УДК 004.056.5

Разработка программного обеспечения для формирования оптимальных маршрутов быстрой доставки товаров

С.А. Макогон, О.Д. Ситникова
Донецкий национальный технический университет
makogon_94@mail.ru, olga_d_s@mail.ru

Макогон С.А., Ситникова О.Д. *Разработка программного обеспечения для формирования оптимальных маршрутов быстрой доставки товаров*

В данной работе была рассмотрена задача построения оптимального маршрута быстрой доставки товаров, существующие программные комплексы, предоставляющие подобный функционал на коммерческой основе, приведено описание проектированияи реализации собственной системы, предоставляющей функционал, необходимый для построения оптимального маршрута быстрой доставки товаров с использованием модифицированного алгоритма муравьиного поиска.

Введение

В современном мире создаются все условия, необходимые для комфортной и как можно менее энергозатратной деятельности человека. Создаются различные устройства, которые минимизируют участие человека практически во всех отраслях производства. Однако, очень большое количество разнообразных задач невозможно выполнить без прямого либо косвенного вмешательства человека. Данная проблема обусловлена недостаточным уровнем развития технологий на данном этапе существования человечества. Одной из задач, требующих непосредственно вмешательства человека, является доставка товаров.

Практически все задачи, связанные с доставкой и распределением товаров, являются задачами такого подраздела экономики, как логистика. Именно эта отрасль занимается изучением способов построения как можно более коротких маршрутов для оптимизации работы по распределению, перевозке и доставке товаров.

В математической интерпретации, данные задачи сводятся к решению NP-трудной задачи, которая называется «Задача коммивояжера».

Технологии не стоят на месте, и, поэтому, для построения кратчайших маршрутов доставки создаются различные приложения, обеспечивающие временной расчет своевременного выполнения задания по обеспечению потребителей необходимыми товарами.

Постановка задачи

Необходимо спроектировать и реализовать программное обеспечение, реализующее основные функции по поиску оптимального кратчайшего маршрута для быстрой доставки товара.

Торговая сеть большого города включает N магазинов (торговых центров), в каждом из которых есть кулинарный отдел, предоставляющий услугу доставки горячих обедов по заявкам с учетом меню. Услуга пользуется спросом, количество заявок очень большое. Особенность услуги в том, что она должна быть выполнена в кратчайший срок за время не более T минут.

В каждом торговом центре имеется Ki курьеров. Курьер выезжает на доставку сразу по нескольким адресам. Распределением заказов занимается один диспетчер сети. Нужно помочь ему сформировать маршруты для курьеров с минимизацией расходов на все маршруты и при ограничении времени доставки: каждый курьер должен быть в пути не более T минут. Также каждый курьер не может обслужить более Z заказов в одном маршруте. Все заказы должны быть обслужены в обязательном порядке. Известны адреса заявок. Все расстояния известны, также, известно ориентировочное время в пути между любыми точками. Загрузка всех курьеров не обязательна, критерием является минимизация суммарного расстояния (то есть затраты на бензин) при ограничении на время по маршруту.

Основная цель — спроектировать рабочую систему для построения оптимального маршрута быстрой доставки товаров с использованием технологий, предоставляемых картографическими

API. Изучить и использовать алгоритмы для поиска кратчайших путей, закрепить и улучшить навыки анализа, проектирования и конструирования программного обеспечения.

Освоить базовые понятия и функционал HTML, Javascript, CSS и развить практические навыки работы с данными языками. Получить практические навыки, связанные с использованием возможностей картографических API, в частности Google Maps API, Open Street Maps API и GraphHopper Route Optimization API.

Одним из главных требований к продукту выступает модульность. Приложение должно быть расширяемым – иметь возможность изменить или увеличить часть функционала системы без необходимости переписывания кода всей системы или замены архитектуры.

Актуальность работы обусловлена востребованностью подобных систем на рынке, потребностью в создании открытой системы, реализующей основные функции, с учетом преимуществ и недостатков существующих аналогов, а также личной необходимостью.

Структура программной системы

Система должна иметь модульную структуру (разделяться на пакеты/модули) для обеспечения расширяемости системы без необходимости редактирования существующего кода, а также с целью возможности повторного использования кода.

