МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
МЕХАНИКО-МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ
Кафедра дифференциальных уравнений и системного анализа

ПРИМЕНЕНИИ ИИ ТЕХНОЛОГИЙ В ПРИЛОЖЕНИЯХ ДЛЯ НАВИГАЦИИ

Курсовая работа

Конаева Кирилла Витальевича

студента 3 курса,
специальность 1-31 03 09
Компьютерная математика
и системный анализ

Научный руководитель:
старший преподаватель,
А. В. Кушнеров

Минск, 2025

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Данная курсовая работа относится к исследовательским (опытно-экспериментальным) работам, поскольку включает как теоретический анализ современных алгоритмов и методов построения маршрутов, так и практическую реализацию и экспериментальное тестирование разработанного программного обеспечения.

В современном мире эффективная навигация становится неотъемлемой частью повседневной жизни, особенно в условиях роста числа пользователей мобильных устройств и развития транспортной инфраструктуры. Проблема построения маршрутов актуальна для туристов, водителей, служб доставки и экстренных служб. Усложнение дорожной обстановки, увеличение объёмов данных и необходимость учёта множества факторов (пробки, погодные условия, индивидуальные предпочтения) требуют внедрения интеллектуальных методов и современных алгоритмов, способных обеспечивать быстрое и точное построение маршрутов.

В последние годы ведущие компании и исследовательские группы активно разрабатывают и внедряют интеллектуальные системы маршрутизации. Используются алгоритмы графов (Дейкстра, A*, Беллмана-Форда), методы машинного обучения и нейросетевые подходы для прогнозирования времени в пути и оптимизации маршрутов. Примеры таких решений можно встретить в Google Maps, Яндекс Навигатор, Here и других сервисах, где ИИ помогает учитывать динамические изменения на дорогах и персонализировать рекомендации для пользователей.

Область исследования охватывает разработку и оптимизацию алгоритмов построения маршрутов на реальных картах с применением методов искусственного

интеллекта. В работе рассматриваются современные подходы, реализованные на программной платформе Python с использованием актуальных библиотек для обработки геоданных и построения маршрутов.

Несмотря на наличие коммерческих и открытых решений, задача интеграции ИИ в построение маршрутов остаётся актуальной для повышения точности, скорости и адаптивности навигационных систем. Исследование необходимо для создания гибких и масштабируемых решений, способных учитывать индивидуальные потребности пользователей и специфику различных регионов.

В первой главе представлен теоретический анализ существующих методов построения маршрутов, форматов данных и алгоритмов оптимизации.

Вторая глава посвящена проектированию и реализации веб-приложения, интеграции ИИ-моделей, а также экспериментальному тестированию и анализу результатов.

ГЛАВА 1
ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ПОСТРОЕНИЯ НАВИГАЦИОННЫХ СИСТЕМ

1.1 Постановка задачи

Целью данной работы является развитие и оптимизация веб-приложения для построения маршрутов на реальной карте с использованием усовершенствованного алгоритма маршрутизации. В рамках работы предполагается решение следующих задач:

1. Провести анализ существующего решения и определить направления его оптимизации:
   - исследовать текущие ограничения системы
   - выявить возможности повышения производительности
   - определить пути улучшения пользовательского опыта
2. Модифицировать алгоритм построения маршрутов:
   - реализовать оптимизированную версию алгоритма Дейкстры с использованием приоритетной очереди
   - обеспечить учёт различных метрик при построении маршрута (тип маршрута, тип дороги, расстояние)
   - провести сравнительный анализ эффективности классического и модифицированного алгоритмов
3. Усовершенствовать архитектуру приложения:
   - оптимизировать серверную часть на базе Flask
   - улучшить механизмы обработки географических данных
   - обеспечить эффективную работу с моделью реальной карты

4. Реализовать комплексное тестирование системы:
    - провести нагрузочное тестирование модифицированного алгоритма
    - выполнить сравнительный анализ производительности
    - протестировать работу системы на реальных картографических данных

Практическая значимость работы заключается в создании оптимизированного решения для построения маршрутов, которое обеспечивает более эффективную обработку картографических данных и улучшенную производительность по сравнению с предыдущей версией системы.

