|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_\_\_\_\_ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_\_\_\_СИСТЕМЫ ОБРАБОТКИ ИФОРМАЦИИ И УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_\_

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ***

***НА ТЕМУ:***

***Анализ эффективного создания и отображения геобъектов и маршрутов в нагруженных геоинформационных системах \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

Студент \_\_\_ИУ5-33М\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_**Е.И.Жидков**\_\_\_\_\_\_\_**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_**Ю.Е. Гапанюк**\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

Консультант **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

*2024 г.*

**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой \_\_\_ИУ5\_\_

(Индекс)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_В.И. Терехов\_\_

(И.О.Фамилия)

« \_\_\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 20 \_\_\_\_ г.

**ЗАДАНИЕ**

**на выполнение научно-исследовательской работы**

по теме \_\_ *Анализ эффективного создания и отображения геобъектов и маршрутов в нагруженных геоинформационных системах\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

Студент группы \_\_\_ИУ5-33М\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Жидков Егор Ильич\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия, имя, отчество)

Направленность НИР (учебная, исследовательская, практическая, производственная, др.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_учебная\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Источник тематики (кафедра, предприятие, НИР) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

График выполнения НИР: 25% к \_5\_ нед., 50% к \_9\_ нед., 75% к \_13\_ нед., 100% к \_17\_ нед.

Техническое задание Изучить существующие методы и подходы к визуализации и маршрутизации. Разработать подсистему для работы с геопространственными данными, включая фильтрацию и кластеризацию. Реализовать функции маршрутизации, включая автоматическое и пользовательское построение маршрутов. Разработать инструментарий для обновления и добавления новых объектов и данных. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***Оформление научно-исследовательской работы:***

Расчетно-пояснительная записка на \_\_\_\_\_ листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания «09» \_\_сентября\_\_ 2024 г.

**Руководитель НИР**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_Ю.Е. Гапанюк\_\_\_\_

(Подпись, дата) (И.О.Фамилия)

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_**Е.И. Жидков\_\_\_\_\_

(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

Оглавление

[Введение 3](#_Toc185783123)

[Цели 3](#_Toc185783124)

[Сравнение возможностей отрисовки объектов в ГИС-системах: Yandex API, OpenLayers и OpenStreetMap 4](#_Toc185783125)

[Способы отрисовки данных на карте 5](#_Toc185783126)

[Алгоритм RemoteObjectManager 6](#_Toc185783127)

[Функционал построения маршрутов 8](#_Toc185783128)

[Сравнение, эксперименты и результаты 13](#_Toc185783129)

[Заключение 19](#_Toc185783130)

[Список литературы 21](#_Toc185783131)

Введение

Разработка эффективных картографических сервисов требует использования современных технологий для обеспечения высокой производительности, точности и удобства работы с большими объемами данных. В рамках проекта «Навигатор» была реализована система, которая позволяет оптимизированно отображать объекты на карте с учетом различных уровней приближения и географических фильтров. В этой системе активно используются передовые инструменты фронтенд-разработки, такие как React, и мощные картографические API, например, Yandex API, который предоставляет широкий спектр функций для работы с картами. Для эффективного хранения и быстрого доступа к данным применяется сочетание Postgres и Redis [1].

Одной из важнейших задач в разработке была оптимизация процесса отрисовки объектов на карте [2], что позволяет работать с тысячами объектов без снижения производительности. Для этого была выбрана серверная реализация RemoteObjectManager, которая минимизирует нагрузку на клиентскую часть системы и предоставляет гибкие возможности для кластеризации объектов. Алгоритм, реализованный в этой системе, включает этапы получения и обработки сущностей, что позволяет эффективно управлять большими объемами данных и обеспечивать точное отображение объектов в зависимости от уровня приближения карты.

# Цели

На сегодняшний день при разработке систем ГИС существует актуальная проблема, связанная с размещение большого количества объектов и построением маршрутов. Необходимо, чтобы при взаимодействии пользователя с интерфейсом данные отображались быстро и с минимальной задержкой. Это требует эффективного подхода, то есть такого, который обеспечивает оптимальное использование ресурсов системы (вычислительных, сетевых и памяти) для достижения максимальной производительности при минимальных затратах времени и усилий. Исходя из этого требуется. Улучшить скорость и эффективность отображения множества объектов на карте и формирования маршрутов посредством применения разнообразных методик и алгоритмов.

****Сравнение возможностей отрисовки объектов в ГИС-системах: Yandex API, OpenLayers и OpenStreetMap****

При разработке ГИС-систем выбор платформы для отрисовки объектов играет ключевую роль. **Yandex API** предлагает мощные возможности для маршрутизации, включая поддержку динамических данных о дорожной обстановке и обновляемые дорожные графы. Это делает его удобным для приложений, где критически важны высокое качество маршрутов и актуальность данных. Однако Yandex API имеет закрытую экосистему и требует соблюдения условий лицензии, что ограничивает гибкость интеграции.

**OpenLayers**, напротив, является полностью открытой библиотекой, обеспечивающей высокую степень кастомизации. Она поддерживает широкий спектр источников данных, включая GeoJSON, WMS и WMTS, и предоставляет богатый набор инструментов для обработки и визуализации геопространственных данных. OpenLayers лучше всего подходит для проектов, где требуется высокая степень контроля над отрисовкой и функционалом карты. Однако в сравнении с Yandex API, OpenLayers требует больше усилий для настройки и не предоставляет встроенной маршрутизации.

