Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ КОСМИЧЕСКИХ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Кафедра «Информатика»

УТВЕРЖДАЮ
Заведующий кафедрой
_____ А.И. Рубан
«16» июня 2016 г.

МАГИСТРСКАЯ ДИССЕРТАЦИЯ

Программная система построения маршрута торгового представителя

27.04.03 «Системный анализ и управление»

27.04.03.02 «Системный анализ данных и технологии принятия решений»

Научный руководитель 15.06.16 доцент К.Т.Н. А.А. Даничев
подпись, дата должность, ученая степень инициалы, фамилия

Выпускник 15.06.2016 А.Е. Бридский
подпись, дата инициалы, фамилия

Рецензент 15.06.2016г директор Ю.В. Политик
подпись, дата должность, ученая степень инициалы, фамилия

Красноярск 2016

РЕФЕРАТ

Выпускная квалификационная работа по теме «Программная система построения маршрутов торгового представителя на базе предприятия ООО «Селект»» содержит 58 страниц текстового документа, 19 использованных источников, 24 иллюстрации, 10 формул.

ТОРГОВЫЙ ПРЕДСТАВИТЕЛЬ (ТП), ТОРГОВАЯ ТОЧКА, ПРОГРАММА, МАРШРУТ, ПУТЬ, КЛИЕНТ, ОТЧЕТНОСТЬ, КАРТА.

Объект изучения – исследование вариантов оптимизации маршрутов торгового представитель.

Целью является разработка, обоснование и реализация программно-алгоритмического комплекса, обеспечивающего формирование ежедневного формирования маршрутного листа, предназначенного для работы торгового представителя.

В результате данной работы была разработана система, позволяющая составлять ежедневный маршрут торгового представителя, с учетом всех влияющих на него факторов, в том числе личных особенностей каждого сотрудника.

В итоге был разработан модуль составления маршрутов, на основании решения транспортной задачи методом коммивояжера.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Проходя практику на предприятии ООО «Селект», возможно сделать выводы, что ежедневный маршрут торгового представителя крайне не оптимален и на развоз продукции и объезд торговых точек затрачивается очень много времени.

В связи с этим было принято решение заняться оптимизацией составления маршрута, так как если его составить корректно, с учетом многих, влияющих на него факторов, то рабочее время возможно будет использовать более эффективно.

Из ранее полученного опыта работы торговым представителем, как раз в сфере обувной косметики, можно сказать, что с ненормированным графиком работы, возможно в конце рабочего дня объехать большее количество «новых» торговых точек, с целью получения большей прибыли, как для себя, так и для компании. Но в связи с крайне не оптимальным графиком можно ничего не успеть.

Опираясь на эти факты, можно сделать выводы, что разработанное программное обеспечение дает не только возможность больше зарабатывать торговому представителю и самой фирме, но и избежать халатности, среди работников, так как начальник уже будет точно знать тот минимум, который его подчиненный должен выполнить за день.

В рамках решения данной проблемы, было принято решение о создании программного продукта, который сможет составлять оптимальный маршрут для торгового представителя.

1 Фирма. Маршруты торговых представителей. Алгоритм коммивояжера

1.1 Описание и работа торгового представителя на предприятии ООО «Селект»

Компания «Селект» основана 1 сентября 2012 года, под руководством Политик Юлии Владимировны. Занимается продажей и распространением товаров народного потребления, таких как обувная косметика (Collonil, Duke of Dubbin и DAMAVIK), репелленты от насекомых, комаров, клещей, мошек и слепней, таких брендов, как OZZ, Kontra, Partizan. С лета 2013 года фирма начала продажу товаров для пикника (бренд Пикничок).

Компания «Селект» стабильно развивается в г. Красноярске, по обувной косметике начала работу с регионами Красноярского края, такими, как Ачинск, Канск, Дивногорск, Сосновоборск, а также ведется работа с крупными торговыми компаниями, которые закупают обувную косметику и репелленты от насекомых в городе Норильске.

Торговый представитель — одна из самых востребованных сегодня профессий на рынке труда. Даже в условиях кризиса они, наряду с мерчендайзерами и супервайзерами, не испытывают недостатка в работе.

Торговый представитель осуществляет продажи и собирает запросы с рынка непосредственно в торговых точках, поэтому крайне важно уметь находить подход к клиентам и продавать правильно.

1.2 Начало работы в ООО «Селект»

Карьера в компании начиналась на этапе Start Up в роли торгового представителя, на тот момент в компании было 5 человек.

Как это обычно бывает, на начальном этапе, все помогали друг другу и занимались все всем, то есть четкого разделения обязанностей не было. После проведения первой презентации, изучения рынка, ежедневного посещения торговых точек, обувных магазинов и магазинов товаров народного потребления и бытовой химии, было принято решение заниматься не только элитной обувной косметикой (Collonil, Duke of Dubbin), а добавить дешевую DAMAVIK.

Как оказалось, дешевым товаром торговать намного легче и для него есть большая целевая аудитория, чем для элитных товаров.

В компании начались структурные изменения, был уволен директор и некоторое время требовалось исполнять его обязанности, так как опыта продаж обувной косметики, на тот момент уже было достаточно.

Требовалось ознакомиться с рабочей документацией, узнать все тонкости и нюансы продаж и начать задумываться об оптимизации. «Как продавать больше и эффективнее?» - был главный вопрос. Проштудировав статьи в интернете, обратившись к старшим товарищам и исходя из собственного опыта, были сделаны соответствующие выводы и решено все привести в порядок, чтобы четко отслеживать места, где есть пробелы, а именно:

Порядок должен быть везде, соответственно для торговых представителей была введена отчетность, по форме компании.

Разработана программа мотивации, для торговых представителей и покупателей.

Проведение ежедневных собраний помогало разобраться в трудностях работы с клиентами, обсудить нюансы и поделиться свежим опытом.

В ходе переговоров, была поставлена главная задача, как сделать рабочий день торгового представителя наиболее продуктивным. Этим и было решено заняться вплотную. Ежедневное составление маршрутов вручную, становилось не очень удобным, так как количество торговых точек с каждым днем росло.

С приходом нового директора, порядки в фирме немного поменялись, он привел свою команду, ввел систему штрафов и держал всех в «ежовых рукавицах». Обязанности поменялись, и должность с обычного торгового представителя тоже изменилась на супервайзера, в обязанности которого входило контролировать закупки, анализировать продажи и общаться с новыми клиентами, на предмет составления договора и условий сотрудничества.

К лету 2013 года, компания расширила свой ассортимент сезонными товарами от комаров и насекомых, а также товарами для отдыха и пикника, соответственно появилась возможность «заходить» в мясные лавки, крупные и мелкие магазины бытовой химии, с богатым ассортиментом продукции, что обеспечивало лояльность покупателей.

К середине июля клиентская база практически полностью сформирована, как оказалось у нас не такой уж и большой город, даже маленькие торговые точки могут быть огромной сетью. Появились клиенты, которых нужно было посещать раз, два в неделю, которых нужно посещать раз в две недели и сформировался график посещения торговых точек, в том числе и по времени.

Одной из целей для торгового представителя было, по-прежнему, заключение договоров с новыми торговыми точками. Актуальная клиентская база составляла порядка 50-55 клиентов. С каждым днем составлять оптимальный маршрут было сложнее, поэтому в рамках летней производственной практики, было принято решение о составлении программного продукта, который оптимально составляет маршрут, именно этим и было решено заняться в рамках данной бакалаврской работы.

1.3 Основные понятия теории графов. Определения

Граф G задается множеством точек или вершин x_1, x_2, …x_n (которое обозначается через X) и множеством линий или ребер a_1, a_2, …a_m (которое обозначается символом A), соединяющих между собой все или часть этих точек. Таким образом, граф G полностью задается парой (X, A).

Если ребра из множества A ориентированы, что обычно показывается стрелкой, то они называются дугами, и граф с такими ребрами называют ориентированным графом. Другое, употребляемое чаще описание ориентированного графа G состоит в задании множества вершин X и соответствия Г, которое показывает, как между собой связаны вершины. Граф в этом случае обозначается парой G=(X, Г).

Путем (или ориентированным маршрутом) ориентированного графа называется последовательность дуг, в которой конечная вершина всякой дуги, отличной от последней, является начальной вершиной следующей.

Ориентированной цепью называется такой путь, в котором каждая дуга используется не больше одного раза. Простой цепью называется такой путь, в котором каждая вершина используется не более одного раза. Очевидно, что простая орцепь является также орцепью, но обратное уже неверно.

Иногда дугам графа G сопоставляются (приписываются) числа – дуге (x_i, x_j) ставится в соответствие некоторое число $c_{i\ j}$, называемое весом, или длинной. Тогда граф G называется графом со взвешенными дугами. Иногда веса приписываются вершинам x_i графа, и тогда получается граф со взвешенными вершинами. Если в графе веса приписаны и дугам, и вершинам, то он называется просто взвешенным. При рассмотрении пути μ ,

представленного последовательностью дуг, за его вес (или длину) принимается число $l(\mu)$, равное сумме весов всех дуг, входящих в μ, т.е. $l(\mu) = \sum_{(x_i, x_j) \in \mu} c_{ij}$.

Гамильтонов цикл в орграфе – это ориентированный цикл (контур), проходящий ровно один раз через каждую вершину графа (т.е. простая орцепь).

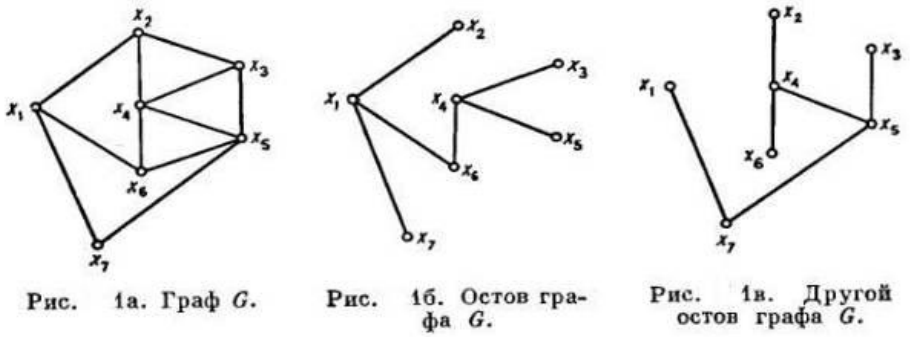


Рис. 1а. Граф *G*. Рис. 1б. Остов графа *G*. Рис. 1в. Другой остов графа *G*.

Рисунок. 1.3.1 - Графы

Если G = (X, A) – неориентированный граф с n вершинами, то остовным деревом (или, короче, остовом) графа G называется всякий остовный подграф G, являющийся деревом (т.е. графом не имеющим циклов, в котором каждая пара вершин соединена одной и только одной простой цепью). Например, если G – граф, показанный на рис. 1а, то графы на рис. 1.б, в являются остовом этого графа.

1.4 Постановка задачи коммивояжера

Постановка задачи. Коммивояжер(бродячий торговец) должен выйти из первого города, посетить по разу в неизвестном порядке города 2,3,4…n и вернуться в первый город. Расстояния между всеми городами известны. В каком порядке следует обходить города, чтобы замкнутый путь коммивояжера

был кратчайшим? В терминах теории графов: найти гамильтонов цикл в графе минимальной длины.

Знаменитая задача коммивояжера, поставленная еще в 1934 г., является одной из самых важнейших задач в теории графов. В своей области (оптимизации дискретных задач) задача коммивояжера служит своеобразным полигоном, на котором испытываются все новые методы.

Задача коммивояжера является так называемой NP-трудной задачей, т.е. задачей, точное решение которой в общем случае может быть получено только за экспоненциальное время. Следовательно, решать ее переборным алгоритмом неэффективно при большом количестве вершин графа.

Одним из подходов к ее решению является сокращение перебора методом ветвей и границ. Этот метод позволяет опознать бесперспективные частичные решения, в результате чего от дерева поиска на одном шаге отсекается целая ветвь. Тем не менее, удовлетворительных оценок быстродействию алгоритма Литтла, основанного на этом методе, и родственных алгоритмов нет, хотя практика показывает, что на современных ЭВМ они иногда позволяют решить задачу коммивояжера для графов с количеством вершин, меньшим 100.

Впервые метод ветвей и границ был предложен Лендом и Дойгом в 1960 для решения общей задачи целочисленного линейного программирования. Интерес к этому методу и фактически его «второе рождение» связано с работой Литтла, Мурти, Суини и Кэрела, посвященной задаче коммивояжера. Начиная с этого момента, появилось большое число работ, посвященных методу ветвей и границ и различным его модификациям. Столь большой успех объясняется тем, что авторы первыми обратили внимание на широту возможностей метода, отметили важность использования специфики задачи и сами воспользовались спецификой задачи коммивояжера. В основе метода ветвей и границ лежит

идея последовательного разбиения множества допустимых решений на подмножества (стратегия «разделяй и властвуй»). На каждом шаге метода элементы разбиения подвергаются проверке для выяснения, содержит данное подмножество оптимальное решение или нет. В качестве примера конкретного применения метода может быть предложена прикладная задача, связанная с проблемой размещения и обслуживания оборудования, в которой требуется определить оптимальную траекторию циклического маршрута движения робота-транспортера по траектории цеха с целью периодического обслуживания оборудования.