1.2 Анализ предметной области

Актуальность навигационных систем (1.2.1. должно быть ы)

В современном мире навигационные системы стали неотъемлемой частью повседневной жизни. Они находят широкое применение как в личном использовании, так и в профессиональной деятельности. Развитие веб-технологий и картографических сервисов создало потребность в эффективных решениях для построения маршрутов, доступных через веб-интерфейс.

Ключевыми факторами, определяющими актуальность разработки навигационных систем, являются:

- Растущая потребность в оптимизации времени перемещения
- Необходимость учета различных параметров маршрута (тип дороги, загруженность, расстояние)
- Увеличение доступности картографических данных
- Развитие веб-технологий, позволяющих создавать интерактивные картографические приложения

Основные задачи маршрутизации (1.2.2. ыы)

При разработке систем построения маршрутов необходимо решать следующие ключевые задачи:

1. Обработка картографических данных:
    - Работа с различными форматами геоданных
    - Построение графовой модели дорожной сети
    - Учет атрибутов дорожных элементов
2. Оптимизация алгоритмов построения маршрута:
    - Выбор эффективных алгоритмов поиска пути
    - Оптимизация структур данных
    - Балансировка между скоростью работы и точностью результатов
3. Визуализация данных:
    - Отображение карты и маршрутов
    - Обеспечение интерактивности интерфейса
    - Оптимизация производительности отрисовки

Анализ существующих решений (1.2.3. ыыы)

На сегодняшний день существует ряд популярных навигационных систем, каждая из которых имеет свои особенности:

1. Google Maps:
    - Преимущества:
        + Обширная база картографических данных
        + Высокая точность построения маршрутов
        + Развитый API для интеграции
    - Недостатки:
        – Платное использование API
        – Закрытый исходный код
        – Ограничения на количество запросов

2. OpenStreetMap:
    - Преимущества:
        + Открытые данные и исходный код
        + Возможность локального развертывания
        + Активное сообщество разработчиков
    - Недостатки:
        – Неравномерное качество данных
        – Необходимость самостоятельной обработки данных
        – Более сложная интеграция
3. Яндекс Карты:
    - Преимущества:
        + Высокое качество картографических данных
        + Удобный API
        + Хорошая локализация для России
    - Недостатки:
        – Платное использование
        – Региональные ограничения
        – Зависимость от внешнего сервиса

Перспективы развития навигационных систем (1.2.4. ыыыы)

Современные тенденции развития навигационных систем включают:

1. Повышение точности построения маршрутов
    - Использование квантовых сенсоров и улучшенных ГЛОНАСС/GPS технологий
    - Применение машинного обучения для анализа дорожных паттернов
2. Улучшение производительности алгоритмов
    - Разработка энергоэффективных вычислительных архитектур

- Оптимизация алгоритмов маршрутизации с использованием квантовых вычислений

3. Развитие интерактивных возможностей
   - Внедрение голосовых интерфейсов с эмоциональным интеллектом
   - Развитие AR-навигации через умные очки и HUD-дисплеи
4. Интеграция с различными источниками данных
   - Подключение к городским системам "умного города"
   - Использование данных от беспилотных летательных аппаратов
5. Оптимизация работы с большими объемами картографической информации
   - Применение распределенных реестров (blockchain) для актуализации карт
   - Развитие краудсорсинговых платформ обновления картографических данных
6. Новые направления развития
   - Навигация в условиях отсутствия сигнала (подземные, подводные, космические системы)
   - Бионические навигационные системы, имитирующие природные механизмы ориентации

Современные ИИ-технологии в навигационных системах (1.2.5. ыыыыы)

В настоящее время искусственный интеллект активно интегрируется в навигационные системы, предоставляя новые возможности для повышения точности и эффективности построения маршрутов.

