Санкт-Петербургский государственный университет

ОЛЕШКЕВИЧ Евгений Валерьевич

Выпускная квалификационная работа

Моделирование маршрутизации транспорта для решения задачи оптимизации работы логистической компании

Уровень образования: бакалавриат
Направление 02.03.02 «Фундаментальная информатика и информационные технологии»
Основная образовательная программа 02.03.02 «Программирование и информационные технологии»

Научный руководитель:
доцент, кафедра компьютерных технологий и систем, к.ф.-м.н.
Погожев С. В.

Рецензент:
генеральный директор, ООО «Передовые Технологии – Максимум Качества», к.ф.-м.н.
Тукачев П. А.

Санкт-Петербург
2021

Содержание

Введение

Многим предприятиям приходится сталкиваться с задачей транспортировки своих товаров или объектов производства. Планирование, контроль и управление этим процессом называется логистикой [29].

Задача состоит не только в построение маршрутов, расписаний и графиков, но и в оптимизации затраченных на это ресурсов, построение наиболее выгодных маршрутов или иначе задача маршрутизация транспорта (Vehicle Routing Problem, ЗМТ).

ЗМТ – задача комбинаторной оптимизации, впервые была поставлена в 1959 году в работе [31]. Изначально задача была в определении маршрутов доставки некоего товара со склада множеству клиентов, и с тех пор было разработано множество подвидов и постановок, а также методов и алгоритмов ее решения. Задача определения оптимального расписания маршрутов относится к классу NP-трудных, поэтому большинство разрабатываемых алгоритмов ставят задачу поиска наилучшего суб-оптимального решения за наименьшее время.

Неформальная постановка задачи

Задача маршрутизации транспорта (ЗМТ) является обобщением задачи коммивояжера (ЗК). Отличие заключается в наличии нескольких коммивояжеров (называемых экипажами, транспортными средствами или ТС), пути которых пересекаются только в одной вершине – депо (остальные вершины называются клиентами). Именно она является вторым отличием от ЗК, фиксированная вершина графа, в которой начинается и заканчивается маршрут каждого ТС.

В качестве решения задачи необходимо составить несколько замкнутых путей, начинающихся в депо и покрывающих все вершины графа, чтобы значение интересующего нас функционала качества решения было как можно меньше. В роли функционала может выступать суммарная длина всех

маршрутов, время движения по ним, предполагаемый объем затраченных денег, неустойка за опоздание доставки или сборки в вершине клиента.

Данная задача превосходит по сложности ЗК, а при наличии только одного экипажа является ею. Поэтому ЗМТ – NP-трудная задача.

Цель работы

Целью работы является решение задачи логистики для оптимизация вывоза твердых бытовых отходов (ТБО) на примере города Краснодар. В рамках чего исследуются различные постановки ЗМТ и алгоритмы их решения. А также объединение этих алгоритмов в ансамбль, налаживая взаимодействия алгоритмов. Тем самым получив гибкое настраиваемое программное средство, способную осуществлять построение расписания, не сильно отличающегося от оптимального, не только для конкретного взятого города, но и для большинство городов РФ и СНГ.

Методика исследования

В рамках решения задачи используются такие методы как, алгоритмы оптимизации путем локального, случайного и жадного поиска, математическое моделирование. А также используются различные программные средства для кодирования, получения, отображения и преобразования данных, в частности, Visual Studio, Anaconda3, Jupyter Notebook, OpenStreetMap с использованием языков программирования C++ и Python.

Практическая ценность

Данная работа предполагает создание удобного программного средства для решения задачи маршрутизации с различными требованиями и условиями. Полученное программное средство способно решать задачи логистики не только для вывоза твердых бытовых отходов, но и в целом составлять расписания доставки или сбора с оптимизацией длин полученных маршрутов (но не времени).

Общая постановка задачи маршрутизации транспорта

Задано:

1) множество вершин графа V = $\{v_0, v_1, ..., v_n\}$, где v_0 – депо, а остальные вершины v_i, $i = \overline{1,n}$ – клиенты;
2) матрица расстояний между всеми вершинами $C = \{c_{i,j}\}_{i,j=0}^{n,n}$, где $c_{i,j}$ – расстояние между вершинами v_i и v_j;
3) множество экипажей R = $\{r_1, ..., r_m\}$, количество m может быть задано или также является элементом поиска.

Требуется найти такую матрицу $X = \{x_{i,j}\}_{i,j=0}^{n,n}$, где $x_{i,j} = 1$, если в построенных нами маршрутах есть ребро(v_i, v_j), и $x_{i,j} = 0$, в противном случае, чтобы $\sum_{i=0}^{n}\sum_{j=0}^{n} c_{i,j} \cdot x_{i,j} \to min$, при ограничениях:

1) $\sum_{i=1}^{n} x_{i,j} = 1$, для всех $j = \overline{1,n}$;
2) $\sum_{j=1}^{n} x_{i,j} = 1$, для всех $i = \overline{1,n}$;
3) $\sum_{i=1}^{n} x_{0,i} = m$;
4) $\sum_{i=1}^{n} x_{i,0} = m$.

Первые два ограничения гарантируют однократное посещение каждой вершины-клиента. Ограничения под номерами 3 и 4 требуют использования ровно m маршрутов.

Модификации и обобщения ЗМТ

Обзор типов ЗМТ

Существует огромное множество различных модификаций постановки ЗМТ, даже каждую отдельно взятую задачу из реального мира можно считать как новую постановку. Приведем примеры наиболее распространенных:

1. **Ограничение количества экипажей:** количество экипажей может быть фиксированным, ограниченным снизу и (или) сверху или произвольным. В случае если требуется алгоритмически определить оптимальное количество, то лучше использовать алгоритмы, способные определить количество и работать с фиксированным, отлично могут подойти алгоритмы Моля-Джеймсона, Кристофидеса-Мингоззи-Тосса, муравьиный, жадный или алгоритмы кластеризации с определением числа кластеров.
2. **Количество депо:** разделяют случаи единственного депо или множества депо. ЗМТ с множеством депо зачастую решается 3 способами: кластеризация по депо и решение ЗМТ с одним депо для каждого кластера; алгоритм на муравьиных колониях; генетический алгоритм.
3. **Ограничение длины маршрута:** вводятся ограничения по расстоянию, времени в пути или количеству посещаемых пунктов для каждого маршрута, как правило ограничивает сверху, но бывает и снизу. Данное ограничение способны решать большинство алгоритмов, за исключением некоторых конструктивных методов основанных на слиянии маршрутов, такие как алгоритм Кларка-Райта.
4. **Ограничение симметричности (СЗМТ):** необходимо обеспечить, чтобы длины (или иные метрики) маршрутов не отличались друг от друга слишком сильно. Данное ограничение хорошо обрабатывается многофазными, жадными, муравьиными или генетическими алгоритмами, а также алгоритмом Османа и некоторыми его

модификациями. Подробнее о последнем алгоритме и симметричных ЗМТ можно узнать в [1].