1.5 Решение обобщенной задачи Коммивояжера

Пусть каждой дуге (i, j) графа (I, U) поставлено в соответствие число l_{ij} называемое длиной дуги.

Рассмотрим задачу: определить кратчайший путь из множества A в множество B, пересекающий каждое множество разбиения один и только один раз. Очевидно, что если каждое множество разбиения состоит из одного элемента и A=B=$\{a\}$, то имеем обычную задачу коммивояжера.

Определим функцию $f_{i_m}(\mu_{i_m}), i_m \in I$: положим для произвольного пути $\mu_{i_m} = (i_0, i_1, ..., i_k, ..., i_m)\ f_{i_m} = \bigcup_{k=0}^{m} \{\lambda(i_k)\}$. Итак, значениями функции $f_i(\mu_i)$ будут множества номеров подмножеств разбиения, которые пересекает путь μ_i . Пусть каждое множество Φ_i, $i \in I$, состоит из всевозможных подмножеств множества $\{1, 2, \ldots, p\}$, а $\Phi_j^* = \{1, 2, ..., p\}$, где $j \in B$. Применим для решения этой задачи следующий алгоритм.

Достаточной системой функций в данном случае будут функции

$$F_i^1(\mu(N_i)) = f_i(\mu(N_i)), \_\_ L^1(\mu(N_i)) = l(\mu(N_i))$$

Обозначим через $|A|$ число элементов произвольного конечного множества A.

Шаг 0. Положим $\tilde{l} = \min_{i \in A} \min_j l_{ij}, \tilde{k}_i = 0, i \in I$. Пометим вершины $i \in A$ признаками $N_i = (1,0,0,), f_i = \{\lambda(i)\}, \alpha_i = 0.$. Для помеченных вершин увеличим $\tilde{k}$ на 1. Рассмотрим одну из пометок и перейдем к шагу 1.

Шаг 1. Пусть N_i, f_i, α_i – рассматриваемая пометка. Пометим признаками $N_i = (\tilde{k}+1, i, N_i^1), f_j = f_i + \{\lambda(i)\}, \alpha_j = \alpha_j + l_{ij}$ все те вершины, для которых $\alpha_i + l_{ij} = \tilde{l}, \lambda(j) = \lambda(i) \_ или \_ \lambda(j) \in f_i$. Для вновь помеченных вершин увеличим $\tilde{k}_i$ на 1.

Рассмотрим следующую помеченную вершину, и будем повторять помечивания до тех пор, пока не пометим некоторую вершину $j_r \in B$ так, чтобы для признака f_{i_r} пометки этой вершины выполнялось условие $|f_{i_r}| = p$ или пока нельзя будет сделать дальнейших пометок. В первом случае перейдем к шагу 2. а во втором к шагу 3.

Шаг 2. Строим кратчайший допустимый путь от вершины j_r. Пусть N_{j_r} – пометка вершины, для которой $|f_{i_r}| = p$. Перейдём к вершине $j_{r-1} = N_{j_r}^2$ и рассмотрим пометку $N_{j_{r-1}}$ вершины, для которой $N_{j_{r-1}}^1 = N_{j_r}^3$. Далее перейдем к вершине $j_{r-2} = N_{j_{r-1}}^2$, с пометкой $N_{j_{r-2}}$, для которой $N_{j_{r-2}}^1 = N_{j_{r-1}}^3$. Последовательность $(j_0, j_1, ..., j_r)$ и является кратчайшим допустимым путем.

Шаг 3. Пусть $\tilde{I}$ – множество помеченных вершин. Рассмотрим все возможные числа $\alpha_i + l_{ij} > \tilde{l}$ при $j \in I$. Определим среди этих чисел наименьшее и

возьмем его за новое приближение $\tilde{l}$ к длине искомого пути. Затем перейдем к шагу 1.

Этот алгоритм можно изменить. Если для некоторой пометки N_i, f_i, α_i $\alpha + l_{ij} \le \tilde{l}$ при всех j, для которых $\lambda(j) = \lambda(i)$ или $\lambda(j) \in f_i$, то путь соответствующий этой пометке, уже продлен во все смежные с i вершины. Следовательно, для таких пометок признаки f_i, α_i можно стирать.

1.6 Обзор методов решения задачи коммивояжера

Методы, предложенные для поиска кратчайших гамильтоновых циклов – алгебраический метод, основанный на работе Йоу, Даниэльсона и Дхавана (включает в себя построение всех простых цепей с помощью последовательного перемножения матриц), метод перебора Робертса и Флореса (метод перебора имеет дело с одной цепью, непрерывно продлеваемой вплоть до момента, когда либо становится ясно, что эта цепь не может привести к гамильтонову циклу. Тогда цепь модифицируется некоторым систематическим способом, после чего продолжается поиск гамильтонова цикла. Другой подход – мультицепной метод, предложенный Селби.

1.7 Метод ветвей и границ (МВГ)

Метод ветвей и границ ("поиск с возвратом", "backtracking") для решения задачи о коммивояжере – один из наиболее эффективных и быстрых методов решения задачи о коммивояжере, был разработан Литтлом, Мерти, Суини, Кэрелом в 1963 г. Представляет собой итеративную схему неявного (улучшенного) перебора, который состоит в отбрасывании заведомо неоптимальных решений и является одной из самых эффективных процедур в группе методов ветвей и границ.

Идея метода. Пусть $S(0)$ – множество всех допустимых замкнутых маршрутов (циклов) задачи о коммивояжере с n городами и матрицей затрат $C = \{c_{ij}\}, i = \overline{1,n},\ j = \overline{1,n}$. Множество $S(0)$ состоит из (n-1)! допустимых решений. Метод Литтла основан на разбиении множества $S(0)$ на два непересекающихся подмножества и на вычислении оценок каждого из них. Далее подмножество с минимальной оценкой (стоимостью) разбивается на два подмножества и вычисляются их оценки. На каждом шаге выбирается подмножество с наименьшей из всех полученных на этом шаге оценок и производится его разбиение на два подмножества. В конце концов получаем подмножество, содержащее один цикл (замкнутый маршрут, удовлетворяющий наложенным ограничениям), стоимость которого минимальна.

1.8 Алгоритм метода ветвей и границ

Метод состоит из предварительного этапа и общего, который повторяется необходимое число раз.

Предварительный этап. Приведение матрицы затрат $\{c_{ij}\}$, вычисление нижней оценки стоимости маршрута r.

Вычисление наименьшего элемента по каждой строке (константы приведения):

$$\alpha_i = \min_j c_{ij}, \quad i = \overline{1,n} \tag{1.8.1}$$

Переход к новой матрице с элементами:

$$c'_{ij} = c_{ij} - \alpha_i \tag{1.8.2}$$

Вычисление наименьшего элемента по каждому столбцу (константы приведения):

$$\beta_j = \min_i c_{ij}, \quad j = \overline{1,n} \tag{1.8.3}$$

Переход к новой матрице с элементами:

$$c_{ij}^0 = c_{ij}' - \beta_j \tag{1.8.4}$$

Вычисление нижней оценки стоимости маршрута (сумма констант приведения):

$$r = \sum_{i=1}^{n} \alpha_i + \sum_{j=1}^{n} \beta_j \tag{1.8.5}$$

В результате выполнения предварительного этапа получаем матрицу $\{c_{ij}^0\}$ в каждой строке и в каждом столбце которой есть хотя бы один нулевой элемент.

Общий этап.

Вычисление штрафа “за неиспользование” p_{hk} для каждого нулевого элемента приведенной матрицы $\{c_{ij}^0\}$.

Если ребро (h,k) не включается в маршрут, то в него входит некоторый элемент строки h и столбца k. Следовательно, стоимость «неиспользования» (h,k) во всяком случае, не меньше суммы минимальных элементов строки h и столбца k, исключая сам элемент c_{hk}^0. Отсюда

$$p_{hk} = \min_{j \neq k} \{c_{hj}^0\} + \min_{i \neq h} \{c_{ik}^0\} \tag{1.8.6}$$

Выбор нулевого элемента, которому соответствует максимальный штраф. Если таких элементов несколько, то выбирается любой из них. Разбиение множества всех допустимых маршрутов $S(0)$ на два подмножества: подмножество содержащее ребро (h,k) – $S(h,k)$; подмножество, не содержащее ребро (h,k) – $S(\overline{h,k})$.

Примечание: максимальный штраф означает, что исключение из решения переезда, соответствующего нулевому элементу, приведет к максимальному увеличению стоимости оптимального маршрута.

Вычисление оценок затрат по всем маршрутам, входящим в каждое подмножество.

Обозначим за $\theta(\overline{h,k})$ минимальную оценку стоимости маршрутов, вошедших в множество $S(\overline{h,k})$, т.е. не содержащих ребро (h,k). Для $S(\overline{h,k})$ оценка затрат:

$$\theta(\overline{h,k}) = r + p_{hk} \tag{1.8.7}$$

При вычислении оценки затрат для $S(h,k)$ учитывают, что, если ребро (h,k) входит в маршрут, то ребро (k,h) не может входить в маршрут, поэтому принимаем: $c^0_{kh} = \infty$; если в маршрут включено ребро (h,k), то ни одно другое ребро, начинающееся в пункте h или заканчивающееся в пункте k не может входить в маршрут, поэтому строка h и столбец k вычеркиваются. Полученная матрица приводится, т.е. выполняется предварительный этап алгоритма. Пусть сумма приводящих констант матрицы: r_{hk}. Тогда оценка затрат:

$$\theta(h,k) = r + r_{hk} \tag{1.8.8}$$

Из множеств $S(h,k)$ и $S(\overline{h,k})$ для дальнейшего ветвления выбирается множество, имеющее меньшую оценку. При выборе $S(h,k)$ нужно вернуться к шагу 1, используя на этом шаге приведенную матрицу, полученную на шаге 3.2. При выборе $S(\overline{h,k})$ нужно вернуться к матрице $\{c'_{ij}\}$, принять $c'_{hk}=\infty$ и привести полученную в результате матрицу, после чего перейти к шагу 2, используя на нем эту приведенную матрицу. Если несколько множеств имеют равную минимальную оценку, то дальнейшее ветвление производится для всех множеств с минимальной оценкой. Таким образом метод ветвей и границ позволяет находить все оптимальные решения.

Алгоритм продолжается до тех пор, пока в подмножестве маршрутов с наименьшей оценкой не останется всего один маршрут. В расчетах это соответствует ситуации, когда исследуемая матрица имеет размерность 1x1.

Рассмотрим пример решения задачи о коммивояжере методом ветвей и границ. В таблице (А.1) приведена матрица затрат С размерностью 5x5. Элементы матрицы С, стоящие на главной диагонали, равны: $c_{ii}=\infty$, так как переезд из i-го города в i-ый невозможен.

На предварительном этапе выполняют приведение матрицы затрат, для чего вычисляют константы приведения α_i по строкам (табл. А.2).

Затем переходят к новой матрице затрат с элементами c'_{ij}, которые получены в результате вычитания из всех элементов исходной матрицы c_{ij} констант приведения α_i (табл. А.3). Далее вычисляются константы приведения β_j по столбцам (табл. А.3).

Выполняют переход к новой матрице затрат с элементами c_{ij}^{0}, полученными путем вычитания из элементов матрицы c_{ij} (табл. А.3) констант приведения β_j. Результирующая матрица затрат C^0 приведена (табл. А.4), т.е. в каждой строке и в каждом столбце есть хотя бы один нулевой элемент. Нижняя оценка стоимости маршрута равная сумме констант приведения по строкам и столбцам составляет: r=45. Оценка множества всех допустимых маршрутов $S(0)$ также равна: $\Theta(0)$= r=45. Следует обратить внимание на то, что из нулевых элементов матрицы затрат $\{c_{ij}^{0}\}$ невозможно составить полный замкнутый цикл, следовательно оптимальный маршрут будет иметь большую стоимость по сравнению с полученной оценкой: r=45.

Таблица А.1

	1	2	3	4	5
1	∞	14	9	16	7
2	20	∞	9	19	14
3	18	15	∞	12	12
4	23	10	13	∞	17
5	7	6	6	6	∞

Таблица А.2

	1	2	3	4	5	α_i
1	∞	14	9	16	7	**7**
2	20	∞	9	19	14	**9**
3	18	15	∞	12	12	**12**
4	23	10	13	∞	17	**10**
5	7	6	6	6	∞	**6**

Таблица А.3

	1	2	3	4	5
1	∞	7	2	9	0
2	11	∞	0	10	5
3	6	3	∞	0	0
4	13	0	3	∞	7
5	1	0	0	0	∞
β_j	**1**	**0**	**0**	**0**	**0**	**0**

Таблица А.4

	1	2	3	4	5
1	∞	7	2	9	0^{2}
2	10	∞	0^{5}	10	5
3	5	3	∞	0^{0}	0^{0}
4	12	0^{3}	3	∞	7
5	0^{5}	0^{0}	0^{0}	0^{0}	∞

На первом шаге общего этапа вычисляют штраф “за неиспользование” p_{hk} для каждого нулевого элемента приведенной матрицы $\{c''_{ij}\}$ (А.4). Штраф “за неиспользование” p_{hk} равен сумме минимальных элементов по строке и столбцу, на пересечении которых находится нулевой элемент. Все штрафы указаны непосредственно в соответствующих клетках табл. А.4.