Для обеспечения должного уровня надежности функционирования программное изделие должно:

− минимизировать количество действий, необходимое для выполнения операции;

− уменьшить время ожидания отклика приложения, либо сообщать о проведении длительной операции;

− проводить проверку пользовательского ввода на клиентской и серверной части;

− обеспечить обработку исключительных ситуаций, вызванных ошибочными действиями пользователей с целью предотвращения некорректного завершения работы приложения.

Система должна иметь постоянный доступ к сети Интернет для полного и бесперебойного функционирования и выполнения поставленных задач.

На рисунке 1 приводится диаграмма, иллюстрирующая взаимосвязь модулей программного продукта.

Диаграмма вариантов использования приложения пользователем предоставлена на рисунке 2.

На рисунке 3 изображена диаграмма последовательности для запроса на сервер Google Maps API.

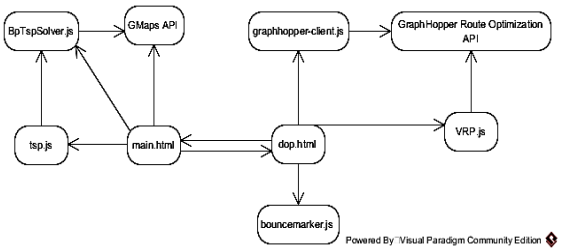

Рисунок 1 — Схема связи модулей приложения

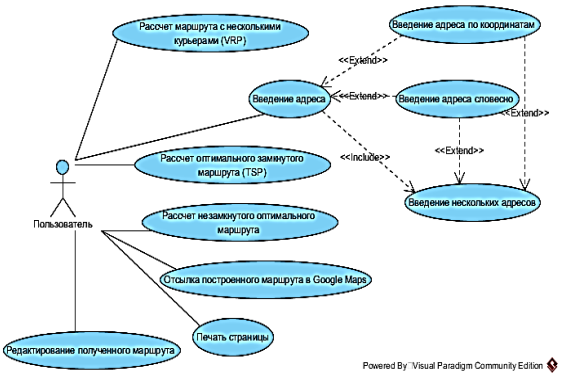

Рисунок 2 – Диаграмма вариантов использования

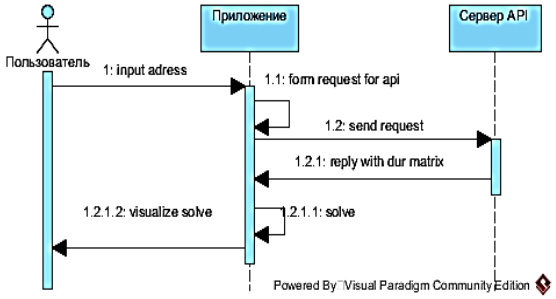

Рисунок 3 – Диаграмма последовательности для запроса на сервер GMaps API

Эта диаграмма наглядно иллюстрирует последовательность действий программного продукта под влиянием запроса со стороны пользователя.

Изначально пользователь производит ввод необходимых данных в форму, программа получает эти данные в «сыром» виде и преобразовывает их в формат, необходимый для корректного выполнения пользовательского запроса на сервер с API. Сервер обрабатывает полученный им запрос и выдает результат, необходимый программному комплексу для решения задачи. После получения ответа, приложение производит необходимые вычисления и предоставляет полученный результат пользователю.

Проектирование структур системы

Ввиду необходимости использования сторонних API для получения необходимых для вычисления данных, следует подготовить почву для формирования корректного формата запросов и создать локально необходимые типы структур. Это становится возможным благодаря наличию документации использованных в системе API.

Для осуществления работы с объектом карт необходимо изначально его создать в контексте программы. Происходит это путем создания какого-либо контейнера на заглавной странице и функции инициализации карты. На рисунке 4 изображен фрагмент кода, выполняющего данное действие.

```
function initMap(center, zoom, div) {
    var myOptions = {
        zoom: zoom,
        center: center,
        mapTypeId: google.maps.MapTypeId.ROADMAP
    };
    gebMap = new google.maps.Map(div, myOptions);
    google.maps.event.addListener(gebMap, "click", function(event) {
        tsp.addWaypoint(event.latLng, addWaypointSuccessCallback);
    });
}
```

Рисунок 4 – Создание объекта карт

В переменную myOptions заносятся данные о зуммировании карты, ее центровке и устанавливается id типа карты.