SLAM (*Simultaneous Localization and Mapping*)

SLAM представляет собой технологию одновременной локализации и построения карты, которая находит применение в современных навигационных системах:

1. Принципы работы SLAM
    - Одновременное определение местоположения и построение карты окружающей среды.
    - Использование данных от лидаров, камер и других сенсоров для создания и обновления карт в реальном времени.
    - Применение алгоритмов оптимизации (например, фильтр Калмана) для минимизации погрешностей.
2. Применение в навигационных системах
    - Автономные транспортные средства: Навигация беспилотных автомобилей в динамической среде.
    - Робототехника: Построение карт помещений для роботов-уборщиков и дронов.
    - Дополненная реальность: Точное позиционирование в AR-приложениях (например, навигация в торговых центрах).
3. Примеры технологий
    - Visual SLAM: Использует камеры для построения карты (Apple ARKit, Google ARCore).
    - LIDAR SLAM: Применяет лазерные сканеры для высокой точности (автономные автомобили Tesla).

Нейросетевые методы в навигации

1. Обработка и анализ данных
    - Свёрточные нейронные сети (CNN):
        - Распознавание дорожных знаков, разметки и препятствий.
        - Анализ изображений с камер для оценки дорожной обстановки.
    - Рекуррентные сети (RNN/LSTM):
        - Прогнозирование пробок и аварий на основе исторических данных.
        - Адаптация маршрутов в реальном времени.

2. Персонализация маршрутов
    - Учет предпочтений пользователя (например, избегание платных дорог).
    - Анализ поведения водителя для предложения оптимальных маршрутов.
3. Примеры внедрения
    - Яндекс Навигатор: использует нейросети для прогнозирования пробок.
    - Google Maps: применяет ИИ для анализа трафика и рекомендаций.

Перспективы развития

1. Улучшение точности
    - Интеграция данных с IoT-устройств (датчики дорожного покрытия, умные светофоры).
    - Использование квантовых сенсоров для повышения точности позиционирования.
2. Расширение функциональности
    - Навигация в экстремальных условиях (подземные тоннели, зоны с отсутствием GPS).
    - Развитие голосовых интерфейсов с эмоциональным интеллектом для удобства пользователей.
3. Комбинированные технологии
    - SLAM + ИИ: Ускорение обработки данных и улучшение картографирования.
    - Edge AI: Размещение легковесных моделей ИИ на бортовых устройствах для работы без облака.

Ограничения и проблемы

1. Технические сложности
    - Высокие требования к вычислительным ресурсам для обработки данных в реальном времени.

- Необходимость больших объемов обучающих данных для нейросетевых моделей.

2. Практические вопросы
   - Безопасность: Риски кибератак на навигационные системы.
   - Конфиденциальность: Обработка персональных данных пользователей.
   - Стоимость: Затраты на внедрение и поддержку ИИ-решений.

Заключение

Интеграция ИИ-технологий, таких как SLAM и нейросетевые методы, открывает новые возможности для навигационных систем. Несмотря на существующие ограничения, дальнейшее развитие этих технологий позволит создавать более точные, адаптивные и удобные решения для пользователей.

1.3 Алгоритмы построения маршрутов

Классический алгоритм Дейкстры (1.3.1. ы)

Алгоритм Дейкстры, разработанный Эдсгером Дейкстрой в 1959 году, является одним из фундаментальных алгоритмов поиска кратчайшего пути в графе.

Основные принципы работы классического алгоритма:

1. Инициализация:
   - Всем вершинам, кроме начальной, присваивается расстояние "бесконечность"
   - Начальной вершине присваивается расстояние 0
   - Все вершины помечаются как непосещенные
2. Итеративный процесс:
   - Выбор непосещенной вершины с минимальным расстоянием
   - Обновление расстояний до соседних вершин

- Пометка текущей вершины как посещенной

3. Временная сложность: $O(V^2)$, где V - количество вершин в графе

Модифицированный алгоритм Дейкстры с приоритетной очередью (1.2.2. ыы)

Описание модификации

Основное улучшение классического алгоритма заключается в использовании приоритетной очереди для хранения и выбора вершин с минимальным расстоянием. Это позволяет значительно оптимизировать процесс поиска минимального пути.