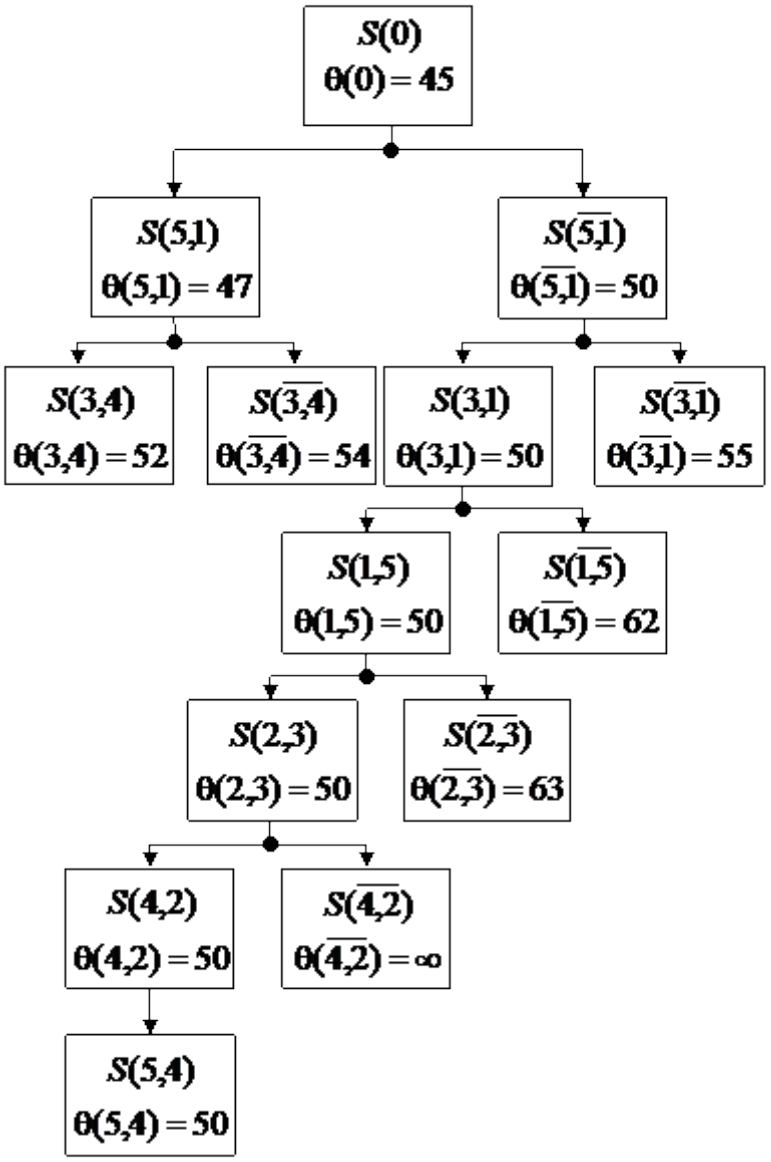


Рисунок 1.8.1.1 Дерево решения для задачи о коммивояжере

На втором шаге выбирают нулевой элемент, которому соответствует максимальный штраф – например, элемент (5,1) и разбивается множество всех допустимых маршрутов $S(0)$ на два подмножества: $S(5,1)$ – подмножество содержащее ребро (5,1); $S(\overline{5,1})$ – подмножество, не содержащее ребро (5,1). Процесс решения наглядно иллюстрирует дерево решений (рис. 1.8.1.1).

На третьем шаге вычисляют оценки затрат по всем маршрутам, входящим в каждое подмножество. Для $S(\overline{5,1})$ оценка затрат:

$$\theta(\overline{5,1}) = r + p_{51} = 45 + 5 = 50$$

Для вычисления оценки затрат для $S(5,1)$ принимают: $c'_{15} = \infty$ и вычеркивают строку 5 и столбец 1, т.е. полагают, что $c_{5j} = \infty$, при $j = \overline{2,5}$; $c_{i1} = \infty$, $i = \overline{1,4}$ (табл. А.5). Полученную таким образом матрицу (табл. А.6) приводят и переходят к новой матрице (табл. А.7). Затем вычисляют сумму констант приведения: $r_{51} = 2$ и оценку затрат подмножества $S(5,1)$: $\theta(5,1) = r + r_{51} = 45 + 2 = 47$.

Таблица А.5

	1	2	3	4	5
1	∞	7	2	9	∞
2	10	∞	0^5	10	5
3	5	3	∞	0^0	0^0
4	12	0^3	3	∞	7
5	0^5	0^0	0^0	0^0	∞

Таблица А.6

	2	3	4	5	α_i
1	7	2	9	∞	**2**
2	∞	0	10	5	**0**
3	3	∞	0	0	**0**
4	0	3	∞	7	**0**

На четвертом шаге из множеств $S(5,1)$ и $S(\overline{5,1})$ для дальнейшего ветвления выбирается множество $S(5,1)$, имеющее меньшую оценку.

Далее итерационно выполняются шаги 1-4 общего этапа. В табл. А.8 приведены штрафы “за неиспользование” нулевых элементов. Максимальный штраф соответствует элементу (3,4), поэтому множество $S(5,1)$ разбивается на два подмножества: $S(3,4)$ и $S(\overline{3,4})$ (рис. 1.8.1.1). Для $S(\overline{3,4})$ оценка затрат: $\theta(\overline{3,4}) = 47 + p_{34} = 47 + 7 = 54$. Для вычисления оценки затрат для $S(3,4)$ принимают: $c''_{43} = \infty$, вычеркивают строку 3 и столбец 4.

Полученную матрицу (табл. А.8) приводят и переходят к новой матрице (табл. А.9). Затем вычисляют сумму констант приведения: $r_{34} = 5$ и оценку затрат подмножества $S(3,4)$: $\theta(3,4) = 47 + 5 = 52$.

Из множеств $S(3,4)$, $S(\overline{3,4})$, $S(\overline{5,1})$ для дальнейшего разбиения выбирают множество $S(\overline{5,1})$, так как его оценка минимальна на этом этапе решения задачи и составляет: $\theta(\overline{5,1}) = 50$. Следовательно, необходимо согласно алгоритму вернуться к матрице табл. 4, принять $c_{51}^{\cdot} = \infty$ и привести полученную в результате матрицу (табл. А.10).

Таблица А.7

	2	3	4	5
1	5	0	7	∞
2	∞	0	10	5
3	3	∞	0	0
4	0	3	∞	7

Таблица А.8

	2	3	4	5
1	5	0^5	7	∞
2	∞	0^5	10	5
3	3	∞	0^7	0^5
4	0^6	∞	∞	7

Таблица А.9

	2	3	5
1	5	0	∞
2	∞	0	5
4	0	∞	7
β_j	0	0	5

После чего переходят к шагу 1 алгоритма и вычисляют штрафы “за неиспользование” нулевых элементов приведенной матрицы (табл. А.11).

Для дальнейшего ветвления выбирается элемент (3,1). Оценка затрат множества $S(\overline{3,1})$ составляет: $\theta(\overline{3,1}) = 55$. Для вычисления оценки затрат

для $S(3,1)$ принимают: $c''_{13} = \infty$, вычеркивают строку 3 и столбец 1 (табл. А.12). Полученная матрица уже приведена. Оценка затрат множества $S(3,1)$ равна: $\theta(3,1) = 50$. Для дальнейшего разбиения выбирается множество $S(3,1)$.

Таблица А.10

	1	2	3	4	5
1	∞	7	2	9	0
2	10	∞	0	10	5
3	5	3	∞	0	0
4	12	0	3	∞	7
5	∞	0	0	0	∞
β_j	5	0	0	0	0	0

Таблица А.11

	1	2	3	4	5
1	∞	7	2	9	0^{2}
2	5	∞	0^{5}	10	5
3	0^{5}	3	∞	0^{0}	0^{0}
4	7	0^{3}	3	∞	7
5	∞	0^{0}	0^{0}	0^{0}	∞

Таблица А.12

	1	2	3	4	5
1	∞	7	2	9	0
2	10	∞	0	10	5
3	5	3	∞	0	0
4	12	0	3	∞	7
5	∞	0	0	0	∞

Таблица А.13

	2	3	4	5
1	7	∞	9	0^{12}
2	∞	0^{5}	10	5
4	0^{3}	3	∞	7
5	0^{0}	∞	0^{9}	∞

На следующей итерации ветвление выполняется относительно элемента (1,5) (табл. А.13, рис. 1.8.1.1). Оценка затрат множества $S(\overline{1,5})$ составляет: $\theta(\overline{1,5}) = 50 + p_{15} = 62$. Для вычисления оценки затрат для $S(1,5)$ принимают: $c''_{53} = \infty$, чтобы избежать неполного замкнутого цикла $3 \to 1 \to 5 \to 3$, вычеркивают строку 1 и столбец 5. Полученная матрица уже приведена. Оценка затрат множества $S(1,5)$ равна: $\theta(1,5) = 50$.

Далее выполняется разбиение множества $S(1,5)$ на два подмножества: $S(2,3)$ и $S(\overline{2,3})$ (табл. А.14, рис. 1.8.1.1).

На последней итерации множество $S(2,3)$ разбивается на два подмножества: $S(4,2)$ и $S(\overline{4,2})$ (табл. А.15, рис. 1.8.1.1).

Таблица А.14

	2	3	4
2	∞	0^{13}	10
4	0^{3}	3	∞
5	∞	∞	0^{10}

Таблица А.15

	2	4
4	0^{∞}	∞
5	∞	0^{∞}

Результат решения задачи в терминах модели задачи может быть представлен в виде матрицы переездов вида:

$$X^* = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}, \quad Z(X^*) = 50.$$

В терминах исходной задачи в виде маршрута: $1 \to 5 \to 4 \to 2 \to 3 \to 1$. Стоимость маршрута равна 50.

Задача коммивояжера является частичным случаем гамильтоновой задачи о путешественнике. Суть задачи коммивояжера состоит в нахождении суммарной минимальной характеристики (расстояния, стоимости проезда и т.д.), при этом коммивояжер должен пройти все n городов по одному разу, вернувшись в тот город, с которого начал.

Существуют несколько методов решения задачи коммивояжера: метод полного перебора, с помощью метода ветвей и границ (алгоритм Литтла),

алгоритма Крускала, «деревянного» алгоритма и т.д. Однако только метод ветвей и границ дает нам в итоге самое оптимальное решение.

Основная идея метода Литтла состоит в том, что вначале строят нижнюю границу длин маршрутов для всего множества гамильтоновых контуров . Затем все множество контуров разбивают на два подмножества таким образом, чтобы первое подмножество состояло из гамильтоновых контуров, содержащих некоторую дугу (i,j), а другое подмножество не содержало этой дуги.

Для практической реализации идеи метода ветвей и границ применительно к задаче коммивояжера нужно найти метод определения нижних границ подмножества и разбиения множества гамильтоновых контуров на подмножества (ветвление). Такое определение нижних границ базируется на том утверждении, что если ко всем элементам i-й строки или j-го столбца матрицы C прибавить или отнять число , то задача останется эквивалентной прежней, то есть оптимальность маршрута коммивояжера не изменится, а длина любого гамильтонова контура изменится на данную величину .

1.9 Обзор программных продуктов, позволяющих решать проблему построения маршрутов

Было проведено исследование программных продуктов, которые позволяют решать проблему построения маршрутов. В ходе анализа, выяснилось, что разработанные программные продукты разные по своему назначению и функционалу, требуют тонкой настройки, либо имеют некорректный алгоритм работы, из-за этих факторов возникает множество проблем с построением маршрута.

1.9.1 Программный продукт ОПТИМУМ ГИС

ОПТИМУМ ГИС – система планирования оптимальных маршрутов посещений торговых точек выездными сотрудниками, мониторинга фактического передвижения мобильных сотрудников.

Модульная система ОПТИМУМ ГИС использует технологии спутникового слежения на основе данных ГЛОНАСС или GPS.

В состав системы входят модули **«Маршрутизация++»** и **«Мониторинг»,** которые могут работать совместно или автономно.

Модуль **«Маршрутизация++»** позволяет автоматически создавать оптимальные маршруты посещения объектов выездными сотрудниками, учитывая направление и проходимость дорог, транспортных развязок, время работы на объекте, позволяет распределить точки по зонам ответственности ТП; присвоить координаты торговым точкам; получить отчеты по соответствию маршрутов.

Программа поддерживает распечатку предполагаемого маршрута на бумаге.

Оператор получает возможность просмотра на карте всех объектов, закрепленных за конкретным мобильным сотрудником, и может оптимизировать распределение объектов по зонам ответственности каждого.