5. **Ограничение грузоподъемности (ЗМТУГ):** потребности клиентов могут быть различны, а каждое ТС не может перевозить груза больше своей грузоподъемности, также могут рассматриваться несколько ТС с разной грузоподъемностью. ЗМТУГ схожа с двумя предыдущими, поэтому к ней применимы те же алгоритмы с небольшими модификациями.
6. **Неоднородность автопарка:** модификация предыдущего пункта (ЗМТУГ) с условием наличия нескольких ТС имеющих разные грузоподъемность, расход топлива или стоимость старта (зарплата водителя, аренда ТС и прочее). Данное ограничение является логическим продолжением предыдущего и может использовать аналогичные методы решения с небольшими доработками.
7. **Допущение раздельной доставки (ЗМТРД):** в данной модификации одна вершина-клиент может присутствовать в нескольких маршрутах, а в качестве решения необходимо не только предоставить маршруты, включающие в себя все вершины, но и указать на сколько удовлетворяется спрос каждой вершины при посещении. ЗМТРД является одной из сложнейших формулировок ЗМТ, так как значительно увеличивает пространство решений. Относительно просто решается муравьиным или генетическим алгоритмами, но наиболее эффективно алгоритмом VND-STS [27].
8. **Наличие обратного груза:** при посещении очередной вершины-клиента ТС может получить дополнительный груз, который необходимо вернуть в депо. Постановка сопряжена с ограничением грузоподъемности. Данная постановка описана в [26].
9. **Наличие временных окон на доставку:** необходимо доставить груз клиенту в определенное время. Является одной из сложнейших постановок ЗМТ, подробнее можно посмотреть в [25].

10. **Ограничение экологии (учет издержек эксплуатации ТС):** каждое ТС наносит ущерб экологии, вызывая некоторые издержки. Считается как время поездки, так и простоя, в зависимости от своего текущего состояния ТС приносит разные убытки, которые могут быть разными для разных ТС. В данном случае ведется оптимизация как количества ТС, так и назначения их на маршруты (самое “грязное” ТС можно назначить на самый короткий маршрут или вовсе исключить). Данное ограничение как правило предусматривает задачу выбора оптимального транспорта, поэтому решается аналогично постановке из пункта 6 с небольшими модификациями.
11. **Задача сбора и доставки:** накладывает ограничение на порядок вершин. Прежде чем доставить груз одному клиенту необходимо его забрать у другого. Подробный пример данной постановки можно посмотреть в [6].
12. **Динамическая ЗМТ:** в процессе работы могут появляться новые клиенты или исчезать старые. Данная постановка сложно тем, что для нее практически все алгоритмы оказываются не рабочими, а решения зачастую может не существовать вовсе. Описание динамической ЗМТ можно найти в [24].
13. **Стохастическая ЗМТ:** расстояние между вершинами не является постоянной величиной, а зависит от некоторого параметра (чаще всего времени), также спрос в вершинах-клиентах может зависеть от того же или другого параметра. Решение данной постановки приводится в [17].
14. **Топология и способ задания данных:** вершины могут быть заданы на плоскости, а расстояние между ними вычислять геометрически. Или же задается матрица расстояний, которая может быть симметричной или нет. При решении данной постановки основное отличие будет заключаться в обработке исходных данных. Но также это может накладывать ограничение на возможность использования некоторых алгоритмов. Так, если данные задаются матрицей расстояний, то нельзя

использовать алгоритмы, работающие за счет изучения геометрической структуры (алгоритм заметания, некоторые алгоритмы кластеризации).

15. **Задача маршрутизации запасов:** объединяет в себе ЗМТ и задачу управления запасами. Тем самым надо оптимизировать не только маршрутизацию доставок груза до клиентов, но и стоимость хранения этого груза ими, а также потерю прибыли клиента, если его запас иссяк. Подробнее о маршрутизации запасов написано в [6, 26].

Также существуют и другие типы ЗМТ. Если условия нескольких типов не противоречат друг другу, то они могут применяться одновременно. Тем самым получаем огромное количество комбинаций постановок. Подробнее о других типах можно узнать в [6, 17, 22].

Неформальная постановка задачи

В работе рассматривается задача построения расписания вывоза бытовых отходов с площадок их сбора в городе Краснодар. В качестве исходных данных используется информация об адресах площадок пунктах сбора, числе контейнеров на них, их объема и частоты необходимого опустошения в неделю. Данные берутся с сайта администрации и городской Думы Краснодара [7].

Используя информацию об адресах площадок с помощью сервиса OpenStreetMap [8, 9] была построена матрица расстояний между вершинами (площадками).

Следует отметить, что с точки зрения решения задачи нет разницы между доставкой груза от депо к клиентам или наоборот.

Задача будет состоять из комбинаций типов ЗМТ 1, 5, 6, 10, 14, 15. Входные данные:

- Несколько видов ТС и количество каждого вида, с расходом топлива, стоимостью старта (зависимой от амортизационной цены ТС, оплаты услуг водителя) и максимально допустимой вместительностью.

- Множество вершин с выделенным депо. Для каждого клиента заданы спрос и максимально допустимый период ожидания, до завершения которого данный клиент снова требует посещения (или необходимая частота посещения в неделю). В случае если клиент не был посещен в течение всего периода ожидания решение считается недопустимым.
- Матрица расстояний, не является симметричной.

Требуется предоставить набор маршрутов, разделенных на 7 групп по дням недели. Для каждого такого набора количество используемых транспортов одного типа не может превышать заданного числа. Число посещений каждой вершины-клиента соответствует требуемой частоте посещений во входных данных, причем число дней между посещениями (разность в номерах групп по модулю 7) не может превышать заданного. Иначе говоря, мы не можем посетить одну вершину в понедельник и вторник, а потом не посещать вовсе. Например, допустимым решением в таком случае может быть только прибытие в данную вершину в понедельник и в четверг или пятницу.

Формализованная постановка задачи

Заданы:

- множество вершин V = $\{v_0, v_1, ..., v_n\}$, где v_0 – депо, а остальные вершины – клиенты;
- функция $D(v_i) > 0$ – спрос вершины v_i, $i \neq 0$; $D(v_0) = 0$;
- функция $Freq(v_i) > 0$ – необходимая частота посещений в неделю вершины v_i, $i \neq 0$; $D(v_0) = 0$;
- матрица расстояний между всеми вершинами C, где $c_{i,j}$ – расстояние между вершинами v_i и v_j;
- множество типов ТС: T = $\{t_1, ..., t_m\}$;

- функция $CS(t_i)$>= 0 – цена запуска ТС типа t_i(CS – cost start), прибавка к стоимости маршрута, не зависящая от его длины, данное значение отвечает за оплату смены водителя, а также учета амортизационной стоимости или аренды ТС;
- функция $CF(t_i)$>= 1 – оценка объема расхода топлива ТС типа t_i (CF – fuel consumption);
- функция $W(t_i)$ >= 0 – грузоподъемность ТС типа t_i;
- функция $Count(t_i)$ >= 0 – количество ТС типа t_i.

Требуется составить группу из 7 расписаний, по одной группе на каждый день недели.

Расписание на день d:

- $G_d = (V_d, R_d), V_d = \{0, a_1, ..., a_{n_d}\}$ – множество индекс вершин, входящих в группу;
- $R_d = \{r_{1,t_1}, ..., r_{k,t_m}\}$ – множество маршрутов, R_{d,t_j} – только те маршруты из R_d, для которых определено ТС типа t_j;
- маршрут с использованием ТС типа t_j: $r_{i,t_j} = \{0, a_1, ..., a_k, 0\}$ – упорядоченные индексы вершин, входящих в маршрут.

Требования, предъявляемые к расписанию:

1. $\sum_{r \in R_d} r.contains(a_i) = 1$ для всех $i = \overline{1, n_d}; d = \overline{1, 7};$ – каждая вершина-клиент посещена в одном расписании единожды.
2. $\sum_{index \in r_{i,t_j}} D(v_{index}) < W(t_j)$, для каждого $r_{i,t_j} \in R_d$, $d = \overline{1, 7}$ – ограничение грузоподъемности.

3. $\sum_{r_{i,t_j} \in R_{d,t_j}} 1 < Count(t_i)$, для каждого $t_j \in T, d = \overline{1,7}$ – за один день не было использовано ТС одного типа больше, чем имеется.
4. Пусть $Groups(v_i)$= $\{g_1, ..., g_l\}$ – множество индексов дней, в группах которых состоит вершина v_i. Тогда: $Len(Groups(v_i))$ = Freq(v_i), а $g_{j+1} - g_j \leq [\frac{7}{Freq(v_i)}]$, $j = \overline{1,l}$, если $j = l$, то $g_{j+1} = g_1 + 7$ – ограничения, что каждая вершина будет посещена столько раз, сколько требуется, а период между посещениями не превышает допустимого числа (округление происходит вверх).