**OpenStreetMap (OSM)** выступает как источник данных и, в отличие от Yandex API и OpenLayers, сам по себе не является API. Однако, используя API поверх OSM, такие как Leaflet или Mapbox, разработчики могут создать гибкие карты с подробной информацией об объектах. Основное преимущество OSM – открытые данные, которые можно адаптировать под различные задачи. Однако обновление данных в OSM зависит от сообщества, что может привести к менее актуальной информации о дорожной сети в сравнении с Yandex API.

В итоге, выбор платформы зависит от конкретных требований: для коммерческих приложений с высокой точностью и актуальностью данных выгоднее использовать Yandex API, для кастомных решений – OpenLayers, а для проектов с ограниченным бюджетом или открытым исходным кодом – OpenStreetMap в сочетании с соответствующими библиотеками.

Способы отрисовки данных на карте

Производительность карты — одна из ключевых задач. Для отрисовки объектов существует два основных подхода: создание геобъектов или использование объектного менеджера. Создание и обработка геобъектов достаточно ресурсоемка и подходит, если на карте не более сотни объектов. При большем количестве объектов рекомендуется объектный менеджер, работающий не с геобъектами напрямую, а с их оверлеями — визуальными представлениями объектов, требующими меньше ресурсов для обработки.

В Yandex API доступно три типа менеджеров: ObjectManager, LoadingManager и RemoteObjectManager [3].

ObjectManager — наиболее простая структура, которая принимает JSON-описание объектов [4], а затем отрисовывает их оверлеи и, при необходимости, создает кластеры. LoadingObjectManager дополнительно берет на себя загрузку данных с сервера, но логика отрисовки и кластеризации сохраняется как у ObjectManager. RemoteObjectManager также загружает данные с сервера, но не выполняет кластеризацию самостоятельно, полагаясь на заранее кластеризованные данные, полученные с сервера.

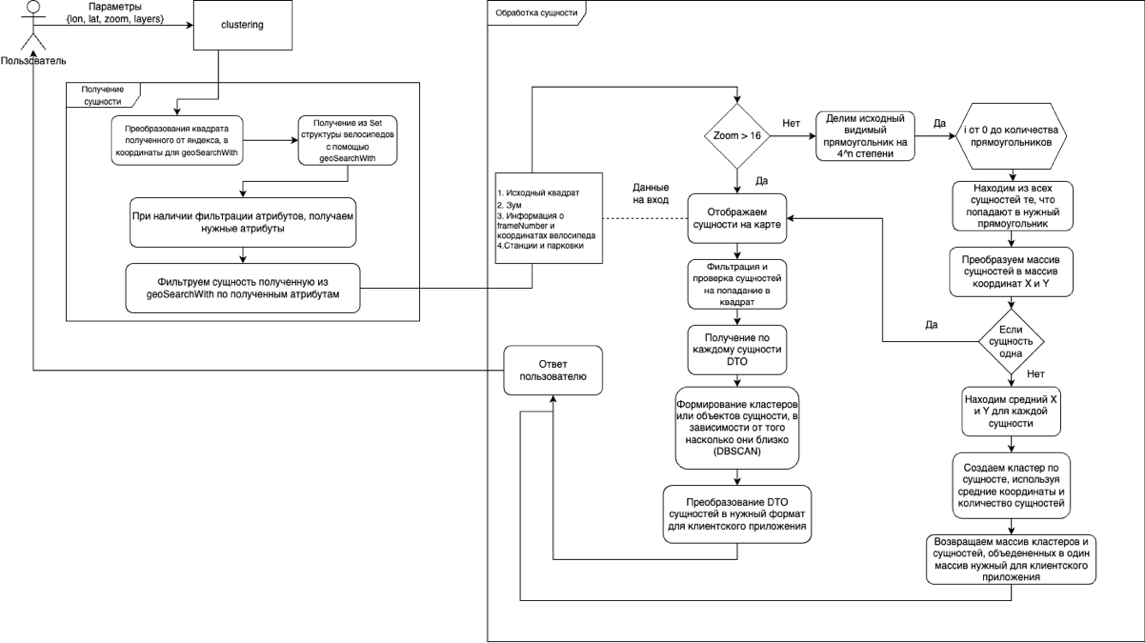
*ObjectManager* требует ручной загрузки данных и создания необходимой JSON-структуры на клиенте. Если метка оверлея должна быть динамической, потребуется проходить по каждому объекту, вычислять его параметры и создавать новый класс метки. Это становится особенно трудоемко, если шаблон метки содержит условия или массивы. Кроме того, для кластеризации используется вычислительная мощность браузера, что увеличивает нагрузку на обработку и рендеринг. Настроить принцип кластеризации объектов также не представляется возможным.

*LoadingObjectManager* самостоятельно загружает данные с сервера, что требует дополнительной серверной реализации, однако он сохраняет остальные ограничения ObjectManager.

*RemoteObjectManager* требует сложной серверной реализации для генерации объектов и ручного создания кластеров. Этот подход предполагает значительные вычислительные нагрузки на сервер, что требует увеличенных мощностей для обработки данных.

Алгоритм RemoteObjectManager

На рис. 1 представлена схема работы данного алгоритма. В ней есть два основных блока: «Получение сущности» и «Обработка сущности». Рассмотрим подробнее каждый из блоков.