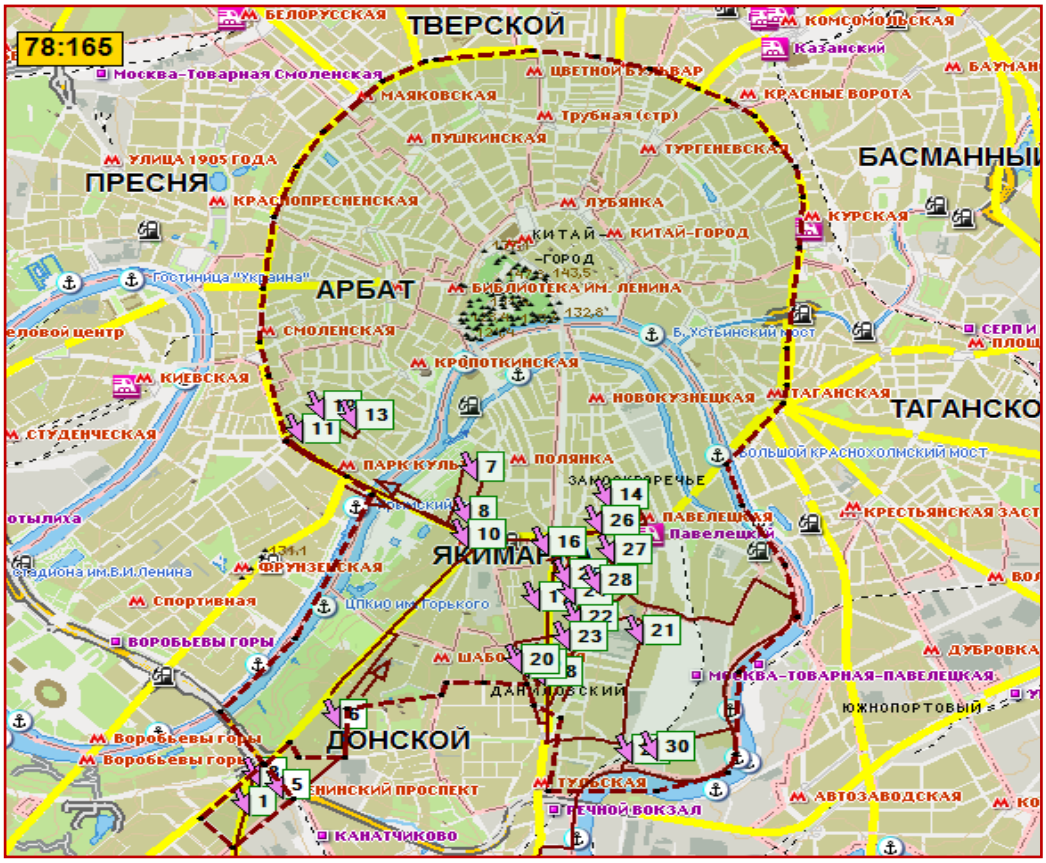


Рисунок 1.9.1.1 - Маршрутизация ++. Сценарий № 1

Вводная:

У нас есть набор торговых точек (ТТ) для посещения и определенное количество выездных сотрудников. При этом сотрудники не работают по определенным зонам и за ними не закреплены торговые точки.

Результат:

Получаем оптимизированные маршруты для каждого сотрудника. При построении маршрута учитываться будет только удаленность точек между собой.

Имеется несколько вариантов распределения торговых точек между торговыми представителями в данной программе, в зависимости от фирмы, города и требуемого распределения.

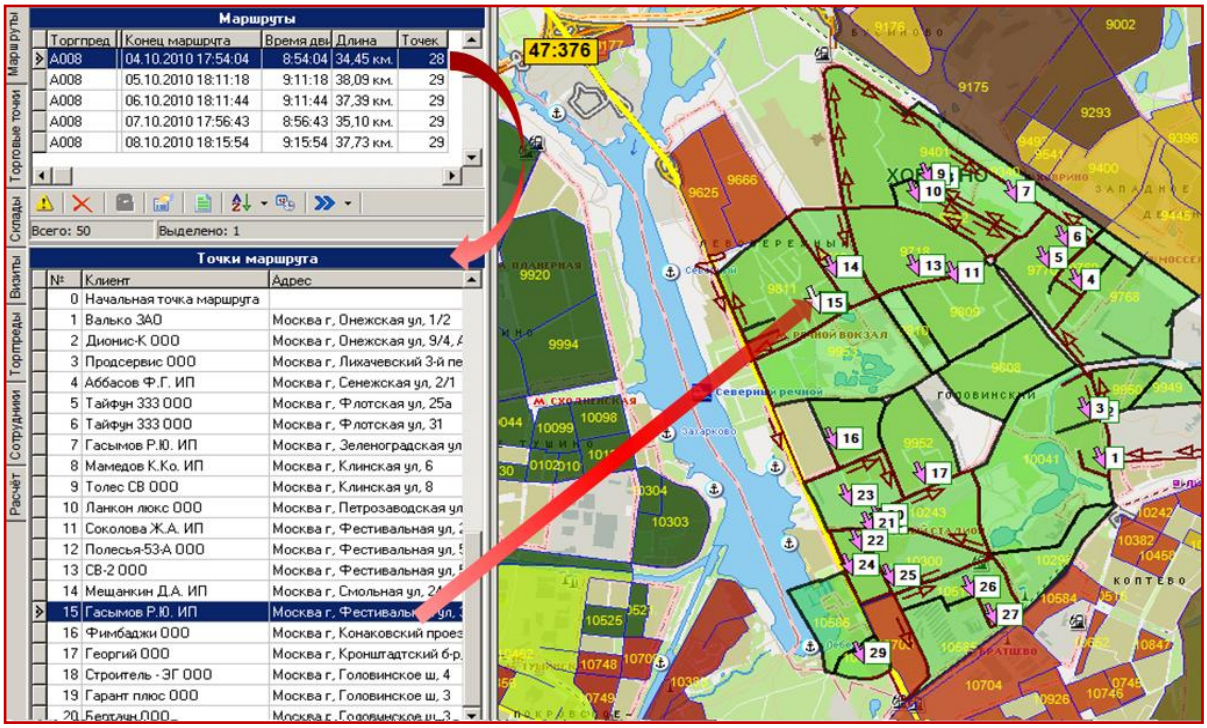


Рисунок 1.9.1.2 - Маршрут торгового представителя

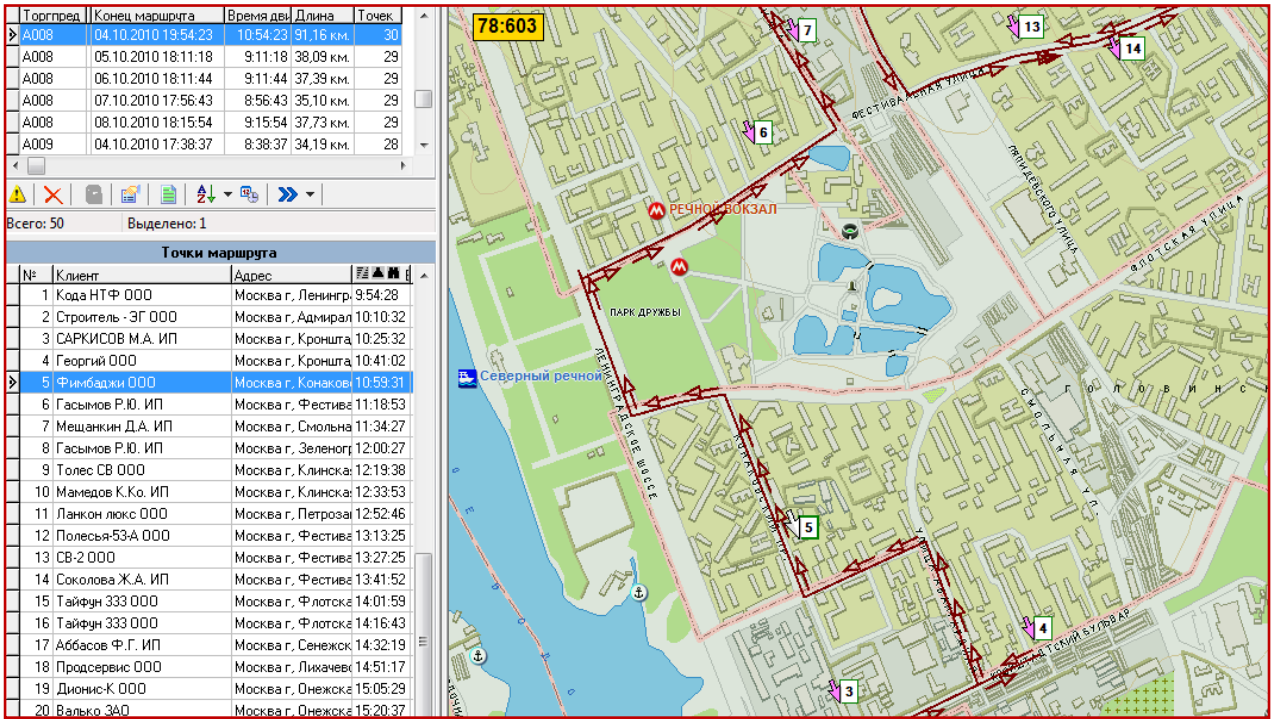


Рисунок 1.9.1.3 - Маршрут торгового представителя. Крупно. С домами.

На рисунках 1.9.1.2 и 1.9.1.3 показаны составленные маршруты торговых представителей на карте, с показанием точек, требуемых для посещения.

Клиент	Расстояние	Время	Время разгрузки	Время приезда	Время отправления	Адрес
Начальная точка маршрута	22085	0:44:10	0:00:00	2010-10-01 09:00:00.000	2010-10-01 09:00:00.000	
РАМАРИЯ	180	0:02:10	0:14:00	2010-10-01 09:46:10.000	2010-10-01 09:58:10.000	Москва г, Крутицкий 3-й пер, 15
ДШ ООО	19	0:00:14	0:14:00	2010-10-01 10:00:19.000	2010-10-01 10:14:19.000	Москва г, Крутицкий 3-й пер, 16, ст
ДШ ООО	960	0:03:50	0:14:00	2010-10-01 10:14:33.000	2010-10-01 10:28:33.000	Москва г, Крутицкий 3-й пер, 16, ст
ОЛЬФ ООО	606	0:02:25	0:14:00	2010-10-01 10:32:23.000	2010-10-01 10:46:23.000	Москва г, Краснохолмская наб, 13, 1
ДЭР ООО	200	0:02:24	0:14:00	2010-10-01 10:48:49.000	2010-10-01 11:02:49.000	Москва г, Товарищеский пер, 1
НЕДИЛЬСКИЙ	352	0:04:13	0:14:00	2010-10-01 11:05:13.000	2010-10-01 11:19:13.000	Москва г, Таганская пл, 2
СЕЙЛМЕН	607	0:02:26	0:14:00	2010-10-01 11:23:26.000	2010-10-01 11:37:26.000	Москва г, Факельный Б. пер, 12
ЭДЕМ	92	0:01:06	0:14:00	2010-10-01 11:39:52.000	2010-10-01 11:53:52.000	Москва г, Таганская ул, 44
МУСТАФАЕВ	496	0:05:57	0:14:00	2010-10-01 11:54:58.000	2010-10-01 12:08:58.000	Москва г, Таганская ул, 31/22
АС ГРУПП	52	0:00:37	0:14:00	2010-10-01 12:14:55.000	2010-10-01 12:28:55.000	Москва г, Андроньевская Б. ул, 11/1
СТУПЕНЬ	971	0:03:53	0:14:00	2010-10-01 12:29:33.000	2010-10-01 12:43:33.000	Москва г, Верхняя ул, 5
Двадцать первый	1618	0:06:28	0:14:00	2010-10-01 12:47:26.000	2010-10-01 13:01:26.000	Москва г, Волочаевская ул, 19
НИКО	259	0:03:06	0:14:00	2010-10-01 13:07:54.000	2010-10-01 13:21:54.000	Москва г, Нижегородская ул, 98
ПРОДТОРГ	0	0:00:00	0:14:00	2010-10-01 16:08:10.000	2010-10-01 16:22:10.000	Москва г, Автомобильный проезд, 10,
ТЮТ ООО	486	0:05:50	0:14:00	2010-10-01 16:22:10.000	2010-10-01 16:36:10.000	Москва г, Автомобильный проезд, 10
ГЛЕЙД	21834	0:43:38	0:14:00	2010-10-01 16:42:00.000	2010-10-01 16:56:00.000	Москва г, Остаповский проезд, 5, ст
Конечная точка маршрута	0	0:00:00	0:00:00	2010-10-01 17:39:37.000	2010-10-01 17:39:37.000	
Общий километраж	**55143**					
время в движении		**2:35:38**				
Время работы на точках			6:04:00			
Визитов	26					
Документов	26					

Рисунок 1.9.1.4 - Детальный отчёт по маршруту торгового представителя

На рисунке 1.9.1.4 отображен отчёт, выводимый в конце рабочего дня по каждому рабочему, расстояние в метрах, время в пути, разгрузки, приезда и отправления из торговой точки, а также ее адрес, расстояние пройденное за рабочий день, количество визитов и требуемых для сдачи отчетных документов (накладных).