Алгоритмы решения ЗМТ

Как было сказано ранее существует множество различных формулировок ЗМТ, столь различные, что решение их одним алгоритмом не представляется возможным. Поэтому было разработано большое количество различных алгоритмов. Для лучшего их понимания они разбиты на классы (группы), хотя само деление весьма условно, и один алгоритм может принадлежать нескольким классам.

Выделяют следующие классы:

1) точные комбинаторные методы;
2) методы локальной оптимизации;
3) конструктивные алгоритмы;
4) эвристические алгоритмы;
5) многофазные алгоритмы;
6) метаэвристические алгоритмы.

Точные комбинаторные методы

Данные методы перебирают с предпочтениями различные возможные варианты маршрутизации транспорта. Главное их достоинство – точное оптимальное решение. Но они обладают существенный недостатком – непримелимо большое время работы, порой уже для графа с 30 вершинами нельзя за приемлемое время решить подобным способом.

- полный перебор;
- метод ветвей и границ.

Методы локальной оптимизации

Данная группа представляет собой улучшение имеющегося решения в рамках его небольшой окрестности. Может производиться как улучшение каждого маршрута в отдельности, так и всех вместе:

- 2-3-k-opt – применение 2-3-k-opt к каждому маршруту;

- Vertex/EdgeShiht – смещает вершину или ребро в рамках одного маршрута;
- Vertex/EdgeSwap – обмен местами между 2 вершинами или ребрами как внутри одного маршрута, так и между разными;
- Vertex/EdgeEject – перенос вершины или ребра из одного маршрута в другой.

Конструктивные методы

Алгоритмы этой подгруппы основаны на построении маршрутов из входного графа:

- алгоритм Кларка-Райта – определяет каждую вершину как отдельный маршрут, а затем производит их слияние. Более подробное описание можно найти ниже или в [1, 2, 4];
- алгоритм Моля-Джеймсона – усовершенствования версия предыдущего алгоритма, способен находить решения для неопределенного заранее числа маршрутов [1, 2];
- алгоритм Кристофидеса-Мингоззи-Тосса – двухэтапный алгоритм, сначала определяет число маршрутов, затем производит их построение [1, 2].

Эвристические алгоритмы

Данная группа включает в себя большой набор алгоритмов, основанных на практических методах и не дающих оптимального точного решения. Этих алгоритмов столь много, что их делят на подгруппы, одними из них являются:

- Жадный алгоритм – эвристический подход, основанный на череде принятий локальнооптимальных решений, предполагая что результат также окажется оптимальным. Несмотря на то, что данное предположение зачастую неверно, алгоритм способен находить достаточно хорошие решения.

- Ограниченный перебор – осуществляет перебор комбинаций и строит дерево принятия решений о принадлежности вершины к тому или иному маршруту, подобно методу ветвей и границ. Однако алгоритм имеет ограничение на высоту данного дерева, при превышении которой производится удаление корня и наименее эффективных ветвей [10, 11, 12].

Многофазные алгоритмы

Многофазность алгоритмов может проявляться по-разному. Зачастую есть 3 способа:

1) кластеризация вершин-клиентов, решение ЗК для каждого кластера;
2) решение ЗК для всех вершин, разбиение полученного маршрута на несколько непересекающихся;
3) применение конструктивных методов, после чего полученное решение используется в качестве стартового для одного из методов локального поиска или как часть метаэвристического подхода.

Примеры:

- алгоритм заметания – геометрический алгоритм, требующий чтобы вершины располагались на плоскости, более подробно будет описан ниже;
- алгоритм лепестков – представляет собой улучшение алгоритма заметания, подробнее можно посмотреть в [15];
- алгоритм Фишера-Джекумера – проводит кластеризацию, решая задачу о назначениях, а далее решается ЗК для каждого кластера [13];
- алгоритм Браммела-Симчи-Леви – схож с предыдущим, однако отличается способом кластеризации[14];
- алгоритм Кристофидеса-Мингоззи-Тосса [1, 2].

Метаэвристические подходы

Данная группа пересекается с некоторыми вышеуказанными подходами. Метаэвристика не представляет собой законченный алгоритм, готовый к использованию, а набор практик и стратегий, с различными параметрами, задав которые можно получить готовый алгоритм. Также можно обнаружить подходы, которые требуют преобразования условия задачи или иных действий. Метаэвристики основаны на явлениях природы или поиске с исключениями.

Основными преимуществами данного подхода является возможность обхода локальных минимумов.

Примеры.

- Генетический алгоритм – алгоритм оптимизации, производя имитацию естественного отбора в популяции допустимых решений задачи. Данный алгоритм способен работать с большим набором ограничений, алгоритм будет кратко описан ниже, а подробнее в [3, 16, 17].
- Алгоритм Османа – алгоритм оптимизации, основанный на поиске с исключениями. Более подробно рассмотрим его ниже, а также в [1, 18].
- Алгоритм Генро-Герца-Лапорте – модернизация алгоритма Османа [1, 19].
- Алгоритм Тейлорда – модернизация алгоритма Османа [1, 20].
- Алгоритм Ксю-Келли – модернизация алгоритма Османа [1, 21].
- Алгоритм Риго-Рокарола – модернизация алгоритма Османа [1, 32].
- Муравьиный алгоритм – алгоритм оптимизации, основанный на моделировании поиска ресурсов муравьиной колонией, подробнее будет рассмотрен ниже или в [22, 30].
- Моделируемый или детерминированный отжиг – эвристический подход, основанный на процессах в кристаллической решетки металла при охлаждении его до точки плавления. Будет рассмотрен ниже, а также в [1, 21].

Описание программы

Общее описание

Программа для решения задачи состоит из нескольких частей:

1. Получение данных.
2. Преобразование и фильтрация данных.
3. Кластеризация данных.
4. Построение первичных опорных решений для каждого кластера.
5. Улучшение первичных решений.
6. Объединение кластеров в общее решение и его улучшение.
7. Прочие алгоритмы, не использованные в построении решения.
8. Отдельно: возможные способы улучшения программы и алгоритмов.

Далее подробно опишем каждую из частей.

Получение данных

Выгрузка необходимых данных, а именно координат площадок ТБО, происходила с сайта администрации и городской думы города Краснодар [7]. Данные были представлены таблицами в формате EXCEL разбитыми по административный округам: Западный, Карасунский, Прикубанский, Центральный, их расположение показано ниже.

Информация о площадке ТБО состоит из адреса, количества контейнеров, объема каждого контейнера и необходимой частоты опустошения в неделю. В дополнение к площадкам в качестве депо был взят адрес Краснодарского автотранспортного предприятия по уборке города (г. Краснодар, ул. Рашпилевская 325).

Далее из сервиса OpenStreetMap загружается файл формата graphml, из которого с использованием библиотеки osmnx языка python можем получить граф дорог. Применив алгоритм Флойда-Уоршелла, построим матрицу расстояний между всеми вершинами и сохраним информацию только об интересующих нас вершинах.

Преобразование и фильтрация данных

В данных была информация обо всех площадках ТБО, которые находятся в юрисдикции Краснодарской администрации. Однако не все из них относятся непосредственно к городу. Вокруг Краснодара находится несколько городов-спутников (хуторов, поселков, станиц и прочего), как и с большинством других городов. Но зачастую они находятся на удалении от нескольких сотен метров до нескольких километров, поэтому обрабатываться они должны отдельно. все подобные площадки были исключены из общего числа данных. Тем самым объем данных сократился с более чем 2500 вершин до 1470.

