_parameters(dict_vehicle_list):
    tree = ET.parse('flows.xml')
    j = 0
    i = 0
    junctionList = list(dict_vehicle_list.keys())
    for i in range(len(junctionList)):
        for key1, vehicles in dict_vehicle_list.items():
            for car in vehicles:
                car.startRoad = car.start + key1
                car.finishRoad = key1 + car.finish
    junction = list(dict_vehicle_list.keys())
    cars = list(dict_vehicle_list.values())
    for trip in tree.getroot():
        start = cars[j][i].startRoad
        finish = cars[j][i].finishRoad
        eventTime = cars[j][i].eventTime
        trip.set('from', start)
        trip.set('to', finish)
        trip.set('depart', str(eventTime))
        if i < len(cars[j]) - 1:
            i += 1
        else:
            i = 0
            j += 1
        start = ''
        finish = ''
        eventTime = 0
    root = tree.getroot()
    sorted_root = ET.Element(root.tag)
    sorted_elements = sorted(root, key=lambda child:
float(child.get('depart')))
    for element in sorted_elements:
        sorted_root.append(element)
    tree = ET.ElementTree(sorted_root)
    tree.write('flows.xml')

```

```

def generate_ways():
    ways_move = []
    incoming_routes = []
    exiting_routes = []
    temp = sumolib.net.readNet('1Junction.net.xml')
    routes = temp.getEdges()

    for route in routes:
        if len(route.getIncoming()) > 1 and len(route.getOutgoing())
== 0:
            exiting_routes.append(route.getID())
        else:
            incoming_routes.append(route.getID())

    ways_move.extend([[x, exiting_routes] for x in incoming_routes])
    return ways_move

def flux_to_route(ratio, flux_list, period, ways, cars):
    i = 0
    offset = 0
    error = 0
    set_list = [] * cars
    incoming_routes = 0

    for way in ways:
        ratio_flux_list = [
            cars / len(ways) * ratio[incoming_routes] * flux_list[i]
for i in range(len(flux_list))
        ]

        for exiting_routes in way[1]:
            while i < len(ratio_flux_list):
                if offset >= math.floor(ratio_flux_list[i]):
                    error += ratio_flux_list[i] - offset
                    if error >= 1:
                        v = V(period * i + period / ratio_flux_list[i]
* offset, way[0], exiting_routes)
                        set_list.append(v)
                        error -= 1
                    i += 1
                    offset = 0
                    continue
                v = V(period * i + period / ratio_flux_list[i] *
offset, way[0], exiting_routes)
                set_list.append(v)
                offset += 1

            i = 0
            incoming_routes += 1

    set_list.sort(key=lambda x: x.depart)
    os.system(f"randomTrips.py -n 1Junction.net.xml -o flows.xml -e
{len(set_list)}")

```

```

i = 0
tree = ET.parse('flows.xml')

for trip in tree.getroot():
    trip.set('depart', str(set_list[i].depart))
    trip.set('from', set_list[i].attr_from)
    trip.set('to', set_list[i].to)
    i += 1

tree.write("flows.xml")

```

Файл «env.py»

```

class TLSCEnvironment(SumoEnvironment):
    def __init__(
        self,
        net_file: str,
        route_file: str,
        initial_green: dict,
        out_csv_name: Optional[str] = None,
        use_gui: bool = False,
        virtual_display: Tuple[int, int] = (3200, 1800),
        begin_time: int = 0,
        num_seconds: int = 4500,
        max_depart_delay: int = -1,
        waiting_time_memory: int = 1000,
        time_to_teleport: int = -1,
        delta_time: int = 5,
        yellow_time: int = 2,
        min_green: int = 10,
        max_green: int = 50,
        single_agent: bool = False,
        reward_fn: Union[str, Callable, dict] = "diff-waiting-
time",
        observation_class: ObservationFunction =
DefaultObservationFunction,
        add_system_info: bool = True,
        add_per_agent_info: bool = True,
        sumo_seed: Union[str, int] = "random",
        fixed_ts: bool = False,
        sumo_warnings: bool = True,
        additional_sumo_cmd: Optional[str] = None,
        render_mode: Optional[str] = None,
        period: int = 480,
        vehicleDictionary: dict = {},
        agentIsOn: bool = True,
    ) -> None:
        super(TLSCEnvironment, self).__init__(net_file,
                                                route_file,
                                                initial_green,
                                                out_csv_name,
                                                use_gui,
                                                virtual_display,
                                                begin_time,
                                                num_seconds,

```

					ПГУ 1.090301.02.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		93