После того, как объект карты создан, можно перейти к созданию второго необходимого для работы элемента, а именно — метки. На рисунке 5 изображен фрагмент кода для инициализации пользовательского маркера.

```
var icon;
icon = new google.maps.MarkerImage("newicons/red" + (num + 1) + ".png");
var marker = new google.maps.Marker({
    position: latlng,
    icon: icon,
    map: gebMap
});
```

Рисунок 5 – Инициализация пользовательского маркера

Как видно, создается переменная с названием icon, в которой будет храниться иконка, необходимая для отображения маркера. Создается так же и переменная marker, в которой будет храниться информация о самом маркере в форме, доступной для интерпретации посредством Google Maps API. В переменную position заносятся данные о координатах положения маркера, в переменную icon заносятся данные об отображаемой на месте маркера картинке, в переменную map заносится информация на какую карту будет отображаться заданная метка.

Для осуществления запроса на построение маршрута необходимо создать структуру данных, приведенную на рисунке 6.

Для использования маршрутов создается объект типа DirectionsService и вызывается DirectionsService.route() для инициализации запроса в службу Directions.

```
var myGebDirections = new google.maps.DirectionsService();

myGebDirections.route({
    origin: origin,
    destination: destination,
    waypoints: wayArrChunk2,
    avoidHighways: avoidHighways,
    avoidTolls: avoidTolls,
    unitSystem: directionunits,
    travelMode: travelMode
  },
```

Рисунок 6 – Создание маршрута

Данный объект может содержать следующие поля:

- origin (обязательное) — указывает начальную точку маршрута. Может быть указан в виде строки, значения LatLng (координат) или объекта google.maps.Place.
- destination (обязательное) — указывает конечную точку маршрута. Параметры аналогичны параметрам поля origin.
- travelMode (обязательное) — указывает какой способ передвижения необходимо использовать при построении маршрута
- transitOptions (необязательное) — указывает значения, которые относятся только к тем запросам, где поле travelMode имеет значение TRANSIT.
- drivingOptions (необязательное) — указывает значения, относящиеся только к запросам, где поле travelMode имеет значение DRIVING.
- unitSystem (необязательное) — указывает, какую систему единиц измерения следует использовать для отображения результатов.
- waypoints [] (необязательное) – определяет массив промежуточных точек DirectionsWaypoint.
- optimizeWaypoints (необязательное) — указывает, что маршрут, использующий предоставленные значения waypoints, может быть оптимизирован путем расположения этих промежуточных точек в более эффективном порядке.
- provideRouteAlternatives (необязательное) — если это поле имеет значение true, служба Directions может указывать несколько альтернативных маршрутов в ответе.
- awoidHighways (необязательное) — если это поле имеет значение true, рассчитанный маршрут должен избегать крупных магистралей, если это возможно.
- awoidTolls (необязательное) — если это поле имеет значение true, рассчитанный маршрут должен избегать платных дорог, если это возможно.
- region (необязательное) – код региона, указываемый как значение ccTLD ("домен верхнего уровня") из двух символов[1].

Программная реализация

Для разработки функционала программной системы был выбран мультипарадигменный язык программирования Javascript в связке с HTML и CSS. В качестве интегрированной среды разработки была выбрана программа Notepad++ ввиду ее доступности и открытости для разработчиков.

Язык Javascript является простым, доступным для понимания, кроссплатформенным, мультипарадигмиальным языком программирования. Обычно, данный язык используется в браузерах, как средство для придания интерактивности веб-страницам.

Однако, существуют примеры использования данного языка в браузерных операционных системах, в качестве средства для создания серверных приложений, в прикладном программном обеспечениии даже в офисных приложениях. Ввиду перечисленных особеностей, в последнее время ведутся работы по созданию новых средств для выполнения задач разработки в более масштабном объеме. Основными архитектурными чертами языка являются:

- динамическая типизация;
- слабая типизация;
- автоматическое управления памятью;
- прототипное программирование;
- функции как объекты первого класса.

При разработке Javascript на него оказали влияние множество языков. Одним из главных критериев реализации была цель сделать язык как можно более похожим на Java, но более легким для использования даже рядовыми пользователями[2].