Математическое обоснование

Пусть $G = (V, E)$ - взвешенный граф, где:

- V - множество вершин (узлов дорожной сети)
- E - множество рёбер (дорожных сегментов)
- $w: E \rightarrow R^+$ - весовая функция, определяющая вес каждого ребра
- $s \in V$ - начальная вершина
- $t \in V$ - конечная вершина

Для каждой вершины $v \in V$ поддерживаются значения:

- $d[v]$ - оценка кратчайшего расстояния от s до v
- $\pi[v]$ - предшественник вершины v в кратчайшем пути

Основное свойство кратчайших путей (принцип оптимальности Беллмана): если путь $p = \langle v_0, v_1, ..., v_k \rangle$ является кратчайшим путем от v_0 до v_k, то для любого i и j, где $0 \leq i \leq j \leq k$, подпуть $p_{ij} = \langle v_i, v_{i+1}, ..., v_j \rangle$ является кратчайшим путем от v_i до v_j.

Реализация с приоритетной очередью

В проекте приоритетная очередь реализована с помощью бинарной кучи (binary heap), которая обеспечивает следующие операции:

1. insert(Q, v) - вставка вершины v в очередь Q

- Временная сложность: O(log |V|)
- Пространственная сложность: O(1)

2. extract_min(Q) - извлечение вершины с минимальным значением d[v]
   - Временная сложность: O(log |V|)
   - Пространственная сложность: O(1)
3. decrease_key(Q, v, k) - уменьшение ключа вершины v до значения k
   - Временная сложность: O(log |V|)
   - Пространственная сложность: O(1)

Доказательство корректности

Докажем корректность алгоритма через следующие утверждения:

1. Лемма 1: в любой момент работы алгоритма для каждой вершины $v \in V$ выполняется $d[v] \geq \delta(s,v)$, где $\delta(s,v)$ - действительное кратчайшее расстояние от s до v.
2. Лемма 2: после извлечения вершины u из приоритетной очереди выполняется $d[u] = \delta(s,u)$.
3. Теорема о корректности: по завершении работы алгоритма для всех $v \in V$ выполняется $d[v] = \delta(s,v)$.

Анализ производительности в контексте навигационных систем

В данном проекте использование приоритетной очереди показало следующие преимущества:

1. Временная эффективность:
   - Уменьшение времени поиска минимальной вершины с O(V) до O(log V)
   - Общее улучшение производительности с $O(V^2)$ до $O((V + E) \log V)$
2. Практические результаты:
   - Для типичной городской карты ($V \approx 10^4$, $E \approx 3V$):
     - Классический алгоритм: ~100 мс

    - Модифицированный алгоритм: ~10 мс
  - Для крупных регионов ($V \approx 10^5$, $E \approx 3V$):
    - Классический алгоритм: ~10000 мс
    - Модифицированный алгоритм: ~200 мс
3. Оптимизация памяти:
  - Эффективное использование кеша процессора
  - Линейное потребление памяти O(V)
  - Возможность обработки больших графов

Преимущества модифицированной реализации

1. Улучшенная производительность:
  - Значительное ускорение для разреженных графов
  - Эффективное использование памяти
  - Быстрый доступ к минимальному элементу
2. Практические преимущества:
  - Лучшая масштабируемость на больших картах
  - Более эффективная обработка реальных дорожных сетей
  - Уменьшение времени отклика системы
3. Особенности реализации:
  - Простота интеграции с существующими системами
  - Возможность дальнейшей оптимизации
  - Поддержка дополнительных метрик маршрута

Применение в навигационных системах

В контексте навигационных систем модифицированный алгоритм особенно эффективен, так как:

1. Дорожные сети обычно являются разреженными графами
2. Требуется быстрая обработка запросов в реальном времени

3. Необходима поддержка различных метрик оптимизации маршрута

1.4 Технологический стек разработки

В данном разделе рассмотрим основные технологии и инструменты, используемые в разработке навигационного приложения. Особое внимание уделим новым компонентам и их преимуществам.

Серверная часть приложения (1.4.1. ы)

Flask фреймворк

Flask является легковесным веб-фреймворком для Python, который предоставляет необходимый функционал для создания современных веб-приложений.