```

max_depart_delay,
waiting_time_memory,
time_to_teleport,
delta_time,
yellow_time,
min_green,
max_green,
single_agent,
reward_fn,
observation_class,
add_system_info,
add_per_agent_info,
sumo_seed,
fixed_ts,
sumo_warnings,
additional_sumo_cmd,
render_mode,
agentIsOn)

self.period = period
self.agentIsOn = agentIsOn
self.vehicleDictionary = vehicleDictionary
self.metricsPeriod = []

def _get_period_metrics(self):
    last_period = 0
    for ts_id in self.ts_ids:
        buff = dict(
            {"step": 0, ts_id + "_stopped": 0, ts_id +
            "_accumulated_waiting_time": 0, ts_id + "_average_speed": 0})
        buff.update({"vehiclesIDson_lane " + lane[:-2]: set() for lane
in self.traffic_signals[ts_id].lanes})
        buff.update({"ts_time": [0, 0]})
        buff.update(
            {"vehicleMeanWaitingTime_lane " + lane[:-2]: float() for
lane in self.traffic_signals[ts_id].lanes})
        buff.update(
            {"vehicleTotalHalting_lane " + lane[:-2]: float() for lane
in self.traffic_signals[ts_id].lanes})
        for i in range(len(self.metrics)):
            if int(self.metrics[i].get("step")) <= last_period +
int(self.period):
                buff |= {ts_id + "_stopped":
                    float(self.metrics[i].get(ts_id +
                    "_stopped")) + float(buff.get(ts_id + "_stopped"))}
                buff |= {ts_id + "_accumulated_waiting_time":
float(self.metrics[i].get(ts_id + "_accumulated_waiting_time")) +
float(buff.get(
                    ts_id + "_accumulated_waiting_time"))}
        buff |= {ts_id + "_average_speed":
            float(self.metrics[i].get(ts_id + "_average_speed")) +
float(
                buff.get(ts_id + "_average_speed"))}

```

					ПГУ 1.090301.02.001 ПЗ	Лист
						94
Изм.	Лист	№ докум.	Подпись	Дата		