Для разработки программной системы был выбран именно этот язык программирования ввиду указанных выше преимуществ, возможности получить доступ к различным веб-ориентарованным системам и личным интересом в изучении возможностей данного языка как для использования в веб-приложениях, так и в качестве работы разрабатываемой системы на стороне пользователя. Так же, приятным дополнением ко всему перечисленному является и то, что для работы с языком Javascript не требуется установка особых интегрированных сред разработки, так как код может выполняться в браузере, а для его написания и редактирования достаточно наличия простого текстового редактора.

Для реализации функционала программного обеспечения использовались следующие библиотеки и API:

- библиотека jQuery;
- Google Maps API;
- OpenStreetMaps API;
- GraphHopper API.

В ходе разработки были созданы собственные библиотеки, написанные с использованием языка Javascript, а именно такие:

- TSPSolver;
- TSP.

Данные средства были выбраны ввиду их доступности, открытости, наличию обширной документации, отзывчивому и приятному коммьюнити разработчиков, которые активно поддерживаются создателями данных средств и предоставляют широкий спектр услуг и возможностей для использования и дальнейшего развития предлагаемых продуктов.

Однако, у выбранных средств имеются и недостатки, которые выясняются при углубленном их изучении. Так, Google Maps API предоставляют услуги по своей работе как на платной, так и на бесплатной основе. При использовании на бесплатной основе существуют весьма существенные ограничения, установленные корпорацией Google на использование картографических API. Данные ограничения заключаются в максимальном количестве ежедневных загрузок карты, установленных в размере 25000, и выполнении запросов к сервисам геолокации, геокодирования, установленных в размере 2500 в день[3].

Основной функцией приложения является предоставление пользователю возможности по построению оптимальных маршрутов быстрой доставки товаров.

Пользовательский интерфейс реализован путем осуществления перехода между между различными операциями. Данные для отображения отрисовываются при загрузке приложения в реальном времени.

Разрабатываемое приложение имеет два модуля, в которых реализован основной функционал. Эти модули представлены статичными HTML-страницами с использованием подключенных в зависимостях исполняемых файлах Javascript и таблиц стилей.

Логика работы данных модулей зависит от действий, выполняемых пользователем во время пребывания на данных страницах.

При разработке методов решения задачи были использованы алгоритм муравьиной коллонии, 2-opt алгоритм и жадный алгоритм. Описание данных алгоритмов будет приведено ниже.

Использованные алгоритмы

Муравьиный алгоритм — один из эффективных полиномиальных алгоритмов для нахождения приближенных решений задачи коммивояжера и решения аналогичных задач поиска маршрутов на графах. Суть метода заключается в исследовании и воспроизведении поведения таких социальных насекомых, как

муравьи, которые ищут пути от колонии к источнику питания. Данный метод представляет собой метаэвристическую оптимизацию[4].

В основе алгоритма заключен метод поиска путей муравьями, а именно маркировка более удачного маршрута большим количеством феромона.

Любой муравьиный алгоритм, вне зависимости от возможных вариантов модификаций, представим в следующем виде:

0. Проверка условия выхода. Если проверка не пройдена, то переход на шаг 2.

1. Создаем муравьев.

Начальная точка, куда помещается муравей, зависит от ограничений, которые были заданы по условию задачи. На данном этапе так же задается начальный уровень феромона, который инициализируется небольшим положительным числом для того, чтобы на начальном шаге вероятности перехода в следующую вершину не были нулевыми.

2. Ищем решения.

Вероятность перехода из вершины i в вершину j определяется по следующей формуле (5.1):

$$P_{ij}(t) = \frac{\tau_{ij}(t)^{\alpha}\left(\frac{1}{d_{ij}}\right)^{\beta}}{\sum_{j \in allowed\ nodes} \tau_{ij}(t)^{\alpha}\left(\frac{1}{d_{ij}}\right)^{\beta}} \cdot , \quad (5.1)$$

где τij(t) — уровень феромона,

dij — эвристическое расстояние,

α, β — константные параметры.

При α = 0 выбор ближайшего города наиболее вероятен, то есть алгоритм становится жадным.

При β = 0 выбор происходит только на основании феромона, что приводит к субоптимальным решениям.

3. Обновляем феромон.