**Рис. 1.** Общая схема алгоритма системы

Блок «Получение сущности» отвечает за доступ к данным в Redis. Для обеспечения высокой скорости доступа данные хранятся в структуре SET в формате «идентификатор-координаты», что предоставляет преимущества, такие как уникальность элементов и высокая скорость операций благодаря использованию хэш-таблиц. Поскольку пользователь чаще всего взаимодействует с общей областью карты и незначительным уровнем приближения, нет необходимости загружать все атрибуты сущности сразу.

Для получения требуемых сущностей координаты левого нижнего и правого верхнего углов карты, полученные от пользователя, преобразуются в параметры для команды «GEOSEARCHWITH» [5]. Эта команда выполняет поиск объектов в заданном географическом радиусе от опорной точки, предоставляя возможность дополнительно возвращать координаты и расстояния до найденных объектов. Для универсальности алгоритма, при необходимости фильтрации объектов по какому-либо атрибуту, запрашиваются атрибуты фильтрации для полученных сущностей, после чего осуществляется отбор только тех, которые соответствуют заданным условиям. После этого данные попадают в следующий блок.

Блок «Обработка сущности» отвечает за формирование JSON-структуры, которую будет отрисовывать браузер пользователя. Сначала выполняется проверка уровня приближения карты. Если значение приближения меньше порогового, то прямоугольная область, заданная пользователем, разбивается на прямоугольников, где n — это количество уровней деления, минимальное 3. На следующем этапе необходимо обработать каждый полученный прямоугольник и найти все сущности, которые попадают в его границы. Эти сущности преобразуются в массив координат. Поскольку точность расположения объектов на этом уровне не имеет критического значения, а, количество прямоугольников минимум 64, объекты будут размещаться по центру каждого прямоугольника в виде кластера.

Если значение приближения карты превышает пороговое, то становится важно отрисовать не только кластеры, но и сами объекты. В этом случае область не разбивается на прямоугольники. Поскольку это уже не кластеризация, может потребоваться информация о сущностях, и для этого мы получаем необходимые атрибуты.

Поскольку объекты могут перекрывать друг друга, если находятся вблизи, для кластеризации используется более мощный инструмент — метод DBSCAN [6]. Этот метод был выбран, так как он способен строить кластеры без предварительного знания их количества. Он группирует объекты по их плотности, что позволяет настраивать расстояние, при котором объекты считаются частью одного кластера, в зависимости от масштаба. Таким образом, можно достичь высокой точности при построении кластеров.

Полученные объекты и кластеры преобразуются в нужную структуру и отправляются пользователю для отрисовки.

Учитывая перечисленные недостатки, предпочтительнее использовать RemoteObjectManager, так как он позволяет минимизировать клиентскую нагрузку на вычисления данных, обеспечивает возможность создания пользовательской кластеризации по заданному алгоритму, избавляет от необходимости пересчитывать динамические метки и позволяет расширять сущности кластера без обработки атрибутов каждой отдельной составляющей.

В связи с этим был разработан сервис, который способен отрисовывать оптимизировано объекты с помощью серверной реализации RemoteObjectManager.

Функционал построения маршрутов

В системе «Навигатор» также реализован функционал для построения маршрутов, который позволяет пользователю генерировать маршруты по объектам ракетно-космической отрасли. Система поддерживает два типа маршрутов: автоматически сгенерированные и сгенерированные пользователем в реальном времени.

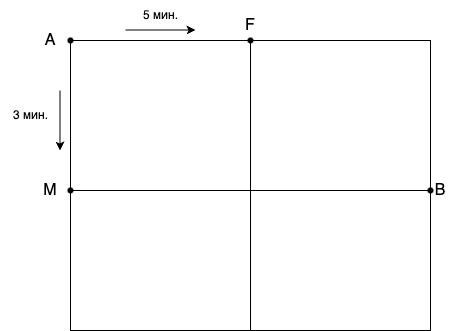
Автоматически сгенерированные маршруты строятся с использованием предустановленных алгоритмов на основе данных о расположении объектов и заданных параметров маршрута. Однако для более интерактивного подхода предусмотрена возможность создания маршрутов пользователем в реальном времени. Пользователь может выделить n-угольник на карте, который будет определять интересующую его область, и построить маршрут по объектам, попавшим в эту область.

Кроме того, на точки, попавшие в область маршрута, могут быть наложены различные фильтры. Это могут быть фильтры по количеству свободного времени, виду транспорта, а также приоритизация точек маршрута. Для построения маршрутов используется API Яндекс Карт, который предоставляет функционал для вычисления маршрутов между точками на карте. В основе маршрутизации стоит две основные вещи – дорожный граф и алгоритм вычисление оптимального пути, в нашем случае это алгоритм Дейкстры [7].

Дорожный граф – сетка дорог, которая состоит из большого количества соединенных между собой фрагментов. Каждый фрагмент несет в себе большое количество информации: координаты, направление движения, средняя скорость перемещения. Дорожный граф не является статическим, так как появляются новые пути и изменяются старые, в нашем случае API Яндекс карт сам следит за их обновлением. Так как дорожный граф не является статическим, то требуется применять к нему алгоритм для вычисления оптимального пути.

Для вычисления оптимального маршрута, система использует информацию о каждом отрезке в графе и скорость движения на каждом участке. Рассмотрим более подробно алгоритм вычисления маршрута.