```

for lane in self.traffic_signals[ts_id].lanes:
    buff |= {"vehiclesIDson_lane " + lane[:-2]: set(
        self.metrics[i].get("vehiclesIDson_lane " + lane[:-2], [])) |
    set(
        buff.get("vehiclesIDson_lane " + lane[:-2], []))}
    if len(self.metrics[i].get("vehiclesIDson_lane " + lane[:-2])) >
0:
        buff |= {"vehicleMeanWaitingTime_lane " + lane[:-2]:
            float(buff.get("vehicleMeanWaitingTime_lane " +
lane[:-2])) + self.metrics[
                i].get(
                    "vehicles_accumulated_waiting_time_on_lane "
+ lane[:-2])}
        buff |= {"vehicleTotalHalting_lane " + lane[:-2]:
            float(buff.get("vehicleTotalHalting_lane " + lane[:-
2])) + self.metrics[i].get(
                "vehicles_halting_on_lane " + lane[:-2])}
buff |= {"ts_time": [x + y for x, y in
zip(self.metrics[i].get("ts_time"), buff.get("ts_time"))]}
else:
    buff |= {"step": self.metrics[i].get("step")}
    buff |= {
        ts_id + "_stopped": float(buff.get(ts_id + "_stopped")) /
float(self.period / self.delta_time)}
    buff |= {ts_id + "_accumulated_waiting_time":
        float(buff.get(ts_id + "_accumulated_waiting_time"))
/ float(
            self.period / self.delta_time)}
    buff |= {ts_id + "_average_speed": float(buff.get(ts_id +
"_average_speed"))}
    buff |= {"vehiclesIDson_lane " + lane[:-2]:
len(buff.get("vehiclesIDson_lane " + lane[:-2], [])) for
        lane in self.traffic_signals[ts_id].lanes}
    buff |= {"vehicleTotalHalting_lane " + lane[:-2]: float(
        buff.get("vehicleTotalHalting_lane " + lane[:-2]))
/
float(self.period / self.delta_time) for
        lane in self.traffic_signals[ts_id].lanes}
buff |= {"ts_time": [time / self.period for time in
buff.get("ts_time")]}
buff |= {"vehicleMeanWaitingTime_lane " + lane[:-2]: float(
    buff.get("vehicleMeanWaitingTime_lane " + lane[:-2])) /
float(self.period / self.delta_time) for
        lane in self.traffic_signals[ts_id].lanes}
self.metricsPeriod.append(buff.copy())
buff = dict(
    {"step": 0, ts_id + "_stopped": 0, ts_id +
"_accumulated_waiting_time": 0,
        ts_id + "_average_speed": 0})
buff.update(
    {"vehiclesIDson_lane " + lane[:-2]: set() for lane in
self.traffic_signals[ts_id].lanes})
buff.update({"ts_time": [0, 0]})

```

```

buff.update(
    {"vehicleMeanWaitingTime_lane " + lane[:-2]:
float() for lane in
    self.traffic_signals[ts_id].lanes})
buff.update(
    {"vehicleTotalHalting_lane " + lane[:-2]: float()
for lane in
    self.traffic_signals[ts_id].lanes})
last_period += self.period

def save_csv(self, out_csv_name, episode):
    """Save metrics of the simulation to a .csv file.

    Args:
        out_csv_name (str): Path to the output .csv file. E.g.:
        "results/my_results
        episode (int): Episode number to be appended to the output
        file name.
    """
    self._get_period_metrics()
    super(TLSCEnvironment, self).save_csv(out_csv_name, episode)
    if out_csv_name is not None:
        df = pd.DataFrame(self.metricsPeriod)
        Path(Path(out_csv_name).parent).mkdir(parents=True,
exist_ok=True)
        df.to_csv(out_csv_name + f"_conn{self.label}_ep{episode}" +
f"_period{self.period}" + ".csv", index=False)

        for id in self.ts_ids:
            Path(Path(out_csv_name).parent).mkdir(parents=True,
exist_ok=True)
            df.to_csv(out_csv_name + f"_conn{self.label}_ep{episode}"
+ f"_ts{id}" + ".csv",
                    index=False)

```

Файл «nn_agent.py»

```

import torch.optim as optim
import numpy as np
from sumo_rl.exploration.epsilon_greedy_nn import EpsilonGreedyNN

class QNetwork(nn.Module):
    def __init__(self, state_size, action_size):
        super(QNetwork, self).__init__()
        self.fc1 = nn.Linear(state_size, 64)
        self.fc2 = nn.Linear(64, 64)
        self.fc3 = nn.Linear(64, action_size)
    def forward(self, x):
        x = torch.relu(self.fc1(x))
        x = torch.relu(self.fc2(x))
        x = self.fc3(x)
        return x

```

```

class NNAgent:
    """Q-learning Agent class."""