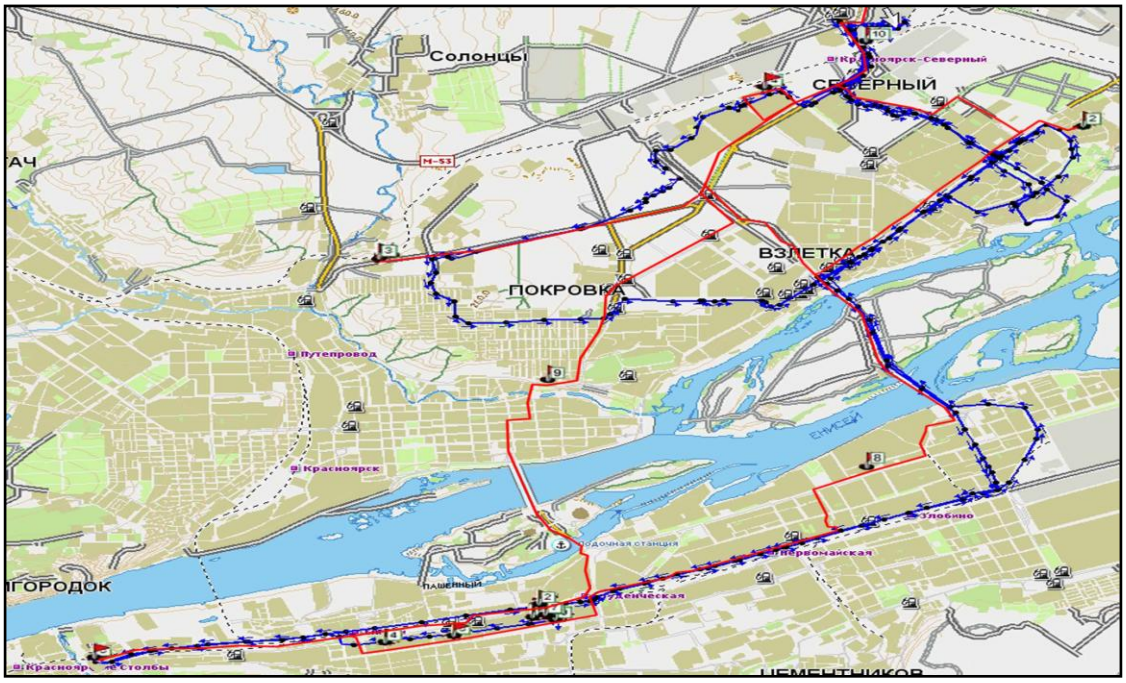


Рисунок 1.9.1.5 – Оптимальный и реальный маршрут автомобиля сотрудника по данным спутников

Получив карту с учетом всех маршрутов (Рисунок 1.9.1.5), диспетчер сразу увидит, где был водитель, и куда заезжал по личным делам.

Минусы:

Требуется тонкая настройка и изменение параметров системы, для успешной работы программы.

Требуется постоянная поддержка работы ПО.

Высокая стоимость ПО (от 75 000 до 200 000 руб), в зависимости от необходимого функционала.

Плюсы:

Отличная возможность отслеживать местонахождение сотрудника.

Оптимальное составление маршрута, в связи с использованием статистических временных данных.

1.9.2 Программа составления маршрутов от ООО "Интегрированные программы"

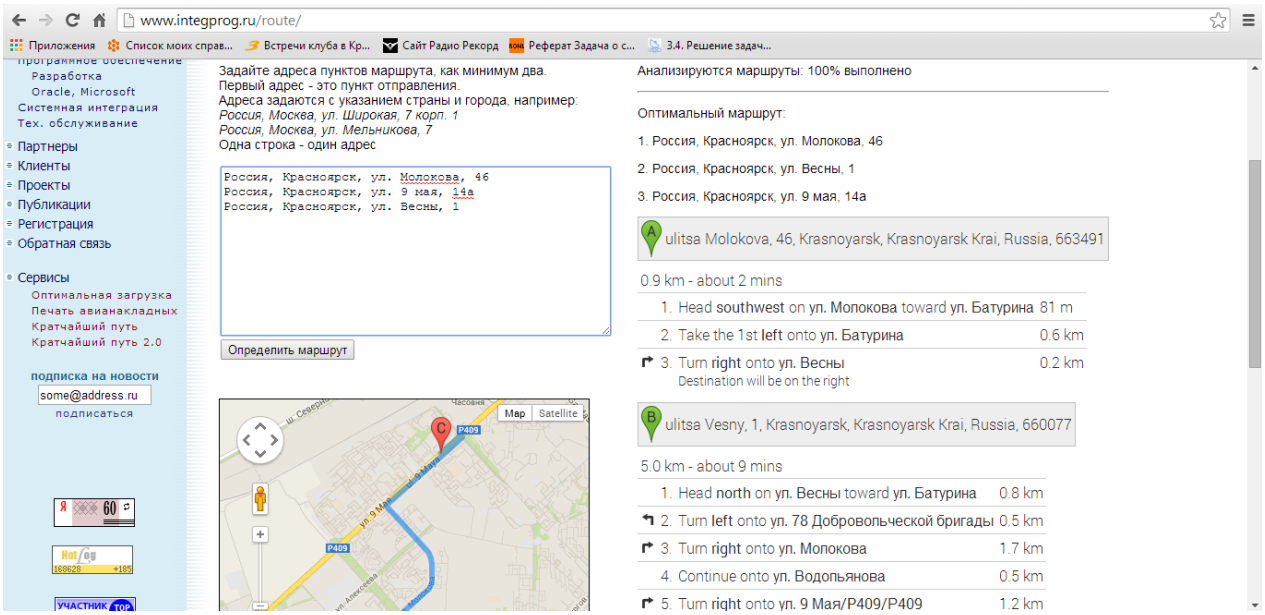


Рисунок 1.9.2.1 – Пример работы программы от ООО «Интегрированные программы»

В свободном доступе можно использовать программу составления маршрутов от ООО «Интегрированные системы» на сайте http://www.integprog.ru/route/.

Google.Maps предоставляет сервис прокладки маршрута по заданным пунктам. Когда Вы вводите несколько адресов, скрипт анализирует все предлагаемые маршруты и выбирает такую последовательность обхода пунктов, при которой общая дистанция маршрута минимальна. К сожалению, нет возможности влиять на выбор конкретной схемы проезда из пункта А в пункт Б. Например в примере, для проезда от ул. Широкой, 7 до ул. Мельникова, 7 Google.Maps предлагает ехать через МКАД, хотя по карте видно, что путь через центр будет короче. Это останется на совести разработчиков.

Обращаем Ваше внимание, что данный веб-сервис лишь использует возможности Google.Maps API и не является сервисом компании ООО

"Интегрированные программы". Предложенный маршрут может не являться оптимальным.

1.9.3 Описание сервиса Yandex API

Для получения времени на маршрут от точки до точки используется сервис Yandex API, который также позволяет рассчитывать маршруты с учетом пробок.

API (Application Programming Interface) — это набор готовых инструментов какого-либо приложения, которые можно использовать в сторонних продуктах.

API Яндекс.Карты — это программный интерфейс, с помощью которого появляется возможность установить карту и весь необходимый для работы с ней инструментарий к себе на сайт.

Расширить возможности карты на вашем сайте можно с помощью автомобильной маршрутизации.

Вы можете задать начальную, конечную и промежуточные точки пути. Между ними автоматически будет построен маршрут напрямую или в объезд пробок. Обратите внимание на то, что маршрут в объезд пробок прокладывается на основе данных о дорожной ситуации на момент построения. Также при построении рассчитывается длина маршрута и приблизительное время его прохождения.

При желании можно разбить маршрут на сегменты — минимальные отрезки пути. Это позволит, например, точно рассчитывать стоимость доставки товаров из магазинов и офисов. На данный момент доступна маршрутизация по всей России и Украине.

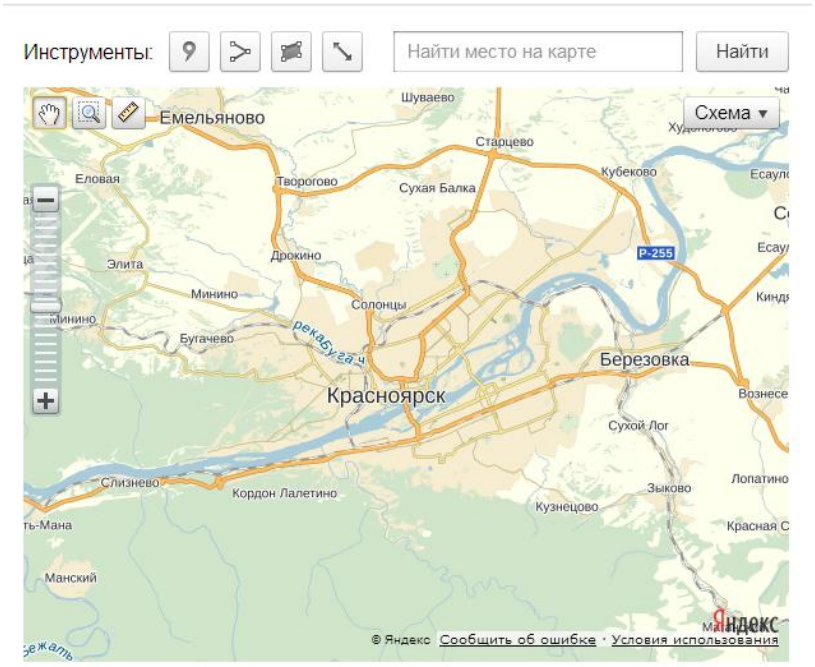


Рисунок 1.9.3.1 – Пример работы приложения Yandex API (Яндекс.Карты)

В версии 2.1 полностью изменился дизайн элементов управления картой. Изменения коснулись не только внешнего вида интерфейса, но и поведения его отдельных частей. Теперь они автоматически адаптируются под фактические размеры контейнера карты. Адаптивное поведение можно настроить и для собственных кнопок и списков.

Выводы

В первой главе представлена информация о предприятии, подробно рассмотрены и проанализированы программные продукты, которые позволяют решать похожие задачи составления маршрута. Анализ работы данных приложений помог в разработке алгоритма составления маршрута для программной системы, в которой были учтены недочеты ранее написанных программ, с учетом требований заказчика.

2 Система построения оптимального маршрута

2.1 Постановка задачи

Торговый представитель должен выйти из офиса и посетить по разу в неизвестном порядке все торговые точки. Торговые точки – вершины графа. За дуги графа взято время между торговыми точками. Маршрут следует выстроить таким образом, чтобы успеть посетить все торговые точки, которые запланированы на рабочий день в назначенное время и максимально освободить время после посещения торговых точек, для поиска новых клиентов.

2.2 Данные для расчета

Входные данные: сотрудник, точки, которые нужно посетить, и время для любых двух точек, требуемое для переезда из одной точки в другую; информация о точках, содержащаяся в массиве points, такая как время, с которого можно посещать точку (visitFrom), время, до которого следует посетить точку (visitTo). Значение const, которое является результатом интенсивных исследований, оценивающее приблизительное время, в среднем затрачиваемое перевозчиком на доставку товара из одной точки в другую.

Выходные данные: конкретные моменты времени, в которые необходимо посетить все точки, уложившись в заданные ограничения, либо сообщение о том, что для данного набора точек невозможно построить расписание.

Статистика посещений, после посещения каждой торговой точки значения вносятся в базу данных в формате:

Таблица 2.2.1 База данных по статистике посещений ТТ и времени на работу в ТТ

Клиент	Расчетное время прохождения пути по программе(a)	Фактическое время в пути(A)	Плановое время выполнения работы в ТТ (b)	Фактическое время за которое выполнена работа в ТТ (B)
1…				
2…				

Время в пути до ТТ рассчитывается по формуле:

$$\text{Т(время в пути до ТТ)} = \frac{t(a)}{k\,(\text{коэфф.сотрудника})}, \quad (2.2.1)$$

$$\textit{где } k = \frac{t(A)}{t(a)}$$

Для расчета времени работы в торговой точке, используется значение Т, рассчитанное по формуле для каждого вида работ в каждой ТТ:

$$T(i) = \frac{t(1)+\cdots+t(n)}{n}, i = [1, n] \quad (2.2.2)$$

2.3 Алгоритм решения задачи построения оптимального маршрута

В ходе решения задачи о составлении оптимального маршрута для торгового представителя, был разработан следующий алгоритм:

1. Упорядочиваем все точки в порядке возрастания их свойства visitFrom; если для двух точек оно равно, то смотрим на свойство visitTo, и упорядочиваем точки в порядке возрастания этого свойства. Далее переходим к шагу 2.

2. Создаем массив timeline, которая будет содержать конкретные моменты времени, в которые будут посещаться точки.

3. Инициализируем переменную i значением 1 и переходим к шагу 4.

4. Для двух точек в массиве points i - 1 и i вычисляем время, требуемое на переезд из точки i - 1 в точку i, и прибавляем к нему время, требуемое на проведение сделки в точке i . Переходим к шагу 5.

5. Если получившийся момент времени попадает в промежуток от points[i].visitFrom до points[i].visitTo, то добавляем этот момент в i-ый элемент массива timeline, переходим к шагу 4.

Если момент времени получился меньше, чем points[i].VisitFrom, то складываем points[i].VisitFrom и время, требуемое на совершение сделки в points[i], и добавляем получившееся в timeline. Переходим к шагу 4.

Если момент времени больше, чем points[i].visitTo, то переходим к шагу 6.

6. Если i > 1 и i < [(размер массива points) - 1], то присваиваем переменной k значение i – 2, присваиваем переменной diffPrev значение, максимально возможное для переменной этого типа, и переходим к шагу 7.