Информация о координатах всех объектов была также сохранена, но после преобразования из всех координат были вычтены координаты депо и проведена нормализация, чтобы разница между минимальной и максимальной координатой равнялась 100.

Кластеризация данных

Все вершины были разделены на 7 кластеров по дням недели. Иерархическая кластеризация осуществлялась дивизивным (*англ.* divisive) методом с изначальным кластером равным всему множеству вершин.

Может показаться, что более эффективно взять изначальное распределение по округам. Однако это неверно потому, что в Краснодаре, как и в большинстве городов, разбиение на округа и районы не делит город на равные области, будь то по численности населения или площади. Так, например, Прикубанский округ по площади превосходит Центральный и Западный вместе и сравним с ними по численности населения. Также деление происходит не по географическим признакам, а по улицам. В результате чего 2 площадки ТБО, расположенные на разных сторонах одной улицы с расстояние в 20 метрах друг от друга, будут отнесены к разным кластерам. А сами полученные кластеры могут получиться несбалансированными.

Поэтому в качестве изначального набора кластеров было взято все множество вершин как один кластер.

Из-за того, что у нас к вершинам дана только матрица расстояния между ними, то кластеризация происходила методом дихотомического деления с модификацией для сохранения баланса кластеров и суммарному потреблению всех вершин кластера.

Алгоритм дихотомической кластеризации:

1. Находим 2 вершины с максимальным расстоянием между ними среди всех вершин текущей группы. Данные вершины послужат началом 2 новым кластерам.
2. Если остались еще нераспределенные вершины, то для каждой из них вычисляются расстояния до кластеров, равные расстоянию ближайших вершин до каждого из кластеров. Если вершин не осталось переходим на шаг 4.
3. Далее ищутся ближайшие вершины к каждому кластеру и приписываются к ним, после чего переходим на предыдущий шаг.
4. Смотрим из расчета суммарного потребления полученных кластеров не превосходит ли их потребление вместимости всех ТС,

предназначенных для обслуживания данного кластера и можем ли мы составить маршрут по ним жадным алгоритмом.

5. Если потребление кластеров превосходит максимальный вывозимый транспортными средствами объем, то выполняем перебалансировку кластеров, переназначая вершины из большего к меньшему.
6. В случае если перебалансировка невозможно вся программа завершает работу. Однако это не означает, что маршрут невозможно составить. А значит лишь то, что алгоритм не предназначен для такого набора данных.

Алгоритм имеет сложность $O(n^2)$, но в случае необходимости множества перебалансировок может возрасти до $O(n^3)$. Однако мы предполагаем, что компания по вывозу отходов имеет излишне превосходящие транспортные ресурсы на случай поломок или неисправностей ТС. В таком случае большое количество перебалансировок или вовсе невозможность данного алгоритма кластеризовать вершины крайне маловероятны.

Ранее было сказано, что некоторые из площадок ТБО необходимо посещать несколько раз в неделю. Для вершины, соответствующих данным площадкам, создадим несколько копий, равное необходимой частоте посещения. Назначим копии другим кластерам так, чтобы разница в днях посещения соответствовала требованиям, описанным выше, чтобы исключить случаи, когда одна площадка посещается 3 дня подряд, а потом не посещается вовсе

Построение первичных опорных решений для каждого кластера

Построение проводилось 4 алгоритмами: заметания, жадным, Кларка-Райта и алгоритмом на муравьиных колониях.

Алгоритм заметания

Данный алгоритм имеет существенное отличие от всех, которые будут описаны далее. Отличие заключается в том, что в алгоритме не используется матрицы расстояний, а только данные о координатах после преобразования описанного ранее.

1. Все вершины сортируются по увеличению угла от депо.
2. Далее в отсортированном порядке все вершины добавляются в маршруты для ТС, начиная с наименьших по расходу топлива, до заполнения данного ТС.
3. Если остались нераспределенные вершины, то они назначаются в те ТС, где осталось свободное место. В случае невозможности распределить все вершины алгоритм завершает работу.

   Далее получаем набор маршрутов ассоциированных с типом ТС.

Параметр алгоритма: процент заполнения ТС, после которого мы считаем его полным и переходим к заполнению нового ТС.

Жадный алгоритм

1. ТС средства сортируются по расходу топлива от меньшего к большему.
2. Для каждого ТС в указанном порядке назначен ближайшую к нему вершину, которая не превысит вместимость ТС.
3. Если остались нераспределенные вершины, то алгоритм завершает работу.

   Результат работы аналогичен предыдущему.

Параметр алгоритма: процент заполнения ТС, после которого мы считаем его полным и переходим к заполнению нового ТС.

Алгоритм Кларка-Райта

Данный алгоритм основан на идее “сбережения” или сокращения стоимости. В процессе работы, начиная с назначения каждой вершины отдельному маршруту, производит слияние пары маршрутов до тех пор, пока

это сокращает общую длину всех маршрутов или до достижения определенного числа маршрутов.

Подсчет матрицы сокращения при слиянии маршрутов для пары маршрутов $r_1 = \{0, \ldots, v, 0\}$, $r_2 = \{0, u, \ldots, 0\}$:

$$s_{v,y} = c_{v,0} + c_{0,u} - c_{v,u}.$$

Иначе говоря: разница стоимости при удалении двух ребер до депо с начала одного и конца другого маршрута и добавлении ребра между этими концами. Стоит заметить, что сокращение не зависит от наполнения маршрута, за исключением конца одного и начала другого.

Еще надо отметить важное свойство данного алгоритма. Кроме решения непосредственно ЗМТ, он также может решать задачу коммивояжера, если производить слияние до единственного оставшегося маршрута.

1. Используя свойство матрицы сокращения, можем подсчитать все $s_{i,j}$ заранее.
2. Линеаризуем матрицу сокращений и отсортируем в порядке уменьшения сокращения.
3. Задаем маршруты, состоящие из единственной вершины-клиента для всех вершин кроме депо.
4. Производим слияние всех маршрутов, пока не останется один. Из которого получим порядок верши-клиентов.
5. Далее действия аналогичны алгоритму заметания. Вершины добавляются в маршруты для ТС, начиная с наименьших по расходу топлива, до заполнения данного ТС. Однако есть небольшая модификация, если следующая вершина для добавления слишком далеко, то ТС заканчивает маршрут. Параметр отвечающий за это действие подбирается автоматически.
6. Если остались нераспределенные вершины, то алгоритм завершает работу.

Параметр алгоритма: процент заполнения ТС, после которого мы считаем его полным и переходим к заполнению нового ТС.

Определение далекой вершины: вершина считается слишком далеко, если вес ребра больше средней длины ребер матрицы расстояний, умноженного на некоторый коэффициент, который подбирается автоматически.

Алгоритм на муравьиных колониях

Данный алгоритм является наиболее ярким представителем метаэвристических алгоритмов, основанных на природных явлениях.

При наблюдении поведения муравьиной колонии обнаружилось, что муравьи способны довольно быстро находить оптимальные пути к источнику пищи, несмотря на то, что каждый отдельный муравей ведет себя случайным образом при поиске пищи.

В процессе поиска муравей отмечает свой путь феромоном, количество которого зависит как от длины маршрута, так и от количества и качества найденной пищи. Данный феромон своим запахом привлекает остальных муравьев, однако они не следуют ровно по маршруту с наибольшим объемом феромона, а выбирают пути случайно в зависимости от объема феромона на каждом из возможных путей.

На самых коротких или выгодных маршрутах муравей оставит наибольшее количество феромона, благодаря чему со временем на оптимальных маршрутах скапливается достаточное количество феромона, значительно превосходящее объем феромона на других.