    def __init__(self, starting_state, action_space, alpha=0.5,
gamma=0.01, exploration_strategy=EpsilonGreedyNN(),
        lossList=[]):
        """Initialize Q-learning agent."""
        self.state = starting_state
        self.state_size = len(self.state)
        self.action_space = action_space
        self.action_size = self.action_space.n
        self.alpha = alpha
        self.gamma = gamma
        self.exploration = exploration_strategy
        self.q_network = QNetwork(int(self.state_size),
int(self.action_size))
        self.optimizer = optim.Adam(self.q_network.parameters(),
lr=alpha)
        self.acc_reward = 0
        self.device = torch.device("cuda" if torch.cuda.is_available()
else "cpu")
        self.lossList = lossList
    def act(self):
        self.state = torch.tensor(self.state,
dtype=torch.float32).to(self.device)
        q_values = self.q_network(self.state)
        action = self.exploration.choose(q_values, self.action_space)
        return action

    def learn(self, state, action, next_state, reward, done=False):
        state = torch.tensor(state, dtype=torch.float32).to(self.device)
        next_state = torch.tensor(next_state,
dtype=torch.float32).to(self.device)
        q_values = self.q_network(state)
        q_values_next = self.q_network(next_state)
        q_value = q_values[action]
        if done:
            target = reward
        else:
            target = reward + self.gamma * torch.max(q_values_next)
        loss = nn.MSELoss()(q_value, target)
        self.optimizer.zero_grad()
        self.lossList.append(int(loss))
        loss.backward()
        self.optimizer.step()
        self.acc_reward += reward

```

Файл «ql_agent.py»

```

from sumo_rl.exploration.epsilon_greedy_q import EpsilonGreedyQ
class QLAgent:
    def __init__(self, starting_state, state_space, action_space,
alpha=0.5, gamma=0.95, exploration_strategy=EpsilonGreedyQ()):

```

```

self.state = starting_state
self.state_space = state_space
self.action_space = action_space
self.action = None
self.alpha = alpha
self.gamma = gamma
self.q_table = {self.state: [0 for _ in range(action_space.n)]}
self.exploration = exploration_strategy
self.acc_reward = 0

def act(self):
    """Choose action based on Q-table."""
    self.action = self.exploration.choose(self.q_table, self.state,
self.action_space)
    return self.action

def learn(self, next_state, reward, done=False):
    """Update Q-table with new experience."""
    if next_state not in self.q_table:
        self.q_table[next_state] = [0 for _ in
range(self.action_space.n)]
    if next_state in self.q_table:
        a = 1
        s = self.state
        s1 = next_state
        a = self.action
        delta_time = self.alpha * (
            reward + self.gamma * max(self.q_table[s1]) -
self.q_table[s][a]
        )
        self.q_table[s][a] = self.q_table[s][a] + self.alpha * (
            reward + self.gamma * max(self.q_table[s1]) -
self.q_table[s][a]
        )
        self.state = s1
        self.acc_reward += reward

```

Файл «epsilon_greedy_nn.py»

```

import numpy as np

class EpsilonGreedyNN:
    """Epsilon Greedy Exploration Strategy."""

    def __init__(self, initial_epsilon=1.0, min_epsilon=0.0,
decay=0.99):
        """Initialize Epsilon Greedy Exploration Strategy."""
        self.initial_epsilon = initial_epsilon
        self.epsilon = initial_epsilon
        self.min_epsilon = min_epsilon
        self.decay = decay

```

```

def choose(self, q_values, action_space):
    """Choose action based on epsilon greedy strategy."""
    if np.random.rand() < self.epsilon:
        action = int(action_space.sample())
    else:
        action = np.argmax(q_values.cpu().detach().numpy())

    self.epsilon = max(self.epsilon * self.decay, self.min_epsilon)
    # print(self.epsilon)
    return action

def reset(self):
    """Reset epsilon to initial value."""
    self.epsilon = self.initial_epsilon

```

Файл «epsilon_greedy_q.py»

```

import numpy as np

class EpsilonGreedyQ:

    def __init__(self, initial_epsilon=1.0, min_epsilon=0.0,
decay=0.99):
        self.initial_epsilon = initial_epsilon
        self.epsilon = initial_epsilon
        self.min_epsilon = min_epsilon
        self.decay = decay

    def choose(self, q_table, state, action_space):
        if np.random.rand() < self.epsilon:
            action = int(action_space.sample())
        else:
            action = np.argmax(q_table[state])

        self.epsilon = max(self.epsilon * self.decay,
self.min_epsilon)
        # print(self.epsilon)
        return action

    def reset(self):
        """Reset epsilon to initial value."""
        self.epsilon = self.initial_epsilon