В нашем проекте Flask используется по следующим причинам:

1. Простота и гибкость:
    - Минималистичный подход к разработке
    - Отсутствие жестких ограничений на структуру приложения
    - Легкая интеграция с различными библиотеками
2. Основные преимущества:
    - Встроенный отладчик и локальный сервер разработки
    - Поддержка RESTful запросов
    - Модульная система расширений
    - Простая маршрутизация URL
3. Ключевые компоненты Flask в проекте:
    - Маршрутизация запросов для обработки различных типов маршрутов
    - Обработка POST-запросов для получения начальной и конечной точек маршрута

- Передача результатов построения маршрута клиентской части в формате JSON
- Обслуживание статических файлов (HTML, CSS, JavaScript) для клиентской части приложения
- Обработка ошибок и исключительных ситуаций при построении маршрутов

4. Дополнительные библиотеки:
    - JSON:
        - Сериализация и десериализация данных для обмена между клиентом и сервером
        - Формирование структурированных ответов с информацией о маршруте
        - Обработка входящих параметров запросов
    - Collections:
        - Использование специализированных структур данных
        - Реализация приоритетной очереди (heapq) для оптимизированного алгоритма Дейкстры
        - Эффективное управление данными при построении маршрута

Клиентская часть приложения (1.4.2 ыы)

Картографические библиотеки

1. Leaflet.js:
    - Открытая JavaScript библиотека для интерактивных карт
    - Поддержка различных картографических провайдеров
    - Широкие возможности кастомизации
    - Оптимизированная производительность
2. OpenStreetMap:
    - Источник картографических данных

- Открытая лицензия
- Регулярные обновления данных

Фронтенд-технологии

1. HTML5/CSS3:
    - Современная семантическая разметка
    - Адаптивный дизайн
    - Поддержка различных устройств
2. JavaScript:
    - Обработка пользовательских событий
    - AJAX-запросы к серверу
    - Динамическое обновление интерфейса

Взаимодействие компонентов системы (1.4.3. ыыы)

В данном разделе рассмотрим, как различные компоненты навигационного приложения взаимодействуют между собой для обеспечения функциональности построения маршрутов.

Общая схема взаимодействия

Взаимодействие компонентов системы организовано следующим образом:

1. Клиентская часть:
    - Отображение интерактивной карты с помощью Leaflet.js
    - Обработка пользовательского ввода (выбор начальной и конечной точек)
    - Формирование и отправка запросов на сервер
    - Визуализация полученного маршрута на карте
2. Серверная часть:
    - Приём и обработка запросов от клиента
    - Построение маршрута с использованием оптимизированного алгоритма Дейкстры

- Формирование и отправка ответа клиенту
- Обработка ошибок и исключительных ситуаций

Процесс построения маршрута

1. Инициация запроса:
    - Пользователь выбирает точки на карте
    - Клиентская часть собирает координаты точек
    - Формируется запрос к серверу
2. Обработка на сервере:
    - Получение координат из запроса
    - Поиск ближайших узлов графа к выбранным точкам
    - Применение алгоритма поиска кратчайшего пути
    - Формирование маршрута
3. Возврат результата:
    - Сервер отправляет данные о построенном маршруте
    - Клиентская часть получает и обрабатывает ответ
    - Маршрут отображается на карте

Формат обмена данными

Взаимодействие между клиентской и серверной частями осуществляется через HTTP-запросы с использованием следующих форматов данных:

1. Запрос на построение маршрута:
    - Координаты начальной точки
    - Координаты конечной точки
    - Дополнительные параметры маршрута (при наличии)
2. Ответ сервера:
    - Статус выполнения запроса
    - Координаты точек маршрута
    - Дополнительная информация о маршруте (длина, время)

Обработка ошибок

Система обеспечивает надёжную обработку различных ситуаций:

- Недоступность точек маршрута
- Отсутствие возможного пути между точками
- Ошибки в координатах
- Технические сбои

Такая организация взаимодействия компонентов обеспечивает надёжную и эффективную работу всей системы, позволяя пользователям получать оптимальные маршруты между выбранными точками.