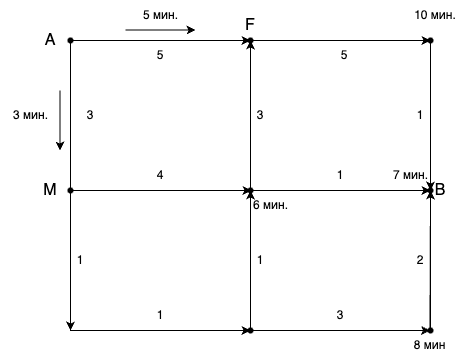
Предположим, что нам требуется добраться из точки A в точку B. Алгоритм начинает с точки А, присваивая ей начальное растояние 0, а остальным вершинам значение бесконечности. Алгоритм проверяет все ребра, исходящие из текущей вершины, и обновляет минимальные расстояния до соседей, если найден путь короче. что показано на рис. 2.



**Рис. 2.** Первая итерация алгоритмы Дейкстра

**Fig. 2.** First iteration of Dijkstra's algorithms

Как только алгоритм завершает анализ всех исходящих ребер текущей вершины, она помечается как обработанная, чтобы избежать её повторной проверки. После этого алгоритм выбирает следующую вершину с минимальным значением пути среди необработанных. Затем алгоритм строит маршрут еще на один шаг во все стороны от точки M, анализируя в какую из точек можно попасть быстрее всего. Продолжая таким образом, алгоритм будет итеративно проходится по вершинам, пока требуемая вершина не станет обработанной, тем самым мы будем знать, какое кратчайшее время потребутеся для перемещения в точку B, пример которого показан на рис. 3, где указаны все веса ребер в графе.



**Рис. 3.** Построенный маршрут с помощью алгоритма Дейкстры

**Fig. 3.** Constructed route using Dijkstra's algorithm

У алгоритма Дейкстры есть особенности и ограничения. Алгоритм может работать только во взевешноом графе с неотрицательными весами. Взвешанный граф – граф, значение к ребрам которого поставлено соответсвенное значние. Неотрицательные веса подразумевают, что эти значения не могут быть отрицательными. В нашем случае, объект не способен перемещаться с отрицательным временем, и мы знаем среднее время прохождение между ребрами. Также в алгоритме можно задать предпочтения маршрутам с низкой стоимостью, вместо равномерного исследования всех возможных путей, он будет

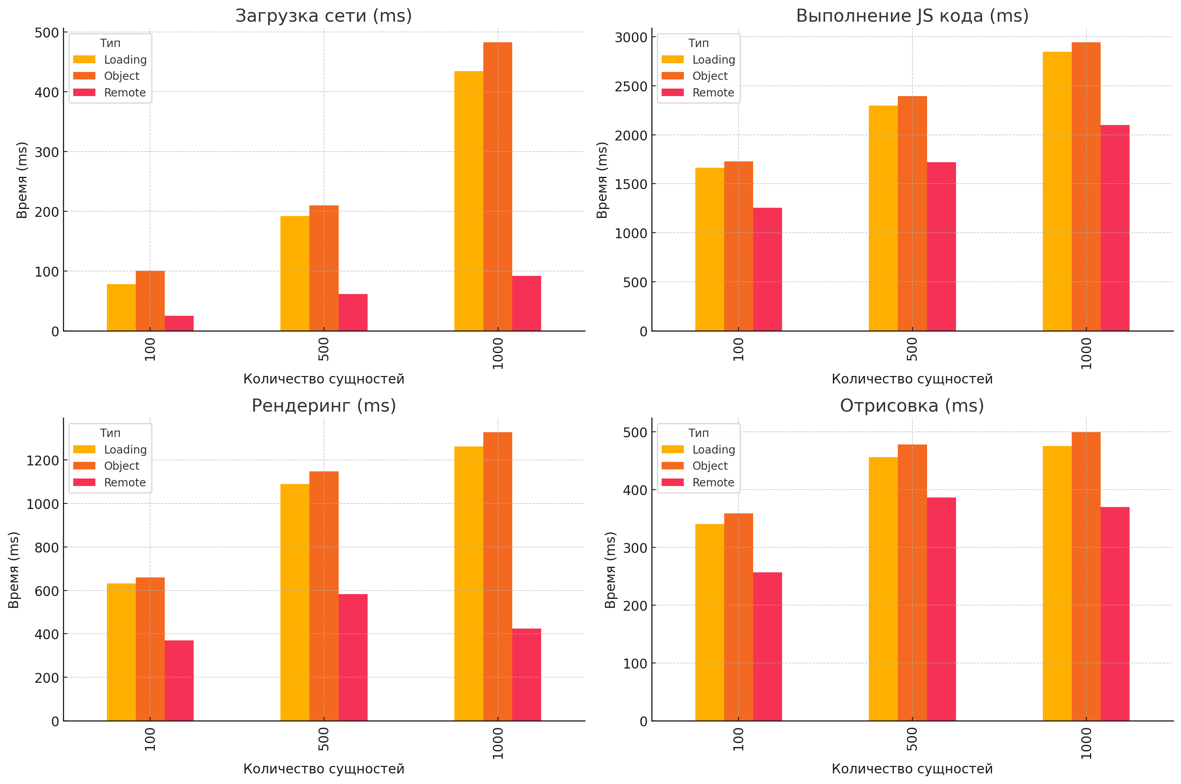
Помимо алгоритма Дейкстры, в системах маршрутизации могут применяться следующие подходы:

1. **Алгоритм A**\*  
   Алгоритм A\* является модификацией алгоритма Дейкстры, который использует эвристическую функцию для предсказания расстояния от текущей вершины до конечной. Алгоритм работает следующим образом:
   * На каждой итерации он выбирает вершину, которая минимизирует сумму текущего расстояния от начальной точки (g(x)) и эвристической оценки расстояния до конечной точки (h(x)).
   * Эвристическая функция, например, может использовать евклидово расстояние между точками.
   * Алгоритм продолжает обработку вершин до тех пор, пока не достигнет конечной точки.
   * A\* позволяет существенно сократить количество проверяемых вершин по сравнению с алгоритмом Дейкстры, если эвристика точна.
2. **Поиск волнами (Breadth-First Search)**  
   Этот метод применяется для графов с одинаковыми весами ребер. Принцип работы:
   * Алгоритм начинает с начальной вершины и посещает всех её соседей.
   * Затем он переходит ко всем соседям следующего уровня (то есть ко всем вершинам, находящимся на одинаковом расстоянии от начальной точки).
   * Обход продолжается, пока не будет достигнута конечная точка. Этот метод эффективен для поиска кратчайшего пути по числу переходов, но его применение ограничено невзвешенными графами.
3. **Алгоритм Беллмана-Форда**  
   Этот алгоритм, в отличие от Дейкстры, может работать с графами, содержащими отрицательные веса ребер. Принцип работы:
   * Алгоритм инициализирует расстояния до всех вершин как бесконечные, за исключением начальной вершины.
   * Затем он последовательно проходит по всем ребрам графа, проверяя и обновляя минимальное расстояние до каждой вершины.
   * Этот процесс повторяется n-1 раз (где n — количество вершин в графе), что гарантирует нахождение кратчайших путей.
   * Если после этого обнаруживаются отрицательные циклы (когда расстояния продолжают уменьшаться), алгоритм сообщает об их наличии.
4. **Жадные алгоритмы**  
   Принцип работы жадных алгоритмов основывается на том, чтобы на каждом шаге выбирать локально оптимальное решение в надежде, что оно приведет к глобально оптимальному результату. Например:
   * Алгоритм стартует из начальной вершины и всегда выбирает ребро с минимальным весом, ведущее к еще необработанной вершине.
   * Такой подход позволяет быстро находить приемлемые маршруты, но результат может не быть оптимальным, поскольку не исследуются альтернативные пути.
5. **Генетические алгоритмы**  
   Это эволюционный подход, который моделирует процесс естественного отбора. Алгоритм работает следующим образом:
   * Создается начальная популяция маршрутов (решений), которые оцениваются с точки зрения их "пригодности" (например, по длине или времени).
   * Затем маршруты "скрещиваются" и подвергаются "мутациям" для создания нового поколения маршрутов.
   * Каждый цикл отбора, скрещивания и мутации приближает решения к оптимальному результату.
   * Этот метод особенно полезен в задачах с высокой комбинаторной сложностью, где невозможно эффективно использовать классические алгоритмы.

Эти методы дополняют функционал маршрутизации, предоставляя широкий спектр инструментов для работы с различными типами графов и условиями. В следующем разделе можно будет провести их сравнение по эффективности, точности, сложности реализации и применимости в реальных задачах.

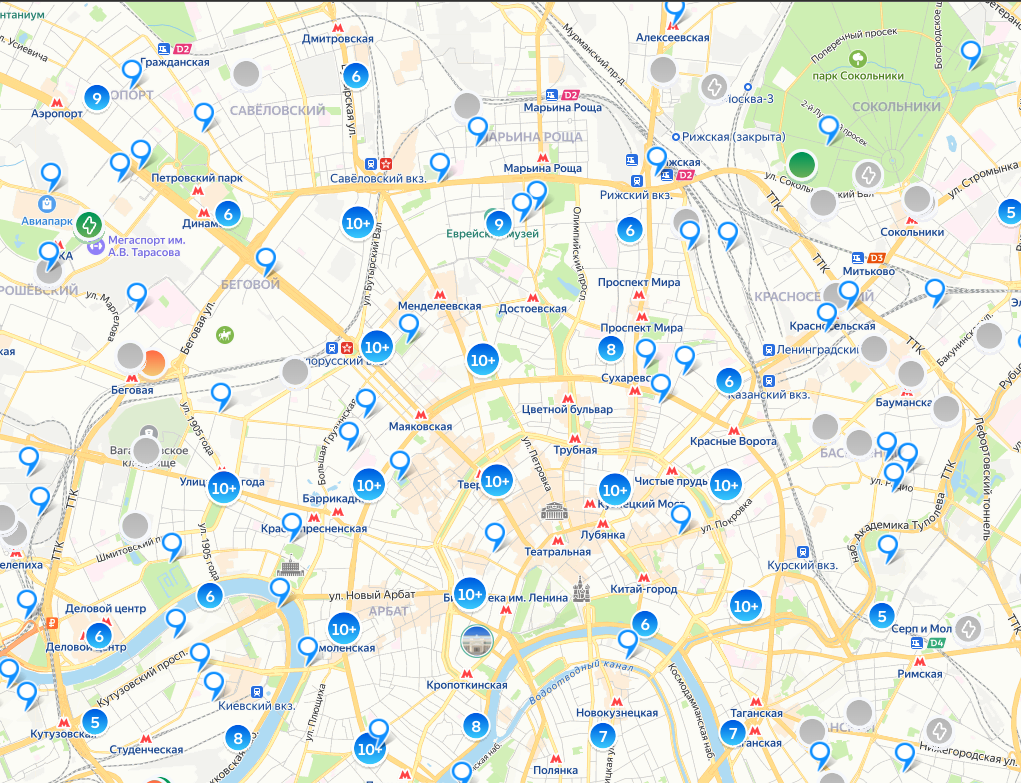
# Сравнение, эксперименты и результаты

В процессе разработки системы «Навигатор» было проведено несколько экспериментов с целью сравнения различных подходов к отображению объектов на карте. В данной части работы представлены результаты проведенных экспериментов на рис. 4, а также анализ различных методов с точки зрения эффективности, точности и нагрузки на систему.