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.02.001 ПЗ

Лист

99

Приложение В

(Обязательное)

Программа моделирования интеллектуального управления дорожным

трафиком

Презентация

					ПГУ 1.090301.02.001 ПЗ	Лист
						100
Изм.	Лист	№ докум.	Подпись	Дата		

Бизнес-проект «Программа моделирования интеллектуального управления дорожным трафиком»

Докладчики:
студенты группы 218ВВ81
Аляев Александр
Лобанов Рамиль

1

Цель и задачи работы

Цель работы – разработать программу и пользовательский интерфейс для неё, реализующую интеллектуальные методы управления светофорными объектами на перекрестке, а также визуализирующую их эффективность при помощи графиков зависимостей различных характеристик дорожного движения.

Задачи работы:

- проанализировать существующие среды имитационного моделирования, фреймворки и библиотеки для создания GUI и алгоритмы интеллектуального управления транспортными потоками;
- разработать имитационную модель дорожного трафика и пользовательский интерфейс для неё, основанную на агентном подходе;
- разработать алгоритм для генерации данных о трафике на основе информации от пользователя;
- разработать интеллектуальные алгоритмы управления светофорным объектом и интегрировать их в имитационную модель;
- провести сравнительный анализ эффективности интеллектуальных алгоритмов в управлении светофорами по сравнению со статическими методами.

3

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.02.001 ПЗ

Лист

101

Аналоги

Программное обеспечение	Разработчик	Достоинства	Недостатки
AVEDEX	MALLENOM TECHNOLOGIES	Учёт нескольких типов участников дорожного движения Интеграция с другими источниками Высокоточная видеоаналитика.	Невозможность самообучения модели Централизованное управление данными с перекрёстков.
Авангард	НПО «РБС»	Сбор данных с разных источников Анализ аварийности Виртуальные модели дорожной сети в реальном времени.	Требует нескольких типов датчиков Отсутствие машинного обучения.
Sentinel Smart Traffic	Complitech	Мониторинг нарушений ПДД Оперативная реакция на ДТП Распознавание номерных знаков.	Невозможность обучения модели Высокая стоимость внедрения.
Умный перекрёсток	СпецДор Проект	Приоритет для общественного транспорта Интеграция с городской ИТС Снижение времени ожидания на перекрёстках.	Высокая стоимость внедрения.