Проведем аналогию муравьиной колонию с нашей задачей. Муравьи будут разделены по типам, каждый соответствует одному типу ТС, а муравьи способны различать феромоны только муравьев своего типа. Депо будет муравейником, из которого все муравьи будут начинать свой путь и при сборе достаточного объема еды (заполнения вместимости ТС) муравей будет возвращаться обратно.

Будем считать, что если какой-либо муравей посетил одну из вершин кроме депо, больше ни один другой ее посещать не захочет, а вершина станет недоступной, так как в ней иссяк запас еды. Желание муравья посетить одну из доступных вершин пропорционально количеству феромона на пути и обратно пропорционально длине этого пути.

Правило перехода: определение следующего шага каждого муравья зависит от его желания (вероятности, шанса) посетить ту или иную вершину:

S– сумма всех желаний перехода для всех доступных вершин и всех типов муравьев.

$$P_{t,i,j} = f_{t,i,j}^{\alpha} * l_{i,j}^{\beta} / S,$$

где t– тип муравья;

i, j– индексы вершин; iпринимает значения тех вершин, где находятся муравьи соответствующего типа, а j– всех доступных (непосещенных ранее) вершин;

$F_t = \{f_{t,i,j}\}$– матрица распределения феромона муравьев типа t по ребрам переходов;

$L_t = \{l_{i,j}\}$, $l_{i,j} = 1/c_{i,j}$– матрица обратных расстояний, элементы которой обратно пропорциональны расстоянию между соответствующими вершинами;

α, β – параметры алгоритма описывающие приоритет между количеством феромона и расстоянию при определении желания посетить ту или иную вершину. Также эти параметры описывают равномерность распределения вероятностей между разными вершинами и типами. Увеличение обоих параметров приведет к увеличению вероятности посещения наиболее желанных вершин.

После каждой итерации испаряется некое количество феромона и добавляется новая порция: $\Delta F_t = \{\Delta f_{t,i,j}\}$; $\Delta f_{t,i,j} = Q * l_{i,j} * count_{t,i,j}$ – матрица прибавки феромона, Q – константа, которая должна зависеть от

длины ребер графа, и как раз функция определения этой зависимости является параметром. Пример функции: константа, среднее длина ребер, средняя длина ребер в маршрутах, построенных жадным алгоритмом.

$count_{t,i,j}$ – число посещения ребра $(i,\ j)$муравьем типа t.

$F_t = F_t * p + \Delta F_t$; p– коэффициент испарения феромона.

Также параметрами являются:

$base\ pheromon$– изначальное количество феромона;

$min\ pheromon$– минимальное количество феромона;

$max\ pheromon$– максимальное количество феромона;

$count\ iter$– количество итераций перед пересчетом матриц феромона;

$time\ work$ – общее времени работы, так как критерия завершения нет;

$criterion\ start$ – критерий запуска нового муравья, объем заполненности, после превышения которого запускается новый муравей данного класса;

$criterion\ end$ – критерий возвращения муравья в муравейник, объем заполненности, после превышения которого данный муравей отправляется в депо.

Алгоритм:

1. Заполнение матриц феромона, считаем наилучшую длину маршрутов равной бесконечности.
2. Инициализация муравьем по одному от каждого класса (в целях увеличения производительности не производится запуск всех возможных муравьев каждого класса, а только по одному представителю каждого типа).
3. Помечаем все вершины-клиенты как непосещенные.
4. Вычисление $P_{t,\ i,\ j}$для каждого из активных муравьев и доступных вершин.
5. Генерация случайного числа, на основе которого определяется следующий шаг алгоритма – номер муравья и куда он перейдет.

6. Меняем положение выбранного муравья на выбранную вершину, которая помечается посещенной.
7. Проверка не превзошел ли данный муравей критерий старта нового муравья и окончания маршрута текущего. Если муравьи данного класса закончились, то новый не запускается.
8. Если остались непосещенные вершины, то вернемся на шаг 4.
9. Если все вершины посещены, считаем длину полученных маршрутов. В случае получения длины меньше наилучшей, обновляем лучший результат и сохраняем маршруты.
10. Если число итераций превысило $count\ iter$, пересчитываем матрицы феромонов.
11. Если время работы не превысило $time\ work$, Очищаем маршруты и переходим на шаг 2.
12. Если время работы превысило $time\ work$, завершаем работу, возвращаем наилучший результат.

Данный алгоритм зачастую находит лучшее решение, по сравнению с предыдущими алгоритмами. А случайность процесса нахождения решений и наличия минимального объема феромона позволяет выходить из локальных минимумов.

Большое количество параметров делает данный алгоритм очень гибким, а при некоторых наборах он становится аналогичен жадному, тем самым содержит в себе его решения. Однако это является и минусом, поскольку достаточно трудоемко производить поиск оптимальных параметров.

Также об алгоритме на муравьиных колониях было написано в [22, 30].

Улучшение первичных решений

На прошлом этапе для каждого кластера были получены наборы маршрутов одним из описанных ранее методов. Однако их можно улучшить другими алгоритмами. Улучшение будет производиться двумя поочередными методами:

1. Модифицированным алгоритмом Османа (Генро-Герца-Лапорте);
2. Алгоритм локальной оптимизации в чередующихся окрестностях.

Модифицированный алгоритмом Османа

Алгоритм Османа основан на поиске с исключениями, которые определяют запрещенные стратегии поиска решения, тем самым ускоряя поиск решения.

Главным преимуществом данного подхода является возможность выхода из точек локального минимума. На каждом шаге выбирается лучшее решение из окрестности текущего (лучшее, не включая текущее). Однако существует вероятность зацикливания между несколькими решениями. Для этого и вводится список исключений, позволяющий обходить такие циклы, по крайней мере если их длина ограничена некоторым параметром алгоритма. В противном случае потребуется запоминание каждого пройденного решения, что невозможно ввиду NP-полноты задачи.

Алгоритм состоит из нескольких частей:

1. Функция задающая параметры алгоритма: время работы или критерий остановки, длины списка запрета и прочие параметры специфичные для различных модификаций (детерминированного или недетерминированного отжига и других).
2. Определение вида функции запрета: Запрещаться может как комбинация нескольких маршрутов, так и факт принадлежности некоторых вершин к определенным маршрутам.
3. Функция определяющая окрестность текущего решения.
4. Функция выбора решения из окрестности (случайный выбор, выбор лучшего решения, метод отжига).

Все эти части в совокупности и определяют ту или иную модификацию алгоритма Османа. Поэтому при его кодировании рекомендуется эти части делать обособленными, чтобы была возможность легко модифицировать алгоритм для поиска лучшего решения.

Алгоритм:

1. Инициализация параметров, констант, текущего и лучшего решения, равных полученному как параметр алгоритма.
2. Построение окрестности текущего решения, элементы которой ассоциированы с изменением длины маршрутов по сравнению с текущим решением.
3. Получение допустимого списком запрета решения из окрестности согласно функции выбора.
4. Обновление текущего решения, обновление списка запрета согласно выбору этого решения. Обновление может не произойти, если полученное решение не принято. Процедура принятия описана ниже.
5. Удаление из списка запрета последней записи, если его длина превосходит список запрета.
6. Если длина текущего решения короче лучшего, то обновить лучшее решение.
7. Если критерий остановки не выполнен, перейти на шаг 2.
8. Если критерий остановки выполнен, вернуть текущее лучшее решение.

Опишем подробнее какие составные части алгоритма были выбраны.

Длина списка запрета вычисляется по формуле, взятой из работ [1, 18], время работы алгоритма ограничено 40 секундами для каждого кластера.

Окрестность решения равна всем решениям полученным из текущего путем обмена вершин между парой маршрутов или переносом одной вершины в другой маршрут, если это не приведет к переполнению вместимости соответствующего ТС.