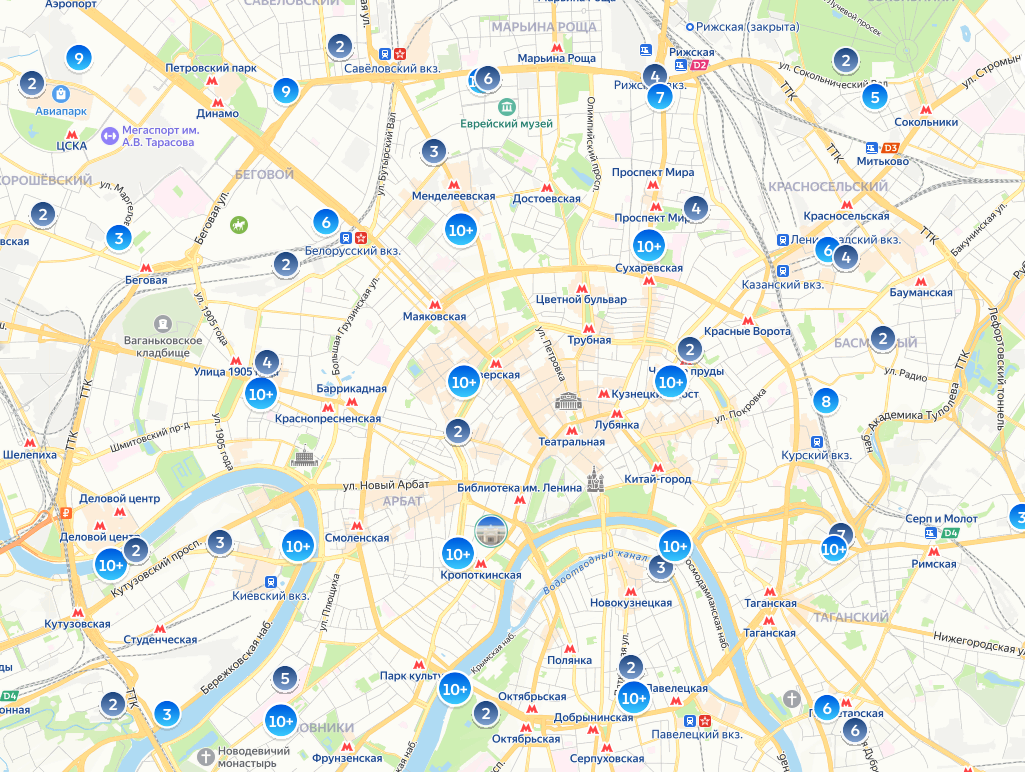
**Рис. 4.** Показатели метрик относительно разных видов Manager

Результаты эксперимента показали, что использование RemoteObjectManager существенно повышает производительность системы при увеличении количества отображаемых объектов на карте. Во всех протестированных сценариях — от небольшого числа сущностей до более нагруженных конфигураций — RemoteObjectManager обеспечил меньшую нагрузку на сеть, более быстрое выполнение JavaScript-кода и сокращенное время рендеринга и отрисовки по сравнению с ObjectManager. При визуальном сравнении на рис. 5 видно, что при использовании ObjectManager, карта становится очень перегруженной, что усложняет поиск объектов, также требуется запрашивать большее количество информации об объектах и данный параметр не является конфигурируемым.



**Рис. 5.** Пример карты с использованием ObjectManager

Используя RemoteObjectManager, на рис 6. видно, что карта стала более читаемой, требуется меньше времени на получение информации, так как сущности объекта не отрисовываются, а самым главным является, то, что если объекты потребуется отрисовать, то данный параметр является конфигурируемым.