7. Если k < 1, то переходим к шагу 9. Иначе вычисляем время, требуемое на переезд из точки k в точку i, прибавляем к нему время, за которое нужно

выполнить сделку в точке i и проверяем, попадает ли это время в промежуток (points[i].visitFrom, points[i].visitTo).

a) Если да, то вычисляем время, требуемое на переезд из точки i в точку k + 1. Затем вычисляем разницу между этим временем и временем для переезда из точки k в точку

k + 1. Присваиваем значение этой разницы переменной diff. diff представляет собой величину, на которую сдвинутся все точки после точки k + 1, если мы решим вставить точку i после точки k. Если diff меньше diffPrev, то присваиваем переменной insertAfter значение k, где insertAfter – индекс элемента в массиве points, после которого необходимо будет, если получится, вставить точку points[i] . Присваиваем diffPrev значение diff. Переходим к шагу 8.

b) Если нет, то переходим к шагу 9.

8. Уменьшаем значение k на 1. Переходим к шагу 7.

9. Переставляем точку i в позицию insertAfter + 1. Сдвигаем все точки после insertAfter + 2 на значение diff и проверяем их на совместимость с их значениями visitFrom и visitAfter. Если какая-либо точка не успевает, то завершаем алгоритм и отправляем клиенту информацию о невозможности построения маршрута. В противном случае переходим к шагу 10.

10. Присваиваем переменной i значение i + 1 и, если значение i меньше, чем количество точек в массиве points, то переходим на шаг 4. Иначе переходим на шаг 11.

11. Присваиваем переменной i значение 1. Переходим к шагу 12.

12. Создаем двухмерный массив intersections (значения пересечений), где количество столбцов равно 2, а количество строк увеличивается в процессе

работы алгоритма. Создаем массив intersectPoints (точки, которые пересекаются) и заносим туда значение points[i].

13. Присваиваем переменной j значение i + 1. Присваиваем интервалу (intersectFrom, intersectTo) значение (points[i].visitFrom, points[i].visitTo). Переходим к шагу 14.

14. Проверяем, пересекаются ли интервалы (intersectFrom, intersectTo) и (points[j].visitFrom, points[j].visitTo). Если да, то присваиваем время начала этого пересечения переменной intersectFrom, а время конца пересечения переменной intersectTo. Помещаем значение этих переменных в элементы массива intersections[Last][0] и intersections[Last][1], где Last – индекс последней строки в массиве intersections. Значение j помещаем в массив intersectPoints[Last], где Last – индекс последнего элемента в массиве intersectPoints, и переходим к шагу 15. Если нет, то переходим к шагу 16.

15. Присваиваем переменной j значение j + 1. Если оно меньше, чем длина массива points, то переходим к шагу 14, иначе переходим к шагу 16.

16. Если длина массива intersections больше 2, то переходим к шагу 17, иначе переходим к шагу 20.

17. Если значение intersects[Last][1] - intersects[Last][0] больше либо равно const * [количество значений в массиве intersectPoints], то переходим к шагу 19, иначе переходим к шагу 18.

18. Удаляем из intersectPoints последний элемент и последнюю строку в массиве intersects и переходим к шагу 16.

19. Решаем задачу коммивояжера для точек, находящихся в массиве intersectPoints. Создаем копию массива points и timeline, pointsTemp и

timelineTemp соответственно. Изменяем порядок точек в массиве pointsTemp и пересчитываем значения в массиве timelineTemp в соответствии с новым порядком точек в массиве pointsTemp. Проверяем, попадают ли все точки в массиве timelineTemp по времени в назначенные им интервалы работы. Если да, то заменяем значения массивов timeline и points значениями timelineTemp и pointsTemp. Если нет, то оставляем их неизменными. Переходим к шагу 20.

20. Присваиваем переменной i значение i + [количество точек в массиве intersectPoints] и, если результирующее значение i меньше, чем длина массива points, переходим к шагу 12. Иначе считаем работу алгоритма законченной.

2.4 Алгоритмическая сложность

Сложность - это количественная оценка ресурсов, затрачиваемых алгоритмом.

Ресурсы:

- Человеческие (создание и понимание алгоритма). Оценивает интеллектуальная сложность. Единого критерия оценки не существует.
- Вычислительные (на выполнение алгоритма):
    - Память. Оценивает пространственная сложность - количество памяти, требующееся для выполнения алгоритма.
    - Процессорное время. Оценивает временная сложность - количество времени, необходимое на выполнение алгоритма.

Если пространственная сложность поддается количественной оценке, то с временной сложностью не все так просто - оценка алгоритма зависит от аппаратуры. Можно сравнивать 2 алгоритма на одинаковой аппаратуре, но чтобы избавиться от аппаратной зависимости используется оценка в виду

функции T(n) от количества обрабатываемых данных. Эта оценка применяется к отдельным частям алгоритма, затем комбинируется в общую оценку. Для конкретизации этой функции используется несколько подходов. В частности O-оценка.

2.5 Сложность алгоритма Коммивояжера

Поскольку коммивояжер в каждом из городов встает перед выбором следующего города из тех, что он ещё не посетил, существует **(n-1)!** маршрутов для асимметричной и **(n-1)!/2** маршрутов для симметричной задачи коммивояжера. Таким образом, размер пространства поиска зависит экспоненциально от количества городов.

Различные варианты задачи коммивояжера (метрическая, симметричная и асимметричная) NP-эквивалентны. Согласно распространенной, но недоказанной гипотезе о неравенстве классов сложности P и NP, не существует детерминированной машины Тьюринга, способной находить решения экземпляров задачи за полиномиальное время в зависимости от количества городов.

Также известно, что при условии **P≠NP** не существует алгоритма, который для некоторого полинома Ƥ вычислял бы такие решения задачи коммивояжера, которые отличались бы от оптимального максимум на коэффициент $\mathbf{2}^{\mathbf{Ƥ(n)}}$.

Однако, существуют алгоритмы поиска приближенных решений для метрической задачи за полиномиальное время и нахождения маршрута максимум вдвое длиннее оптимального. До сих пор не известен ни один алгоритм с полиномиальным временем, который бы гарантировал точность, лучшую чем 1,5 от оптимальной. По предположению **P≠NP**, существует

(неизвестная) константа **C > 0**, такая, что ни один алгоритм с полиномиальным временем не может гарантировать точность **1+C**.

2.6 Сложность построенного алгоритма

Алгоритм сортировки — это алгоритм для упорядочения элементов в списке. В случае, когда элемент списка имеет несколько полей, поле, служащее критерием порядка, называется ключом сортировки. На практике в качестве ключа часто выступает число, а в остальных полях хранятся какие-либо данные, никак не влияющие на работу алгоритма.

Сложность алгоритма сортировки n log n, где n- количество объектов.

$$\sum_{i=1}^{n} O(i) = n^2$$

$$\sum_{i=1}^{n} O(i)^2 = n^4$$

Сложность построенного алгоритма составляет от n^2 до n^4.

Рисунок 2.6.1 – График сложности алгоритма по времени и количеству точек

Выводы

Во второй главе подробно рассматриваются методы и алгоритмы, позволяющие решать задачи о составлении маршрута, разобран и показан разработанный алгоритм решения задачи построения оптимального маршрута для торгового представителя, вычислена алгоритмическая сложность данного алгоритма.

3 Описание программного продукта. Руководство пользователя.

3.1 Рекомендуемые требования к аппаратному обеспечению

Тактовая частота процессора 300 MHz или более, от 128 MB RAM для Windows 2000/XP или Linux, от 512 MB RAM для Windows Vista/Seven;

3.2 Требования к программному обеспечению

- Операционная система Microsoft Windows 2000/XP/Vista/Seven x32|x64 или Linux+Wine);
- 8 MB свободной дисковой памяти на системном диске;
- Расширеный пакет программы Microsoft Visual Studio 2012 с компонентами SQL

3.3 Установка программы

В процессе работы программа сохраняет информацию на носитель, поэтому некоторые функции программы не будет работать, если она будет находиться на CD-дисках. Для установки программы скопируйте папку с программой в директорию **C:\Users***\Documents\VisualStudio2012\Projects\WebSiteOptimization.**

В комплект программы входят:

исполняемый файл MultiSystem.exe

файл базы данных Clients.mdf

3.4 Апробация программы по составлению маршрута торгового представителя

Таблица 3.4.1 - База данных клиентов для проверки работы программы

Контрагент	Адрес
Ионесси	Сурикова 12/1
	Мира 57
	Красноярский рабочий 51
	Металлургов 34
	26 Бакинских Комиссаров 31а
	9 мая 38 ж
	Новосибирская 5
Большая обувь XXXL	Красноярский рабочий 135
Выбор	Парижской Коммуны 31
Д'Ларис	Матросова 8
	Мира 104
Коллекция	78 Добр. Бригады 21
Магазин обуви (ИП Речкина)	Красноярский рабочий 16
Май	9 мая 14а
Обувной на Мира	Мира 106
Обувь для Вас	Красноярский рабочий 55
ИП Калинина Квант	Красной армии 10
Магазин обуви Квант 1 этаж	Красной армии 10
La' Фамилья	Молокова 13
Lloyd	Белинского 3
Clarks	Белинского 3
Marco Tozzi	Партизана Желязняка 23
Осминожка	Молокова 56/1
Pal Zileri	Мира 109
Ralf Ringer	Комсомольский проспект 18
	Мате Залки 5
SOGO	Партизана Желязняка 23
ИП Колосова	78 Добр. Бригады 19
	Академика Вавилова 33
	Красноярский рабочий 27 ст74
	Красноярский рабочий 27 ст77
Саяны (Мандарин)	Красноярский рабочий 74д
	Павлова 8

Конкурент	Красноярский рабочий 62
Оружейный центр "Тигр"	Кольцевая 1б
ИП Семыкина	Аэровокзальная 2а
Egoiste	Телевизорная 1
ИнтерОбувь	Дубровинского 54а
ООО "Валео"	Красная площадь 1
	Красноярский рабочий 106
ООО "ЗдравМед"	Офис, самовывоз (Молокова 46)
ИП Холодилова	Тельмана 28в
ИП Лещева	Воронова 18
Глава крестьянского хозяйства Савин (М-н Креатив)	Весны 1
	Красноярский рабочий 120
ИП Лукина	Урванцева 10
Магазин Март	60 лет Октября 43
	9 мая 20

3.5 Руководство пользователя

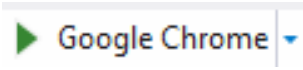
1. Требуется открыть программу составления маршрута

1.1.1 Открываем приложение Microsoft Visual Studio 2012 (Рис. 3.5.1)

1.1.2 Файл – Открыть - Решение или проект… (Ctrl+Shift+O) (Рис. 3.5.2)

1.2.1 Либо перейти в корневой каталог, где находится программа и запустить исполняемый файл WebSiteOptimization (Рис. 3.5.3)

1.3 Авторизация –требуется ввести логин и пароль (Рис. 3.5.4)

1.4 Во вкладке Main.aspx нажимаем кнопку 

(просмотреть)

1.5 В вашем браузере запускается Веб-страница с программой (Рис. 3.5.5)

2. Начало работы

2.1 Задаем продолжительность рабочего дня торгового представителя (Рис. 3.5.6)

2.2 Из загруженной базы данных выбираем контрагента (Рис. 3.5.7)

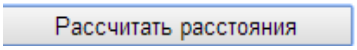
2.3 Выбираем адрес магазина (Рис. 3.5.8)

2.4 Выбираем вид работ («дело»), который требуется выполнить в данной торговой точке (Рис. 3.5.9)

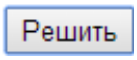
2.5 Задаем предпочитаемое время посещения торговой точки (Рис. 3.5.10)

2.6 Смотрим на карте, как расставлены точки (Рис. 3.5.11)

2.7 Выбранные точки заносятся в таблицу (Рис. 3.5.12)

2.8 После внесения всех торговых точек, которые требуется посетить, нажимаем кнопку 

, если торговые точки не занесены в таблицу выводится ошибка (Рис. 3.5.13)

2.9. Расстояния рассчитаны и занесены в программу, далее нажимаем кнопку 

и получаем готовое решение нашей задачи (Рис. 3.5.14)

2.10 Внесение данных из торговой точки в БД. Прикрепление фотоотчета. Пересчет маршрута по актуальным данным (Рис. 3.5.15)