Уровень феромона обновляется в соответствии с приведённой формулой (5.2)

$$\tau_{ij}(t+1) = (1-p)\tau_{ij}(t) + \sum_{k \in Colony\ that\ used\ edge(i,j)} \frac{Q}{L_k}, \quad (5.2)$$

где p – интенсивность испарения,

Lk – цена текущего решения для k-ого муравья,

Q – параметр, имеющий значение порядка цены оптимального решения.

4. Дополнительные действия (опционально).

Обычно здесь используется алгоритм локального поиска, однако, он может также появиться и после поиска всех решений.

Алгоритм представлен на рисунке 7.

Рисунок 7 – Общий алгоритм муравьиной колонии

Для варианта вычисления кольцевого замкнутого маршрута в приложении использовался алгоритм, включающий в свою работу дополнительный пункт в виде преобразование полученного пути с помощью алгоритма 2-opt.

Суть алгоритма 2-opt заключается в выборе 2 дуг построенного маршрута и их переброски. Если длина полученного пути будет короче, чем длина пути, признанного оптимальным, то новый путь признается оптимальным и записывается.

DirectionsService и вызывается DirectionsService.route() для инициализации запроса в службу Directions

Также, была создана функция, использующая метод жадного алгоритма.

Жадным алгоритмом называется алгоритм, который на каждом своем шаге принимает локально оптимальный выбор, допуская, что итоговое решение также окажется оптимальным. Следует заметить, что далеко не всегда это оказывается верным подходом к решению задачи. Для большого количества алгоритмических задач решение с помощью жадного алгоритма является приемлемым.

Пример использования системы

Данный программный продукт представляет собой модульную систему, реализующую функции поиска кратчайших путей для быстрой доставки товаров. Особенностью является то, что данный продукт можно использовать в качестве встраиваемого модуля в более сложных логистических системах. Стартовый экран представлен на рисунке 8.

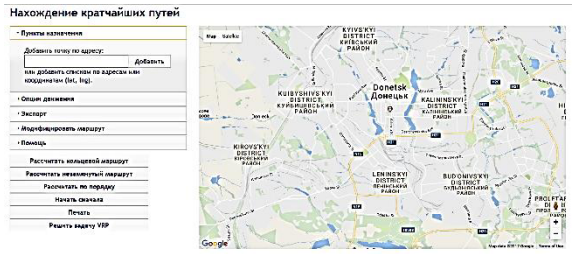

Рисунок 8 – Стартовый (главный) экран приложения

Далее, с главной страницы пользователь получает доступ уже ко всем возможным вариантам использования программной системы. На рисунке 9 показан вариант добавления маркера на карту путем клика на нее

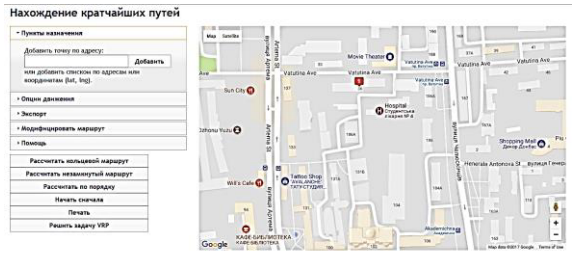

Рисунок 9 – Добавление маркера по клику на карте

При построении кольцевого маршрута заданная точка будет выступать в качестве начально и конечной точки одновременно. На рисунке 10 показан построенный кольцевой маршрут

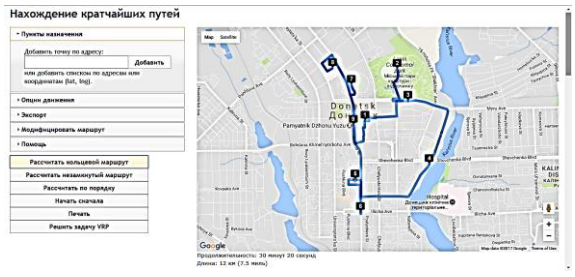

Рисунок 10 – Построенный кольцевой маршрут

Так же на рисунке 11 отображена часть словесных описаний, доступных для рассчитанного маршрута