Список запрета состоит из записи вершины и откуда она перешла, тем самым запрещая ее возврат обратно.

Выбор решения из окрестности происходит путем выбора решения с наименьшей длиной окрестности. В дальнейшем происходит принятие или непринятие данного решения-кандидата согласно методу детерминированного отжига.

Пусть R_i, R_{i+1}– текущее и решение-кандидат, чтобы стать новым текущим. $L(R)$– функция подсчёта длины всех маршрутов решения.

Если $L(R_{i+1}) < \gamma L(R_i)$, то решение принимается, В противном случае алгоритм преждевременно завершает работу. γ – параметр отжига. Неправильный подбор данного параметра может привести к преждевременному завершению работы или к холостой работе и поиску “далеко” от оптимального решения.

Алгоритм локальной оптимизации в чередующихся окрестностях

Данный алгоритм был разработан лично автором и вдохновлен Алгоритмами Османа, VND-STS и Adaptive Large Neighborhood Search, подробнее о них можно узнать в [1, 6, 18, 27].

Основная идея состоит в улучшения решения поиском в случайной окрестности текущего решения. Конкретная окрестность поиска определяется путем случайного выбора одной из изначально заданных функций определения окрестности. Вероятность выбора той или иной функции зависит от ее эффективности по поиску лучшего решения и меняется динамически в процессе работы.

Решение о принятии или непринятии решения из окрестности аналогична предыдущему алгоритму (методом детерминированного отжига).

Алгоритм:

1. Инициализация параметров, констант, текущего и лучшего решения, равных полученному как параметр алгоритма.
2. Инициализация множества функций определения окрестности и базовых вероятностей их выбора.
3. Случайный выбор функции построения окрестности.
4. Получение лучшего решения из окрестности согласно выбранной функции построения.

5. Обновление текущего решения, если полученное решение принято согласно параметру отжига γ. В отличие от алгоритма Османа, в случае непринятие завершение алгоритма не происходит.
6. Если длина текущего решения короче лучшего, то обновим лучшее решение и увеличим вероятность выбора функции построения окрестности, давшей это решение.
7. Если критерий остановки не выполнен, перейдем на шаг 3.
8. Если критерий остановки выполнен, вернем текущее лучшее решение.

Функции построения окрестности, вошедшие в итоговое решение:

1. 2-opt.
2. Перенос одной вершины в рамках одного маршрута.
3. Перенос одного ребра (пары соседних вершин) в рамках одного маршрута.
4. Обмен двух вершин между в рамках одного маршрута.
5. Обмен двух несмежных ребер между в рамках одного маршрута.
6. Перенос одной вершины из одного маршрута в другой.
7. Обмен двух вершин между маршрутами.

Объединение кластеров в общее решение и его улучшение

Ранее говорилось, что все множество вершин было разделено на 7 кластеров. А также, что одна вершина может принадлежать нескольких кластерам, если площадка ТБО, соответствующая данной вершине, требует посещения несколько раз в неделю.

Как раз такие вершины значительно увеличивают длину построенных маршрутов. Поэтому необходима оптимизация маршрутов всех кластеров вместе взятых.

Улучшение будет производится модификацией предыдущего алгоритма, а именно алгоритм локальной оптимизации в чередующихся окрестностях.

Модификация заключается в изменении некоторых функций построения окрестностей: перенос одной вершины из одного маршрута в другой и обмен двух вершин между маршрутами. Так как именно они могут переносить вершину из одного кластера в другой. Остальные окрестности меняют только один маршрут, поэтому их модификация не требуется.

В случае, если перенос или обмен вершин производится между двумя маршрутами, относящихся к одному кластеру, то функция построения окрестности работает аналогично описанным ранее.

А если данные маршруты относятся к разным кластерам, то перенос или обмен производится только тогда, когда данная вершина находится в одном кластере. Иначе говоря, если площадку ТБО соответствующую данной вершине требуется посетить только один раз. В противном случае может произойти случай, который был описан ранее: площадка ТБО посещается несколько дней подряд, а все остальные дни не посещается.

Остальные функции построения окрестности не исключаются из работы алгоритма. Так как после переноса вершин в маршрут в другом кластере последующая оптимизация данного маршрута может привести к улучшению решения.

Прочие алгоритмы, не использованные в построении решения

В данной главе будут описаны алгоритмы, реализованные в рамках программы, но не задействованные в рамках построения решения задачи. Поэтому они будут описаны кратко, с пояснениями причин почему они не были использованы.

Генетический алгоритм

Генетический алгоритм – метаэвристический алгоритм, основанный на природном явлении. В рамках алгоритма моделирования отбора оптимального решения, схожего с биологической эволюцией.

Каждое допустимое решение задачи (удовлетворяющее всем условиям и ограничениям), кодируется некоторым набором данных, называемых геном.

Алгоритм состоит из формировании популяции допустимых решений, называемых особями, определяемых их генами. Далее производится скрещивание и мутация генов случайных особей, описывающих. Следующий этап – селекция.

Скрещивание – выбираются две особи, далее производится комбинация их генов, в процессе чего получаем новый ген и новую особь.

Мутация – выбирается одна особь, ген которой случайно изменяется. В результате создается новая особь.

Селекция – отбор из популяции подмножества особей-решений, длина которых является наименьшей. Данное подмножество становится новой популяцией.

Алгоритм:

1. Формирование популяции.
2. Скрещивание нескольких особей.
3. Мутация нескольких особей.
4. Селекция.
5. Если критерий останова не выполнен, перейти на шаг 2, иначе вернуть лучшее решение из текущей популяции.

Данный алгоритм не был использован, так как за одно и то же время работы дает наименее оптимальное решение среди всех описанных выше алгоритмов.

Алгоритм VND-STS

Алгоритм VND-STS был реализован на основе работы [27].

Главной особенностью данного алгоритма является возможность формирования маршрутов с раздельной доставкой, при которой одна вершина может принадлежать нескольким (как правило одному или двум маршрутам

одного кластера). При чем такие маршруты делят между собой спрос данной вершины.

Благодаря данному свойству значительно расширяется пространство допустимых решений.

Однако это же приводит к увеличению вычислительной нагрузки. В то время как спрос каждой площадки ТБО сильно меньше вместимости ТС, что делает раздельную доставку не столь необходимой.

В результате при том же времени работы алгоритм дает более слабое решение, чем описанные ранее.

Двухэтапный алгоритм с первичной кластеризацией

Был реализован двухэтапный алгоритм с решением задачи коммивояжера методом Лина–Кернигана и разрезанием общего маршрута.

Из-за ошибки реализации итоговый результат был хуже алгоритма Кларка-Райта, а время работы было значительно больше.

Возможные способы улучшения программы и алгоритмов.

Жадный алгоритм: в качестве улучшения можно использовать ту же эвристику, используемую в алгоритме Кларка-Райта, а имено запретить построение маршрутов с слишком длинными ребрами, чтобы исключить случай, когда добавляется единственная подходящая по вместительности вершина, расположенная в противоположной части района (кластера).

Также, ввиду быстроты работы алгоритма, можно применить перебор из 2 параметров на решетке с небольшим шагом, а в дальнейшем либо использовать оптимальные параметры, либо каждый раз находить новые в случае изменения данных.

Алгоритм на муравьиных колониях: данный алгоритм имеет большое количество вещественных параметров. Поиск оптимальных перебором на решетке не представляется возможным, ввиду долгой работы и

большой размерности параметров. поэтому необходимо определить стратегию подбора параметров.

Алгоритм Лина–Кернигана: правильная реализация решения задачи коммивояжера может значительно улучшить как поиск первичного решения и заменить алгоритм Кларка-Райта. Так и улучшение конечного, когда состав каждого маршрута уже не будет далее изменяться применение алгоритма Лина-Кернигана к каждому маршруту приведет к лучшему решению.