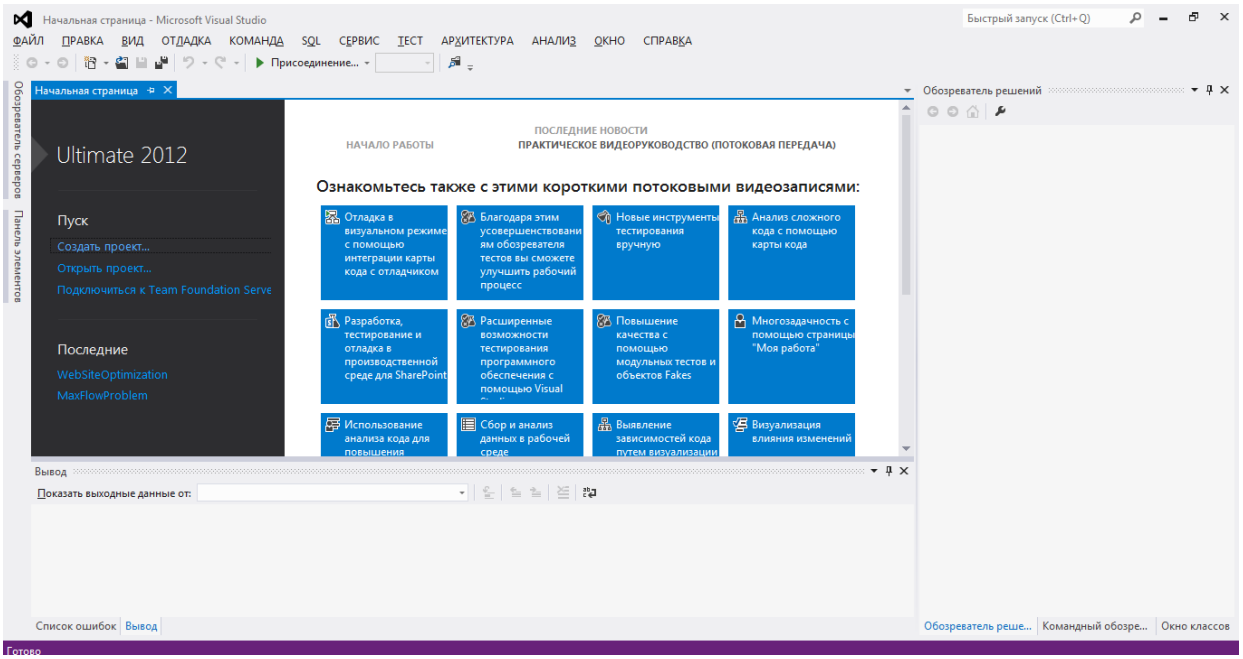


Рисунок 3.5.1 – Открытие приложения Microsoft Visual Studio 2012

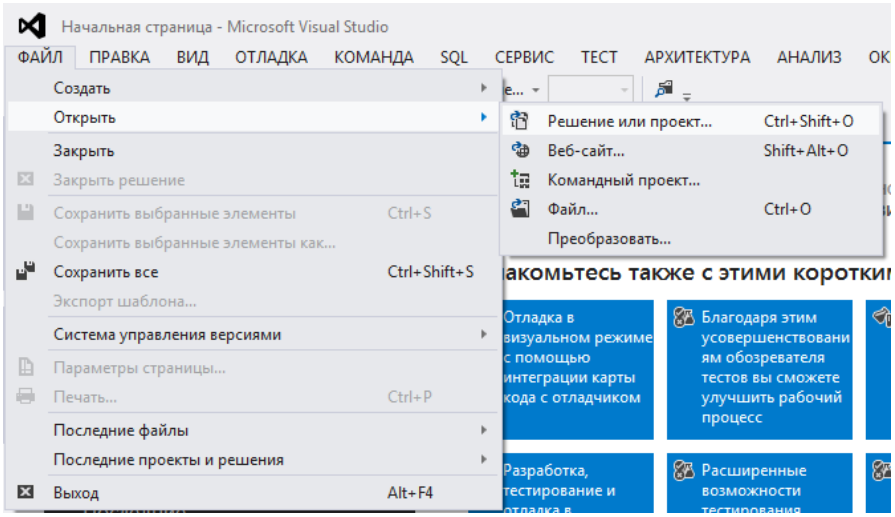


Рисунок 3.5.2 – Открыть решение или проект

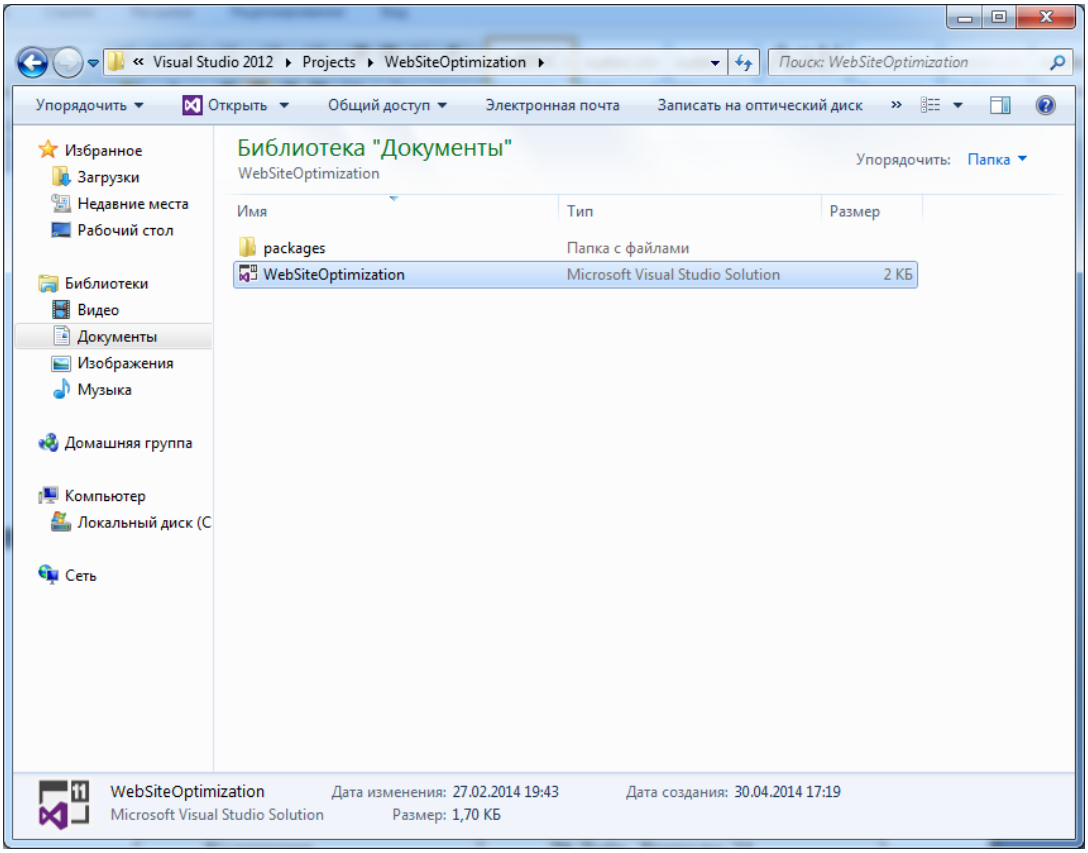


Рисунок 3.5.3 – Открытие программы WebSiteOptimization

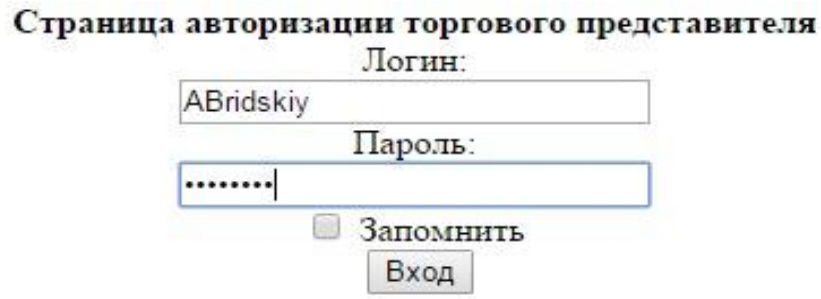


Рисунок 3.5.4 - Аутентификация сотрудника

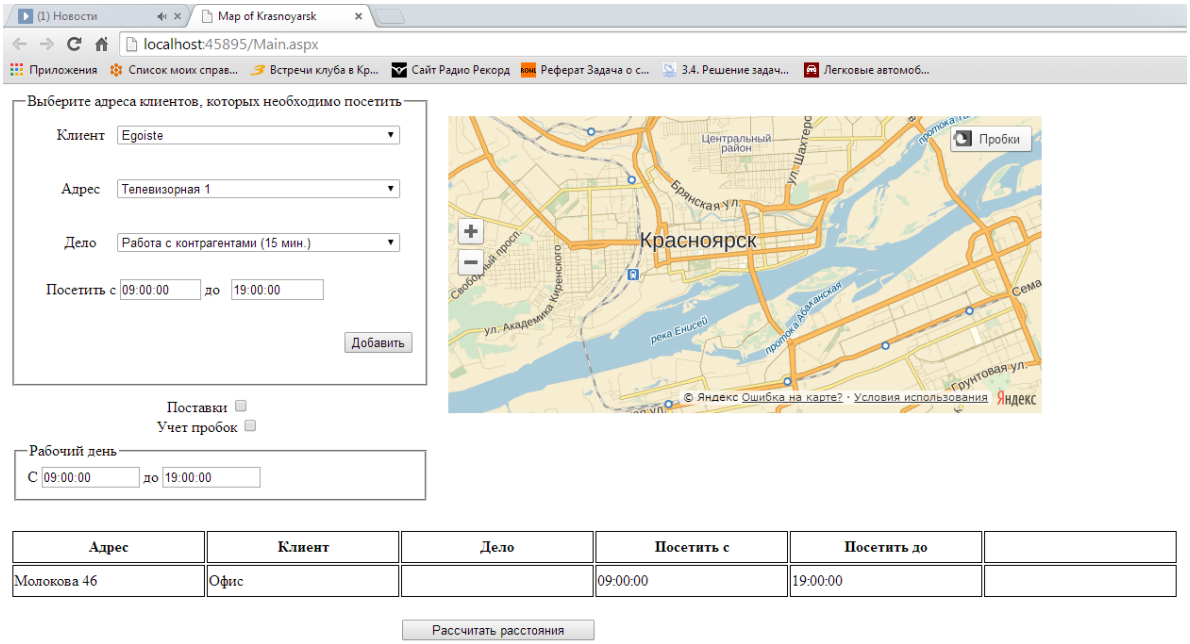


Рисунок 3.5.5 – Пример работы программы

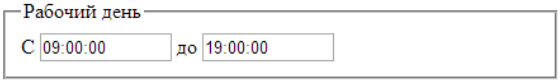


Рисунок 3.5.6 - Установка графика рабочего дня сотрудника

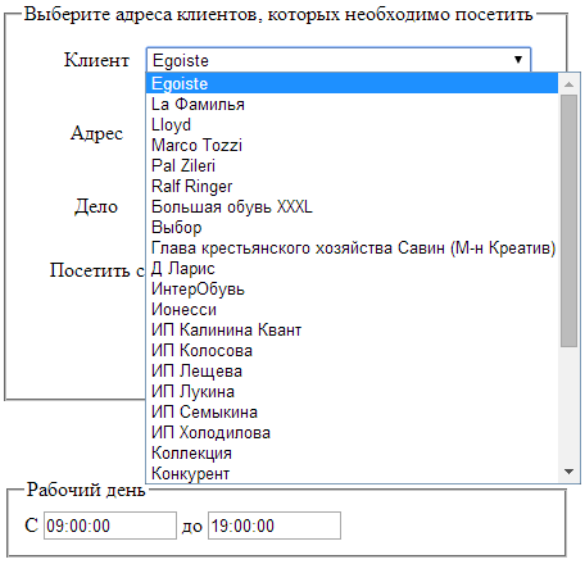


Рисунок 3.5.7 - Выбор клиента из базы данных

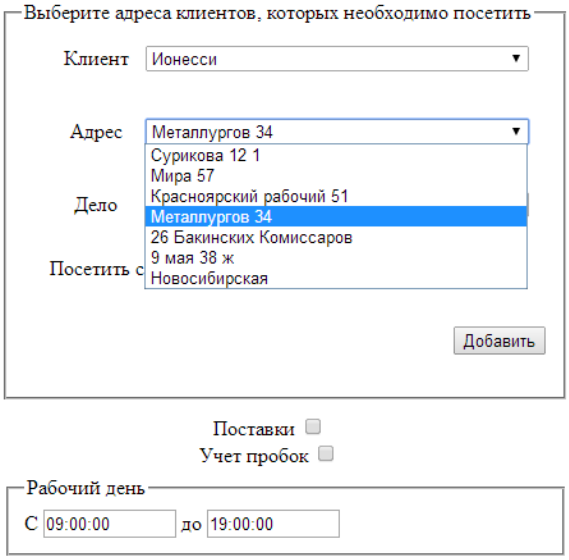


Рисунок 3.5.8 - Выбор адреса магазина

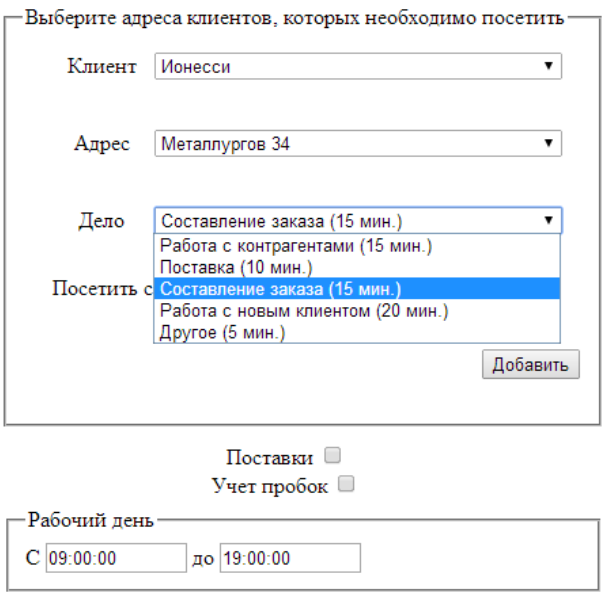


Рисунок 3.5.9 - Выбор вида работ, который требуется выполнить в торговой точке

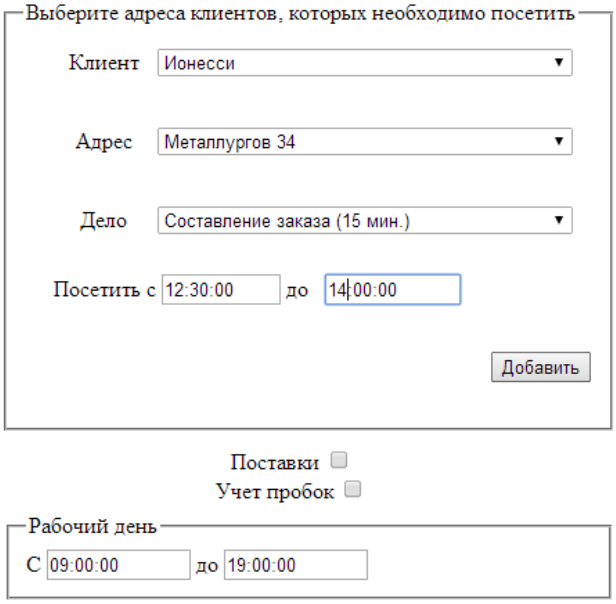


Рисунок 3.5.10 - Задание предпочитаемого времени посещения торговой точки

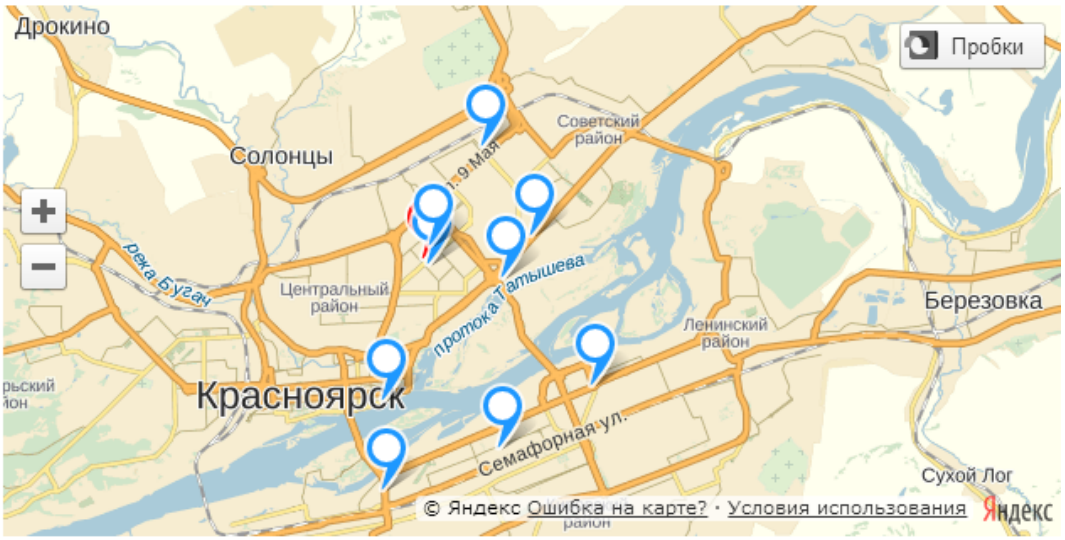


Рисунок 3.5.11 - Расставление торговых точек на карте

Адрес	Клиент	Дело	Посетить с	Посетить до	
Молокова 46	Офис		09:00:00	19:00:00	
Металлургов 34	Ионесси	Составление заказа (15 мин.)	12:30:00	14:00:00	Удалить
Комсомольский проспект 18	Ralf Ringer	Составление заказа (15 мин.)	09:00:00	19:00:00	Удалить
Молокова 13	La Фамилья	Поставка (10 мин.)	09:00:00	19:00:00	Удалить
78 Добр. Бригады 19	ИП Колосова	Поставка (10 мин.)	09:00:00	14:00:00	Удалить
Академика Вавилова 33	ИП Колосова	Поставка (10 мин.)	11:00:00	15:00:00	Удалить
Матросова 8	Д Ларис	Другое (5 мин.)	09:00:00	19:00:00	Удалить
Партизана Железняка 23	Marco Tozzi	Работа с новым клиентом (20 мин.)	17:00:00	19:00:00	Удалить
Красноярский рабочий 62	Конкурент	Поставка (10 мин.)	09:00:00	19:00:00	Удалить
Дубровинского 54а	ИнтерОбувь	Поставка (10 мин.)	09:00:00	19:00:00	Удалить

Рисунок 3.5.12 - Добавление точек в таблицу маршрута

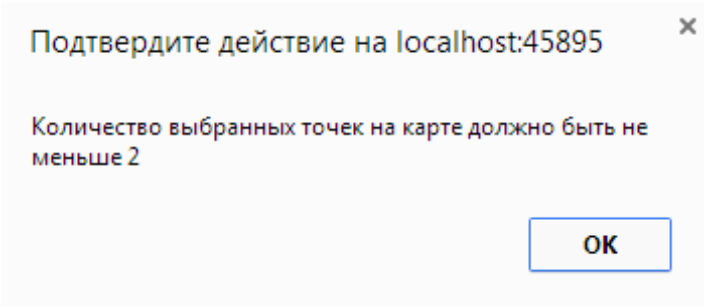
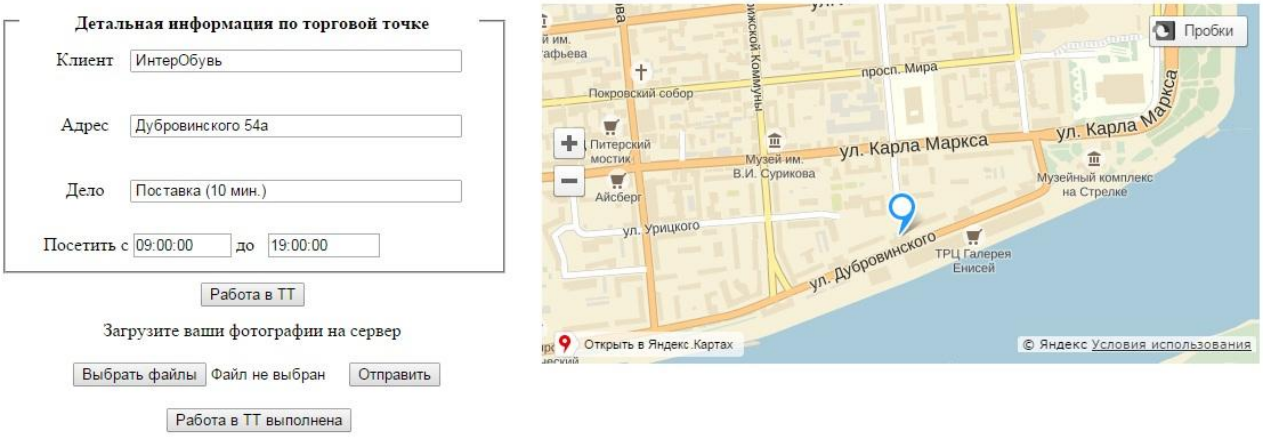


Рисунок 3.5.13 – Ошибка расчета расстояния

От	До	Дело	Приблизительное время
		Сбор	С 09:00:00 до 09:30:00
Молокова 46	78 Добр. Бригады 19	Поставка (10 мин.)	С 09:30:00 до 09:43:23
78 Добр. Бригады 19	Красноярский рабочий 62	Поставка (10 мин.)	С 09:43:23 до 10:11:41
Красноярский рабочий 62	Матросова 8	Другое (5 мин.)	С 10:11:41 до 10:15:01
Матросова 8	Комсомольский проспект 18	Составление заказа (15 мин.)	С 10:15:01 до 11:11:40
Комсомольский проспект 18	Дубровинского 54а	Поставка (10 мин.)	С 11:11:40 до 11:35:37
Дубровинского 54а	Молокова 13	Поставка (10 мин.)	С 11:35:37 до 11:54:17
Молокова 13	Академика Вавилова 33	Поставка (10 мин.)	С 11:54:17 до 12:25:22
Академика Вавилова 33	Металлургов 34	Составление заказа (15 мин.)	С 12:25:22 до 13:03:53
Металлургов 34	Партизана Железняка 23	Работа с новым клиентом (20 мин.)	С 13:03:53 до 17:20:00
Партизана Железняка 23	Молокова 46		С 17:20:00 до 17:23:59

Рис. 3.5.14 - Вывод решения



Риуснок 3.5.15 - Внесение данных из торговой точки в БД

Выводы

В третьей главе подробно рассмотрена разработанная система, использующая в качестве основы представленный во второй главе алгоритм. Показана подробная структура и работа программы на примере конкретной базы данных клиентов, требования и руководство пользователя. Разработанная программная система позволяет составить маршрут торгового представителя, с учетом времени посещения торговой точки, чего нет в ранее реализованных