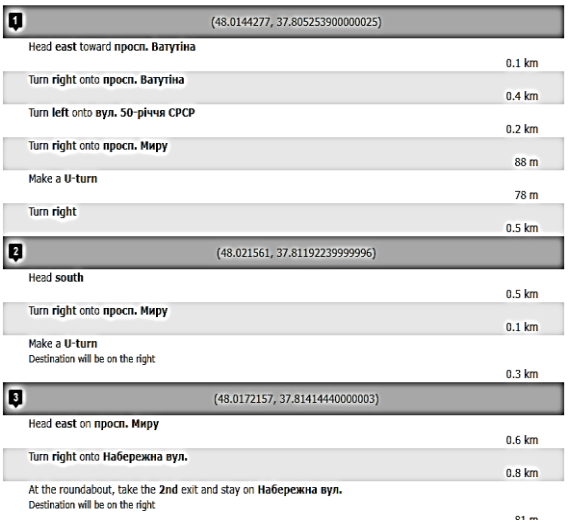

Рисунок 11 – Пример словесного описания построенного маршрута

При нажатии на кнопку «Начать сначала» состояни окна сбросится и вернется к начальному варианту, который был предоставлен на рисунке 8.

При вводе адреса словесно в текстовое поле программа преобразует этот текст в точное географическое название и отметит его на карте. На рисунке 12 показана возможность добавления адреса пользователем с использованием методов геолокации.

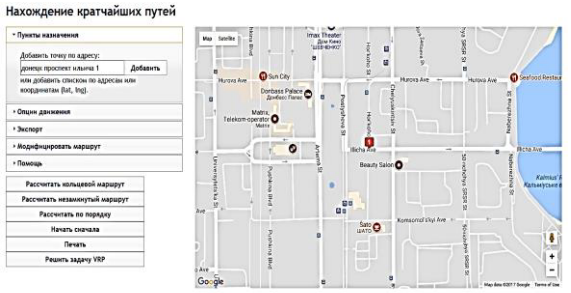

Рисунок 12 – Добавление словесного адреса на карту

При желании пользователя добавить несколько точек на карту, ему будет необходимо нажать на ссылку с текстом «добавить списком по адресам или координатам (lat, lng)».

На рисунке 13 показана возможность добавления нескольких адресов использованием методов прямой и обратной геолокации.

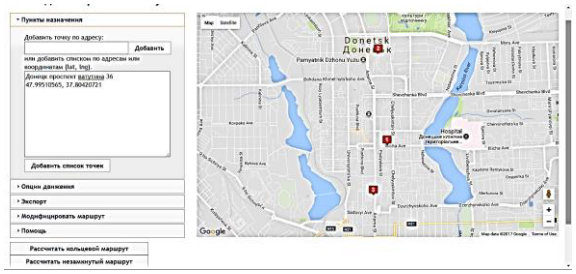

Рисунок 13 – Добавление списка адресов

При построении желаемого маршрута пользователь может указать дополнительные опции маршрута, которые представлены на рисунке 14.

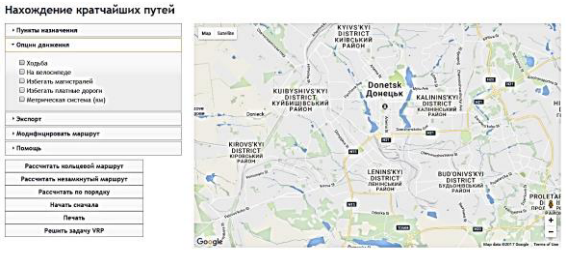

Рисунок 14 – Дополнительные опции построения маршрута

После успешного построения любого из маршрутов пользователю становятся доступны функции экспорта маршрута любым из перечисленных вариантов и его модификация, путем изменения порядка следования на желаемый.

При нажатии на кнопку «Решить задачу VRP» пользователь будет перенаправлен на другую форму, представляющую аналогичный функционал. На этой форме возможно выполнение построения маршрутов для нескольких курьеров с оптимизацией по загруженности всех доступных курьеров и минимизации их пути. под картой отображается суммарная длина всех найденных путей и время, затрачиваемое на выполнение всех заказов, с учетом того, что возможно один курьер будет выполнять все заказы. Форма приведена на рисунке 15.

Тестирование

При разработке данного программного обеспечения было использовано тестирование юзабилити.

Юзабилити-тестирование — это исследование, выполняемое с целью определения, удобен ли некоторый искусственный объект (такой как веб-страница, пользовательский интерфейс или устройство) для его предполагаемого применения[5].