**Рис. 6.** Пример карты с использованием RemoteObjectManager

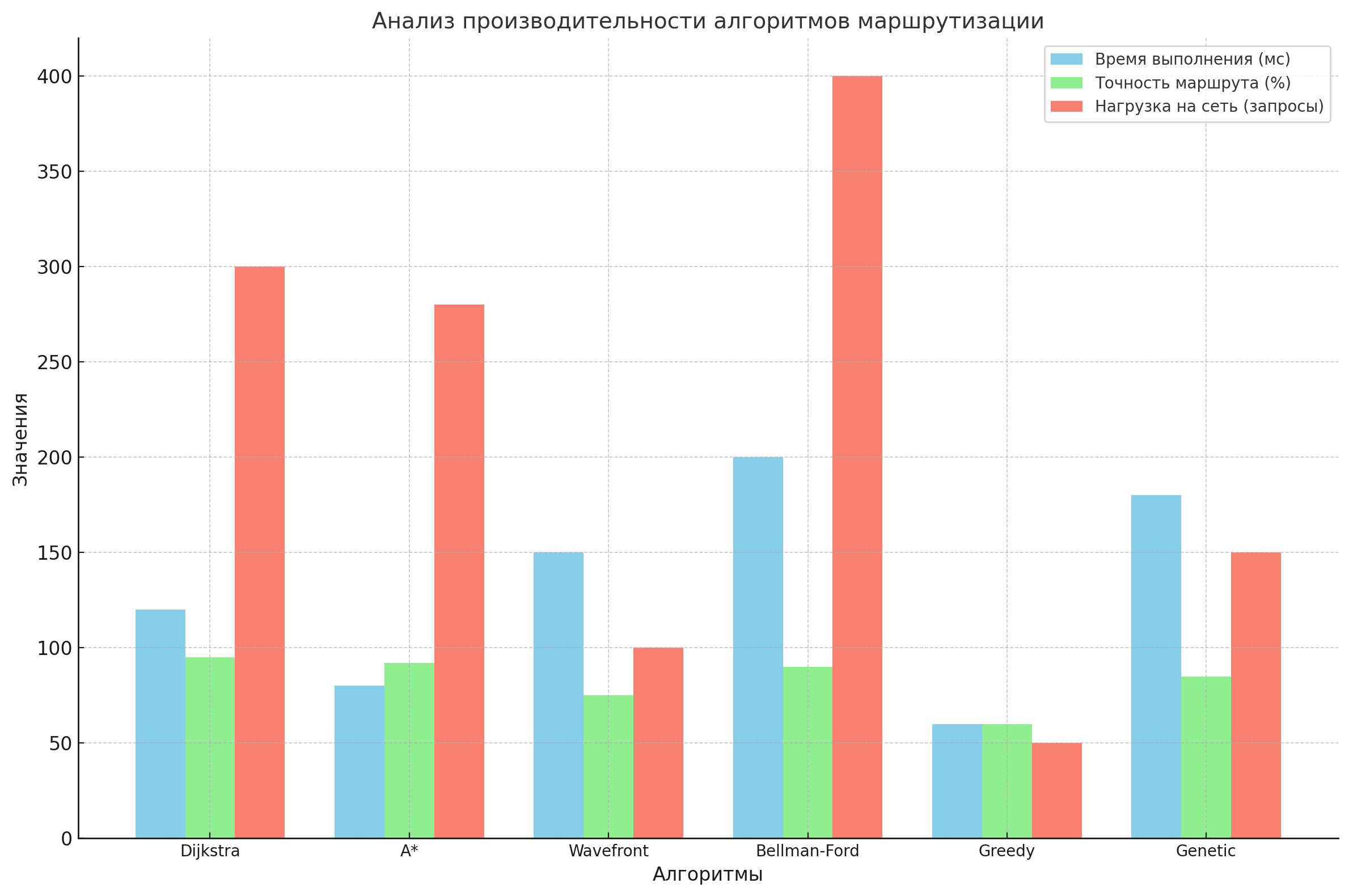
Исходя из всех вышеперечисленных пунктов, улучшения особенно заметны при значительном увеличении количества сущностей, что делает RemoteObjectManager предпочтительным выбором для работы с картами с большим количеством объектов, что видно по табл. 2.

Таблица 2. Сравнительная таблица по выбору менеджера

|  |  |  |
| --- | --- | --- |
| Количество сущностей | Лучший выбор | Причина |
| 100 | RemoteObjectManager | Минимальная нагрузка на сеть, быстрая отрисовка. |
| 500 | RemoteObjectManager | Существенно ниже время выполнения и рендеринга. |
| 1000 | RemoteObjectManager | Лучшая при высокой нагрузке. яц |

Исходя из приведенных экспериментов и сравнений различных менеджеров, удалось понять, какой способ более эффективный и масштабируемый. Существенное снижение времени загрузки сети и оптимизация выполнения клиентского кода при использовании RemoteObjectManager объясняется перераспределением нагрузки на сервер, что минимизирует издержки обработки данных на клиенте.

Для оценки различных методов маршрутизации, описанных выше, был проведен ряд экспериментов, направленных на определение их эффективности, точности и влияния на производительность системы.



**Рис. 7.** Пример карты с использованием RemoteObjectManager

На рисунке 7 представлены результаты анализа производительности при использовании алгоритмов Дейкстры, A\*, поиска волнами, а также жадных и генетических подходов. Основными метриками для сравнения стали:

* Время выполнения алгоритма.
* Загрузка сети.
* Использование ресурсов клиента (время выполнения JavaScript и отрисовка).