приложениях, благодаря реализованному алгоритму это стало возможным. Программа внедрена на предприятие и проста в использовании. Разработанный метод точно удовлетворяет всем пожеланиям заказчика и может быть доработан для использования на других предприятиях.

ЗАКЛЮЧЕНИЕ

В ходе разработки программной системы построения маршрутов торгового представителя, были достигнуты следующие основные результаты:

- Проведено изучение алгоритмов составления маршрутов и программных решений;
- Разработан алгоритм составления маршрута, с учетом предпочтительного времени посещения
- Разработана система, позволяющая построить маршрут с учетом текущей дорожной ситуации, особенностей каждой торговой точки и личных коэффициентов сотрудников;
- Учтены статистические данные по каждому сотруднику, для максимально эффективного расчета маршрута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Карасев, А.И. Курс высшей математики для экономических вузов: Учебник для экономических вузов / А.И. Карасев, З.М. Аксютина, Т.И. Савельева - Москва: Высшая школа, 1982. - С.279-285.
2. Полунин, И.Ф. Курс математического программирования: Учебник для вузов / И.Ф. Полунин - Минск: Высшая школа, 1970. - С. 194-230.
3. Сакович, В.А. Исследование операций: Учебник для вузов / В.А. Сакович - Минск: Высшая школа, 1985. - С.75.
4. Красс, М.С. Математика для экономистов: Учебник для экономических вузов / М.С. Красс - Санкт-Петербург: Питер, 2006. - С.289-299.
5. Сакович, В.А. Управление комплексными поставками: Учебник для вузов / В.А. Сакович - Минск: Высшая школа, 1989. - С.100-108.
6. Холод, Н.И. Математические методы анализа и планирования: Учебник для вузов / Н.И. Холод - Минск: Ураджай, 1989. - С.97-99.
7. Холод, Н.И. Пособие по решению задач по линейной алгебре и линейному программированию: Пособие для вузов / Н.И. Холод - Минск: издательство БГУ, 1971. - С.159.
8. Неруш, Ю.М. Логистика: учеб. – 4-е изд., перераб. и доп. / Ю.М. Неруш – Москва: Издательство Проспект, 2007. – 520 с.
9. Ильенкова, С.Д. Производственный менеджмент. Учебник / С.Д. Ильенкова – Москва: ЮНИТИ, 2000. Миротин Л.Б. Логистика. – М.: Юристь, 2002.
10. Курганов, В.М. Логистические транспортные потоки / В.М. Курганов – Москва: Издательско-торговая корпорация «Дашков и К», 2003.
11. Гудков, В.А. Пассажирские автомобильные перевозки: Учебник для вузов / В.А. Гудков, Л.Б. Миротин, А.В. Вельможин, С.А. Ширяев – М.: Горячая линия – Телеком, 2006. – 488 с.

12. Горев, А.Э. Основы теории транспортных систем: учеб. пособие / А. Э. Горев; СПбГАСУ. – СПб., 2010. – 214 с.
13. Максимов, Ю.А. Алгоритмы решения задач нелинейного программирования / Ю.А. Максимов, Е.А. Филлиповская — М.: МИФИ, 1982.
14. Лопатин, А. П. Моделирование перевозочного процесса на городском пассажирском транспорте / А. П. Лопатин. – М.: Транспорт, 1985. – 144 с.
15. Котиков, Ю. Г. Основы системного анализа транспортных систем: учеб.пособие [Электронный ресурс] / Ю. Г. Котиков. – СПб.: СПбГАСУ, 2001. – 264 с.
16. Писецкая В.В. Логистика автомобильного транспорта: магистерская работа [Электронный ресурс] / Режим доступа: http://masters.donntu.edu.ua/2002/fem/pisetskaya/diss.htm
17. Ганущак Н.К. Исследование существующих алгоритмов решения транспортных задач в ГИС: магистерская работа [Электронный ресурс] / Режим доступа: http://www.masters.donntu.edu.ua/2006/ggeo/ganushchak/diss/index.htm
18. Семенов В.В. Математическое моделирование транспортных потоков мегаполиса [Электронный ресурс] / Режим доступа: http://spkurdyumov.narod.ru/Semenov.pdf
19. Гасников А.В. Введение в математическое моделирование транспортных потоков [Электронный ресурс] / МФТИ, 2010 –Режим доступа: http://crec.mipt.ru/study/courses/optional/gasnikov/Book.pdf