Генетический алгоритм: основной проблемой данного алгоритма – слишком долгое схождение к субоптимальному решения на изначально случайно сгенерированных опорных решениях (популяции).

Однако, в процессе работы предыдущих алгоритмов создается большое количество хороших решений. Если использовать в качестве исходной популяции решения полученные жадным алгоритмом и алгоритмом Кларка-Райта, а также каждое решение полученное на итерациях алгоритмов Османа, на муравьиных колониях и поиска в чередующихся окрестностях, то можно получить популяцию из достаточно хороших решений, что обеспечит более быструю сходимость.

Но такие решения могут быть сильно похожи друг на друга, затрудняя операцию скрещивания. Для решения этой проблемы рекомендуется сформировать популяцию из решений полученных несколькими запусками алгоритма на муравьиных колониях с разными параметрами.

Параллелизация работы: Как было сказано ранее, все вершины делятся на 7 кластеров и для каждого из них решается ЗМТ, независимо от других. А значит их можно запустить в параллельной работе. Все алгоритмы и структуры данных были реализованы для легкой параллелизации по кластерам, а также установлены соответствующие библиотеки из комплекса библиотек C++ Boost [28].

В рамках работы параллелизация проводилась только при тестировании возможности параллельной работы. В целях облегчения тестирования и рефакторинга алгоритма параллелизация всей работы не проводилась.

Алгоритм Османа: на данный момент выбор следующего шага алгоритма происходит за счет процедуры детерминированного отжига. В рамках оптимизации стоит рассмотреть изменение ее на моделирование отжига или применение обоих процедур одновременно, а также принятие случайных решений из окрестностях.

Требуется разработать более продвинутые критерии остановки, нежели просто ожидание истечения срока таймера. В частности предполагается использовать критерий из работы [1]. Однако применения данного критерия приводит к излишне долгой работы, поэтому можно использовать аналогичные критерии. Также критерий остановки может зависеть от числа итераций без улучшения решения.

Алгоритм локальной оптимизации в чередующихся окрестностях: в качестве улучшение можно рассмотреть больше различных способов построения окрестностей, а также подбор параметров вероятностей их выбора.

Длительность работы данного алгоритма также ограничена таймером. Для сокращения времени работы программы предлагается ввести новый критерий:

1. если решения из какой-либо окрестности не были приняты, то помечаем данную окрестность и не используем ее больше.
2. если решение выбранной окрестности принято, то снимаем метки со всех окрестностей, так как изменение точки в пространстве решений изменит все окрестности.
3. если все окрестности помечены – завершаем работу.

Общее по каждому из алгоритмов: все алгоритмы работают с определенным числом маршрутов, найденных при построении первичных решений. Однако такой подход может оказаться неэффективным в поиске решения, поэтому возможность добавления фиктивных пустых маршрутов, реализуемых доступными ТС, может привести к улучшению решения, о чем будет сказано в следующей главе.

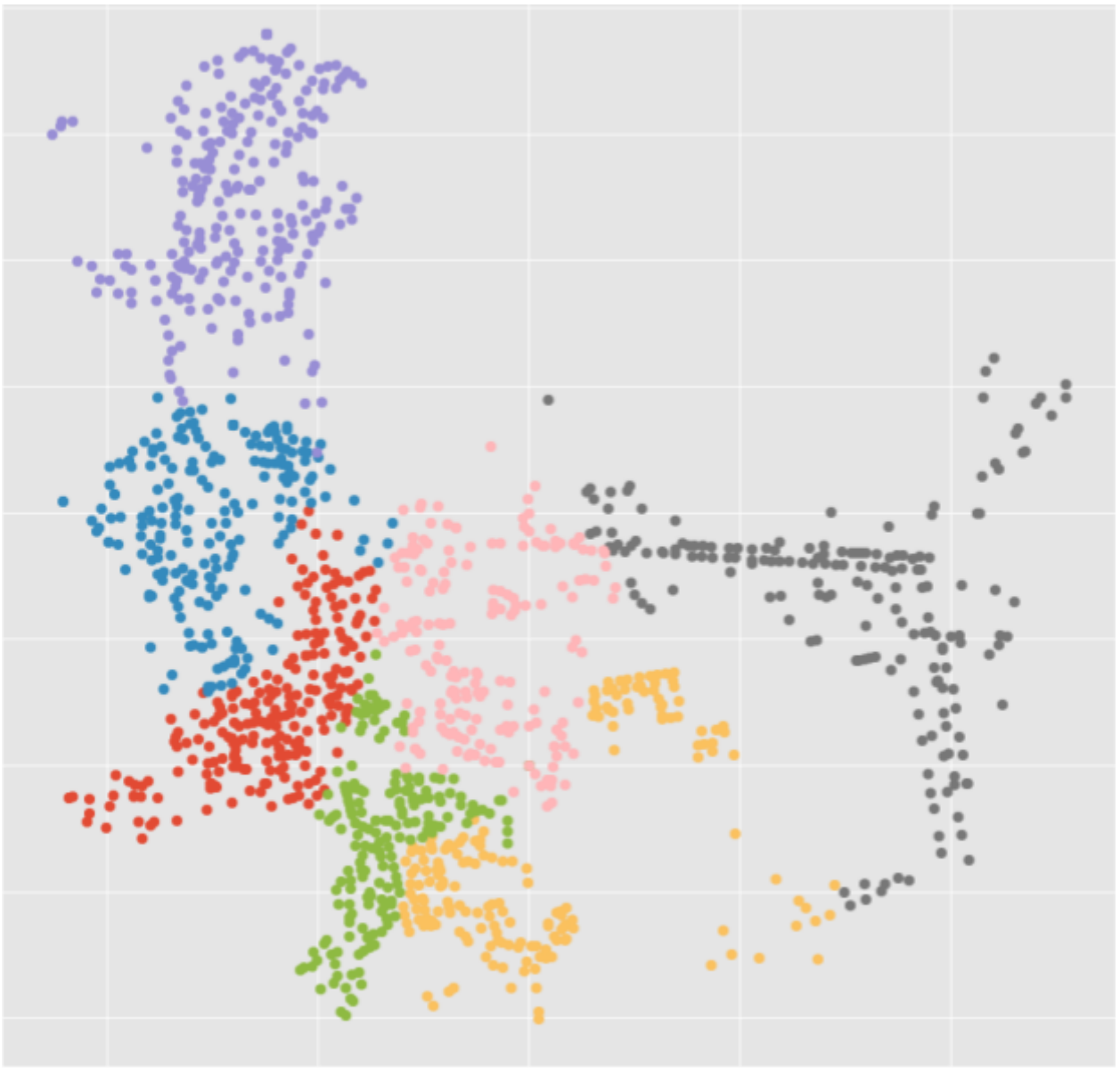
Также ранее был описан порядок действий (применение) алгоритмов в программе. Изменение этого порядка может привести к другому решению, которое может оказаться более эффективным. Так сейчас имеем порядок первичное решение, алгоритм Османа, поиск в окрестностях по каждому кластеру, поиск в окрестностях по всем кластерам. Такой порядок уводит на третьем шаге в настолько глубокую точку локального минимума, что на этапе завершения, улучшения всех кластеров, практически не происходит обмен вершин между кластерами.

В свою очередь, применение этапа поиска в окрестностях по всем кластерам сразу после получения первичных решений может привести к улучшению кластеризации. А далее необходимо будет применить алгоритмы, улучшающие каждый кластер в отдельности.