На основании проведенных тестов была сформирована сравнительная таблица (Таблица 2), в которой отображены ключевые характеристики методов:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Метод** | **Эффективность (время)** | **Точность (оптимальность)** | **Нагрузка на сеть** | **Примечания** |
| Дейкстра | Средняя | Высокая | Средняя | Подходит для взвешенных графов без отрицательных ребер. |
| A\* | Высокая | Высокая | Средняя | Эвристика значительно ускоряет поиск, особенно в больших графах. |
| Поиск волнами | Низкая | Низкая | Низкая | Эффективен только для графов с равными весами ребер. |
| Беллман-Форд | Низкая | Высокая | Высокая | Единственный метод, работающий с отрицательными весами. |
| Жадные алгоритмы | Высокая | Низкая | Низкая | Быстрый, но может пропустить более оптимальные маршруты. |
| Генетические алгоритмы | Средняя | Средняя | Средняя | Подходит для задач высокой комбинаторной сложности, но требует настройки. |

На основе анализа данных, представленных в таблице, можно сделать следующие выводы:

* Алгоритм Дейкстры обеспечивает высокую точность и подходит для большинства типичных задач маршрутизации, если веса ребер положительные.
* A\* превосходит Дейкстру по времени выполнения за счет использования эвристической оценки, что делает его предпочтительным для больших графов.
* Поиск волнами и жадные алгоритмы менее эффективны в сложных случаях, так как либо игнорируют вес ребер, либо не предоставляют оптимальных решений.
* Беллман-Форд имеет узкоспециализированное применение из-за высокой нагрузки на систему, но может использоваться для графов с отрицательными весами.
* Генетические алгоритмы подходят для случаев, где другие подходы менее эффективны из-за высокой сложности.

Карта, построенная с помощью алгоритма A\*, демонстрирует более плавный и точный маршрут, чем карта, построенная с использованием жадного алгоритма, который избегает рассмотрения альтернативных путей.

Таким образом, выбор оптимального метода маршрутизации зависит от требований конкретной задачи: размера графа, структуры данных и ожидаемой точности.

****Заключение****

В ходе разработки системы «Навигатор» удалось достичь значительных улучшений в решении задач по отображению объектов на карте и построению маршрутов по картографическим данным. Особое внимание уделялось вопросам масштабируемости, точности и производительности системы.

Одной из ключевых задач разработки был поиск эффективного способа отображения большого количества объектов на карте. Это требовало решения проблемы высокой нагрузки на клиентскую часть, которая при традиционных подходах приводила к снижению производительности и увеличению времени отклика системы. Использование серверной реализации RemoteObjectManager стало оптимальным выбором. Этот подход позволил минимизировать объем данных, передаваемых на клиент, перераспределив основную нагрузку на сервер. Благодаря этому система получила возможность работать с тысячами объектов без значительного влияния на производительность и время рендеринга.

Для построения маршрутов в «Навигаторе» был реализован Алгоритм Дейкстры, известный своей эффективностью в решении задач поиска кратчайшего пути. Алгоритм обеспечивает оптимальное распределение вычислительных ресурсов, анализируя только те участки графа, которые напрямую влияют на результат. Кроме того, его способность к адаптации под динамические графы, такие как дорожные сети с обновляемыми данными о пробках, делает его особенно подходящим для работы в условиях реального времени.

Помимо вышеуказанных аспектов, разработка включала анализ и сравнение альтернативных подходов, таких как использование API Яндекс Карт, OpenLayers и OpenStreetMap, что позволило оценить их сильные и слабые стороны в контексте задачи. Это позволило выбрать наиболее эффективные инструменты для реализации конкретных функций системы.

Результаты работы показали, что выбранные методы и технологии способны удовлетворить как текущие, так и потенциальные требования к системе. Применение инновационных решений, таких как RemoteObjectManager и адаптированные алгоритмы маршрутизации, обеспечило высокую производительность и масштабируемость системы «Навигатор», делая её перспективным инструментом для использования в проектах, требующих работы с большими объемами геопространственных данных.

Список литературы

1. Калашников, А. Д. Многофункциональная информационная система эволюции развития и современного состояния ракетно-космической отрасли / А. Д. Калашников, Д. А. Ткачева, Е. И. Жидков // Искусственный интеллект в автоматизированных системах управления и обработки данных : Сборник статей II Всероссийской научной конференции: в 5 томах, Москва, 27–28 апреля 2023 года. – Москва: КДУ, Добросвет, 2023. – С. 387-391. – EDN OFKFIE.
2. Многофункциональная интеллектуальная информационная система эволюции развития и современного состояния ракетно-космической отрасли / А. Д. Калашников, Д. А. Ткачева, Е. Д. Вдовкина [и др.] // XLVII Академические чтения по космонавтике 2023 : Сборник тезисов, посвященный памяти академика С.П. Королёва и других выдающихся отечественных ученых — пионеров освоения космического пространства, Москва, 24–27 января 2023 года. Том 1. – Москва: Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет), 2023. – С. 32-34. – EDN DFLFQF.
3. Официальный сайт Yandex API. URL: <https://yandex.ru/dev/jsapi-v2-1/doc/ru/> (Дата обращения 14.11.2024)
4. RFC 7946. Cтандарт GeoJSON. URL: <https://datatracker.ietf.org/doc/html/rfc7946> (Дата обращения 14.11.2024).
5. Официальный сайт документации Redis. URL: <https://redis.io/docs/latest/> (Дата обращения 14.11.2024)
6. D. Deng, "DBSCAN Clustering Algorithm Based on Density," 2020 7th International Forum on Electrical Engineering and Automation (IFEEA), Hefei, China, 2020, pp. 949-953, doi: 10.1109/IFEEA51475.2020.00199.
7. Юрдамян, Н. А. Сравнение алгоритмов поиска кратчайшего пути Дейкстры и A / Н. А. Юрдамян // Наука 2020. Теория и практика : материалы II международной научно-практической конференции, Екатеринбург, 02 апреля 2020 года. – Саратов: Общество с ограниченной ответственностью "Центр профессионального менеджмента "Академия Бизнеса", 2020. – С. 132-134. – EDN TQYXJS.