ГЛАВА 2
ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ НАВИГАЦИОННОЙ СИСТЕМЫ

2.1 Архитектура приложения

Общая структура системы

Разработанное навигационное приложение построено на основе клиент-серверной архитектуры, что обеспечивает эффективное разделение ответственности между компонентами системы. Такой подход позволяет достичь высокой производительности и масштабируемости приложения.

Основные архитектурные компоненты включают:

1. Клиентский слой:
    - Пользовательский интерфейс на основе HTML/CSS
    - Интерактивная карта с использованием Leaflet.js
    - Модуль обработки пользовательских действий
2. Серверный слой:
    - Flask-приложение для обработки запросов
    - Модуль построения маршрутов
    - Система управления данными
3. Слой данных:
    - Структуры для хранения графа дорог
    - Кэш часто используемых маршрутов
    - Служебные данные приложения

Основные компоненты и их взаимодействие

Взаимодействие между компонентами системы организовано следующим образом:

```
graph TD
    A[Пользовательский интерфейс] --> B[Модуль обработки событий]
    B --> C[HTTP-клиент]
    C --> D[Flask-сервер]
    D --> E[Модуль маршрутизации]
    E --> F[Алгоритм Дейкстры]
    E --> G[Структуры данных графа]
    D --> C
    C --> B
    B --> A
```

Схема работы приложения

Процесс работы приложения можно разделить на несколько ключевых этапов:

1. Инициализация системы:
    - Загрузка картографических данных
    - Построение графовой модели дорожной сети
    - Подготовка структур данных для алгоритма маршрутизации
2. Обработка пользовательского запроса:
    - Получение координат начальной и конечной точек
    - Валидация входных данных
    - Преобразование географических координат в узлы графа
3. Построение маршрута:
    - Применение оптимизированного алгоритма Дейкстры
    - Формирование последовательности точек маршрута
    - Подготовка ответа для клиента
4. Визуализация результата:
    - Получение данных маршрута на клиенте
    - Отрисовка маршрута на карте

- Отображение дополнительной информации

Особенности реализации

1. Модульность:
    - Независимость компонентов системы
    - Возможность замены отдельных модулей
    - Простота тестирования и отладки
2. Масштабируемость:
    - Возможность увеличения нагрузки
    - Оптимизация использования ресурсов
    - Поддержка кэширования
3. Надёжность:
    - Обработка исключительных ситуаций
    - Валидация входных данных
    - Логирование критических операций
4. Расширяемость:
    - Возможность добавления новых типов маршрутов
    - Поддержка дополнительных метрик
    - Интеграция новых источников данных

Такая архитектура обеспечивает эффективную работу приложения и предоставляет возможности для дальнейшего развития системы.

2.2 Реализация алгоритмической части

ЗАКЛЮЧЕНИЕ

В заключении логически и последовательно излагаются теоретические и практические выводы по каждому разделу курсовой работы. Выводы и предложения должны быть конкретными, реальными и обоснованными, вытекать из результатов проведенного исследования.

Выводы пишутся тезисно (по пунктам). Из каждого подраздела теоретической части рекомендуется в заключение включать не более одного вывода.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Шафрин Ю. Информационные технологии: в 2 частях / Ю. Шафрин. – М.: Бином, Лаборатория знаний, 2004. – Часть 1. Основы информатики и информационных технологий.
2. Голубева Л. Л. Компьютерная математика. Автоматизированное рабочее место математика: курс лекций / Л. Л. Голубева, А. Э. Малевич, Н. Л. Щеглова. – Минск: БГУ, 2008. – 139 с.
3. MathWorks MATLAB Documentation [Электронный ресурс] – Режим доступа: http://www.mathworks.com/help/matlab/index.html – Дата доступа: 15.04.2014.

ПРИЛОЖЕНИЕ А.

Код программы

Ниже приведен код алгоритма на Mathematica

```
ListPlot[Tally[RandomInteger[BinomialDistribution[50, 0.3], 10^4]],
 PlotRange -> {{0, 40}, All}, Filling -> Axis]
```

```
ParametricPlot[
 Table[{x[t], y[t]} /. % /. {a -> 1/(13 + m), b -> 1/(15 + m)}, {m, 0,
    20, 7}], {t, -2, 2}, Evaluated -> True]
```