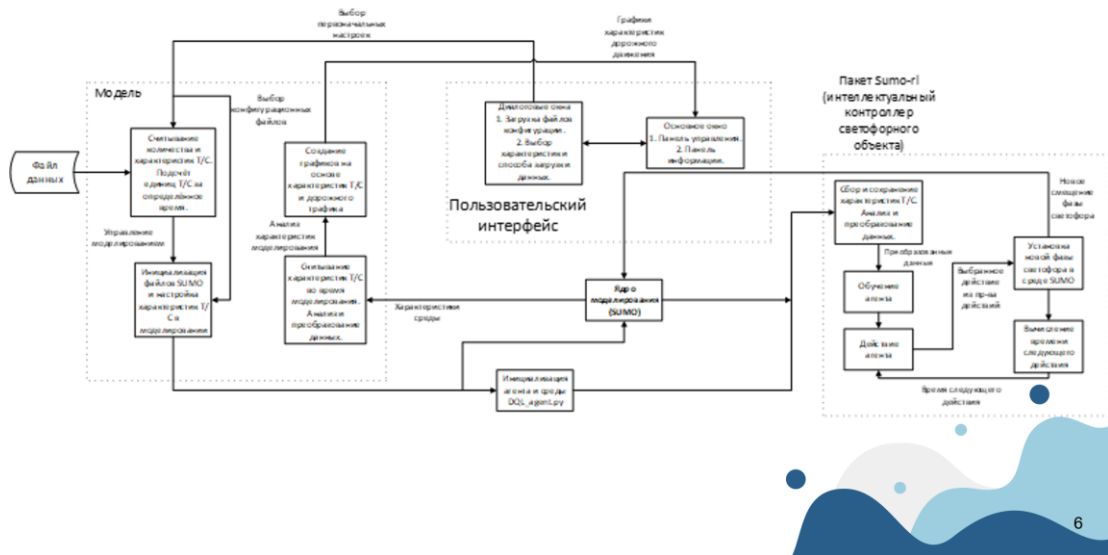
4

Стек технологий



5

Структура программы



Интеллектуальные методы управления

Агентный подход

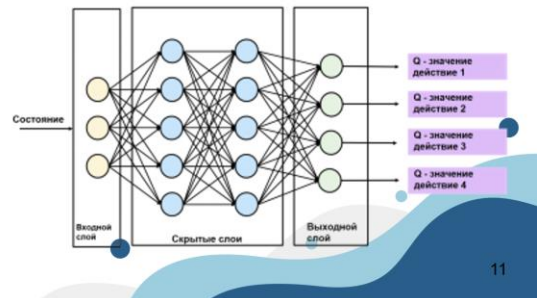


Табличное Q – обучение

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha * (r_t + \gamma * \max_{A'} Q(S_{t+1}, A') - Q(S_t, A_t))$$

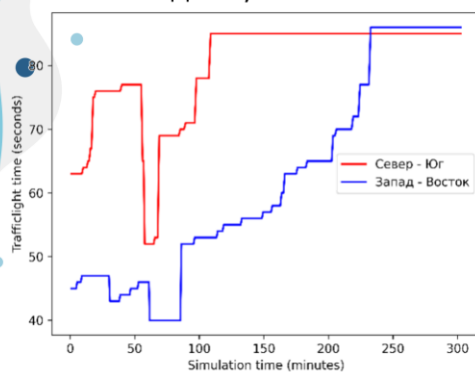
	A ₁	A ₂	A ₃	A ₄	...
S ₁	+1	+2	-1	0	...
S ₂	+2	0	+1	-2	...
S ₃	-1	+1	0	-2	...
S ₄	-2	0	+1	+1	...
...	...	...	...	...	...