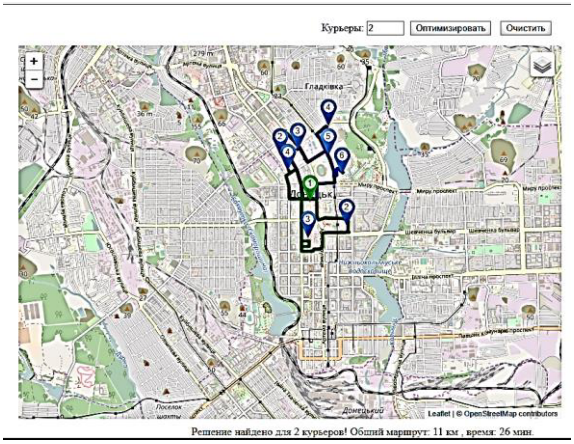

Рисунок 15 – Форма решения задачи VRP

Проверка эргономичности — метод оценки удобства продукта в использовании, основанный на привлечении пользователей в качестве тестировщиков, испытателей и суммировании полученных от них выводов.

При разработке данного программного продукта был использован непрямой метод тестирования. Была собрана фокус-группа (5 человек) экспертов в предметной области. Каждому эксперту было установлено данное приложение. Взаимодействие между экспертом и приложением протоколировалось. Результаты показаны в таблице 1.

Таблица 1 – Результаты юзабилити-тестирования

Результативность	Эффективность	Удовлетворенность
90 %	30%	+
90%	30%	+
100%	30%	+
90%	30%	+
95%	30%	+
85%	45%	+-

В результате тестирования, фокус-группа определила, что приложение является интуитивно-понятным и эффективным. Последний респондент остался не слишком доволен скоростью работы программы и дизайном интерфейса

Выводы

В ходе выполнения работы было рассмотрена транспортная задача по определению кратчайших маршруов доставки товаров. На основании проведенного анализа существующих аналогов были поставлены задачи.

Разработано программное обеспечение, предназначенное для формирования оптимальных маршрутов для быстрой доставки товаров. Данный програмный продукт был написан с использованием языков HTML, CSS и Javascript и может быть использован как в качестве

самостоятельного ПО, так и в качестве встраиваемого модуля в системы типа 1С.

В дальнейшем планируется улучшать и дорабатывать систему.

Литература

1. JavaScript API - Google Maps JavaScript API | Google Developers [Электронный ресурс] // Google. – Режим доступа: https://developers.google.com/maps/documentation/javascript/?hl=ru. – Загл. с экрана|.

2. JavaScript [Электронный ресурс] // Википедия. – Режим доступа: https://ru.wikipedia.org/wiki/JavaScript. – Загл. с экрана|.

3. Цены и планы | Цены и планы для Google Maps API | Google Developers [Электронный ресурс] // Google. – Режим доступа: https://developers.google.com/maps/pricing-and-plans/?hl=ru. – Загл. с экрана|.

4. Муравьиный алгоритм [Электронный ресурс] // YouTube. – Режим доступа: https://www.youtube.com/watch?v=EwDP_bAb-OI. – Загл. с экрана|.

5. Тестирование юзабилити [Электронный ресурс] // artw. – Режим доступа: https://artw.ru/blog/archives/3537/. – Загл. с экрана|.

Макогон С.А., Ситникова О.Д. Разработка программного обеспечения для формирования оптимальных маршрутов быстрой доставки товаров*.В данной работе была рассмотрена задача построения оптимального маршрута быстрой доставки товаров, существующие программные комплексы, предоставляющие подобный функционал на коммерческой основе, приведено описание проектированияи реализации собственной системы, предоставляющей функционал, необходимый для построения оптимального маршрута быстрой доставки товаров с использованием модифицированного алгоритма муравьиного поиска*

Makogon S.A., Sitnikova O.D. Developing of software for constructing optimal routes for the rapid delivery of goods". *In this paper was considered the problem of constructing an optimal route for the rapid delivery of goods, existing software complexes that provide such functionality on a commercial basis, describe the design and implementation of its own system that provides the functionality necessary for constructing the optimal route for the rapid delivery of goods using a modified ant colony optimization algorithm.*

Статья поступила в редакцию 7.10.2017
Рекомендована к публикации доктором технических наук В.Н. Павлышом