Глубокое Q – обучение

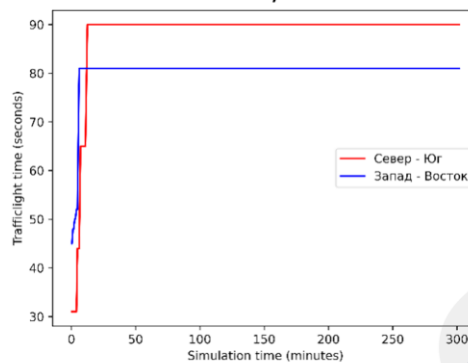


Обучение агента

До обучения



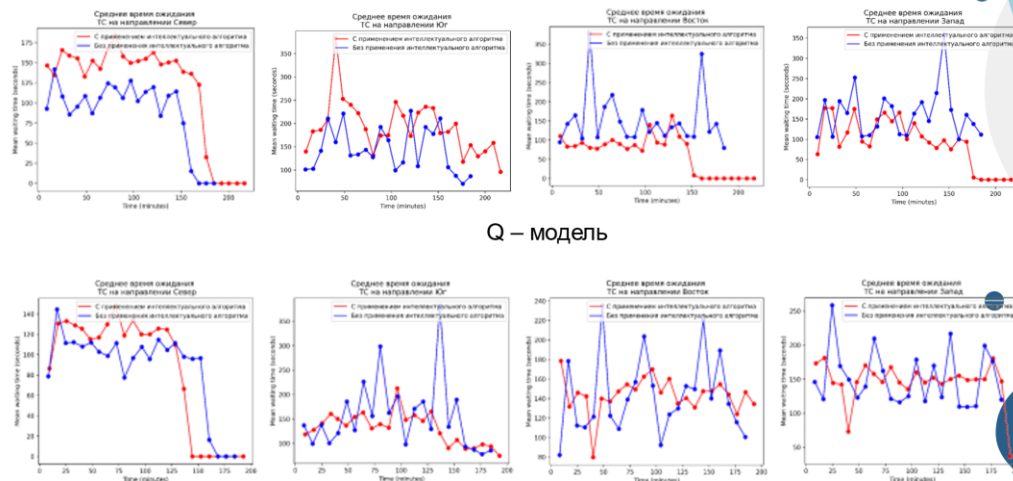
После обучения



- 1000 эпох обучения
- Гиперпараметры: $\alpha = 0.5$, $\beta = 0.6$, $\text{decay} = 0.001$, $\text{start epsilon} = 1$

12

Результаты обучения алгоритмов



Нейронная сеть

13

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.02.001 ПЗ

Лист

104

Сравнение эффективности

Направление	Среднее время ожидания для обучения с подкреплением секунд	Среднее время ожидания без обучения с подкреплением, секунд	Эффективность
Север	136.4	88.5	- 54 %
Юг	193.7	137.7	- 40 %
Восток	48.8	145.4	66 %
Запад	55.45	131.5	57 %

Эффективность Q – обучения

Направление	Среднее время ожидания при использовании нейронной сети секунд	Среднее время ожидания без использования нейронной сети, секунд	Эффективность
Север	84.7	86.7	2 %
Восток	143.7	142.2	- 1 %
Юг	131.1	155.6	15 %
Запад	145.6	150.8	3 %

Эффективность нейронной сети

14

Целевая аудитория и каналы взаимодействия

Аудитория :

- Муниципальные транспортные службы – заинтересованы в улучшении пропускной способности транспортной системы
- Разработчики ИТС – используют алгоритмы в составе более сложных продуктов
- Исследовательские группы – занимаются анализом и улучшением транспортной инфраструктуры

Каналы взаимодействия :

- Участие в профильных выставках и конференциях
- Сотрудничество с органами власти
- Публикации в научных журналах
- Прямые встречи с заинтересованными лицами

16

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.02.001 ПЗ

Лист

105

Перспективы коммерциализации

Инвестиции в область: с 2019 г. инвестиции в ИТС в

- России превысили 100 млрд Р. В ходе национального проекта «Безопасные качественные дороги» в 2024 г. было выделено 3,15 млрд Р на внедрение ИТС в 62 городах в 56 субъектах России.

Рыночная потребность: по прогнозам к 2027 – 2029 гг.

рынок российских ИТС будет расти ежегодно на 8 – 10 % за счёт инвестиций в ИТС проекты.

Простота интеграции : система может быть внедрена на

основе существующей городской инфраструктуры без необходимости капитальных изменений .



17

Финансовые показатели

 **Финансовые показатели стартапа**

- Начальный бюджет: 300 000 Р собственных средств
- Первый контракт: 300 000 Р
- Прогноз доходов (12 мес.): 1 000 000 Р
- Ожидаемая прибыль: +430 000 Р

 **Точка безубыточности**

- Достигается после 2-3 контрактов
- Критический уровень расходов: 570 000 Р
- Средний доход с одного контракта: 250 000 – 300 000 Р

 **Зона устойчивого роста**

- ROE: 75,4%
- Чистый оборотный капитал: 100 000 Р
- Потребность в дофинансировании : 0 – 500 000 Р (для масштабирования)

 **Доходы:**

- Контракты (3-4 в год) → 1 000 000 Р

 **Расходы:**

- Разработка ПО → 250 000 Р
- Маркетинг → 100 000 Р
- Прочие → 220 000 Р

 **Чистая прибыль: +430 000 Р**

Итог:

-  При текущем темпе – прибыль на 11 -м месяце
-  Без инвестиций рост 75,4% в год
-  При инвестициях – масштабирование в 2 раза быстрее

18

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.02.001 ПЗ

Лист

106



Спасибо за внимание!



					ПГУ 1.090301.02.001 ПЗ	Лист
						107
Изм.	Лист	№ докум.	Подпись	Дата		