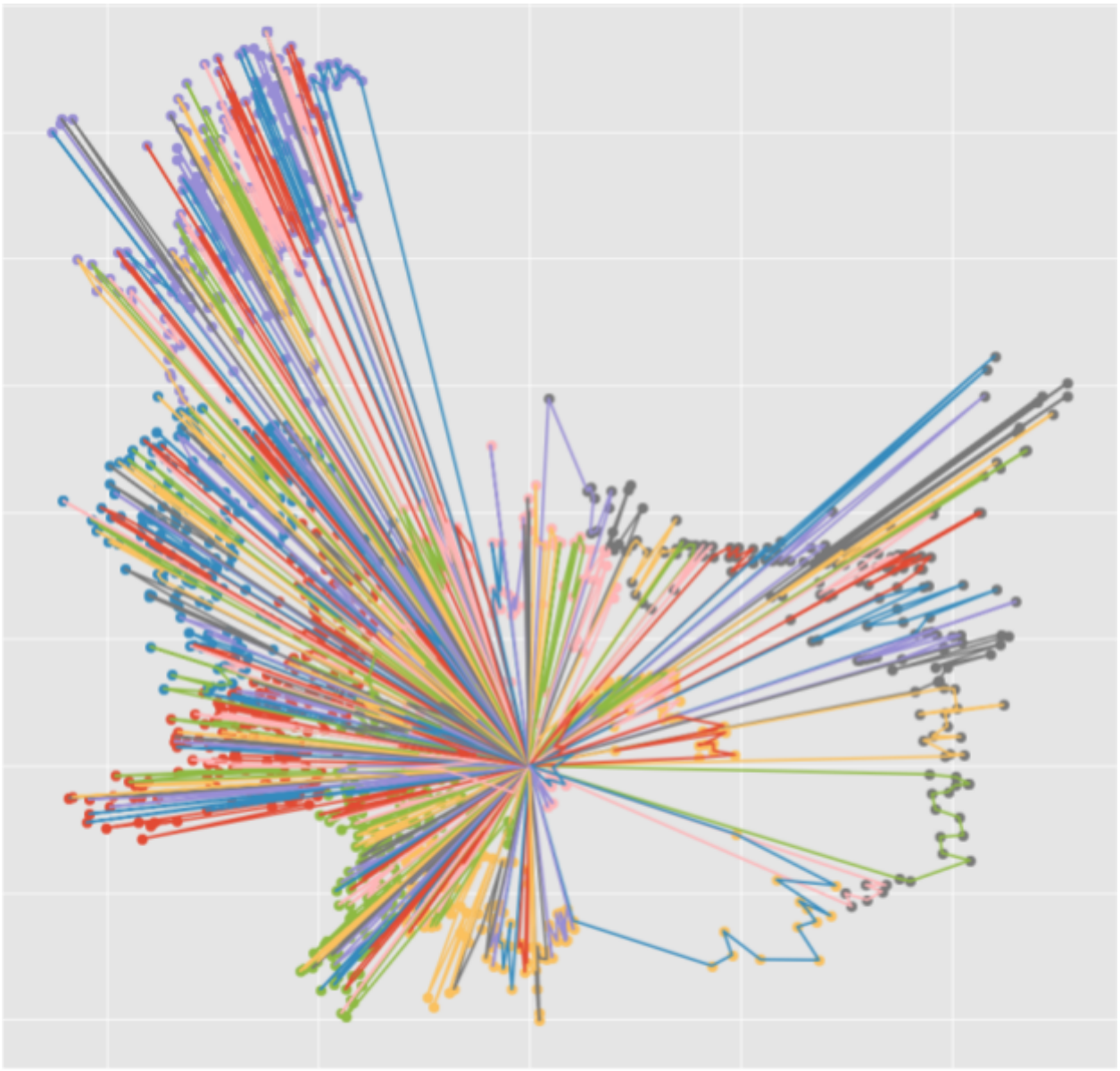
Результаты работы программы

Демонстрация вершин

Расположение вершин на плоскости



Кластеризация вершин



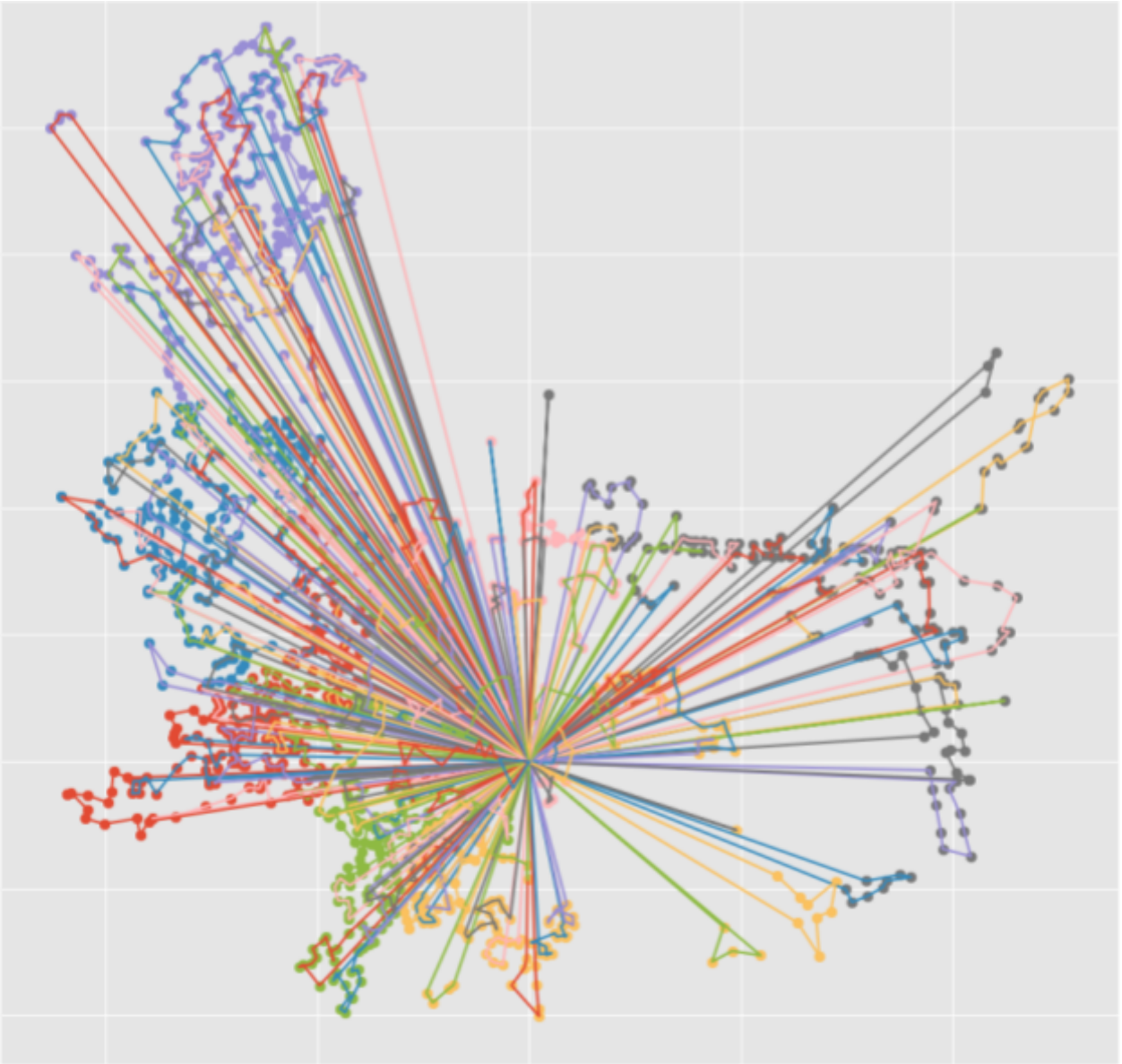
Алгоритм заметания

Параметр: процент заполнения – 99%;

Длина, у.е.: 20492.6;

Время, сек: 0.615;

Количество маршрутов: 164.



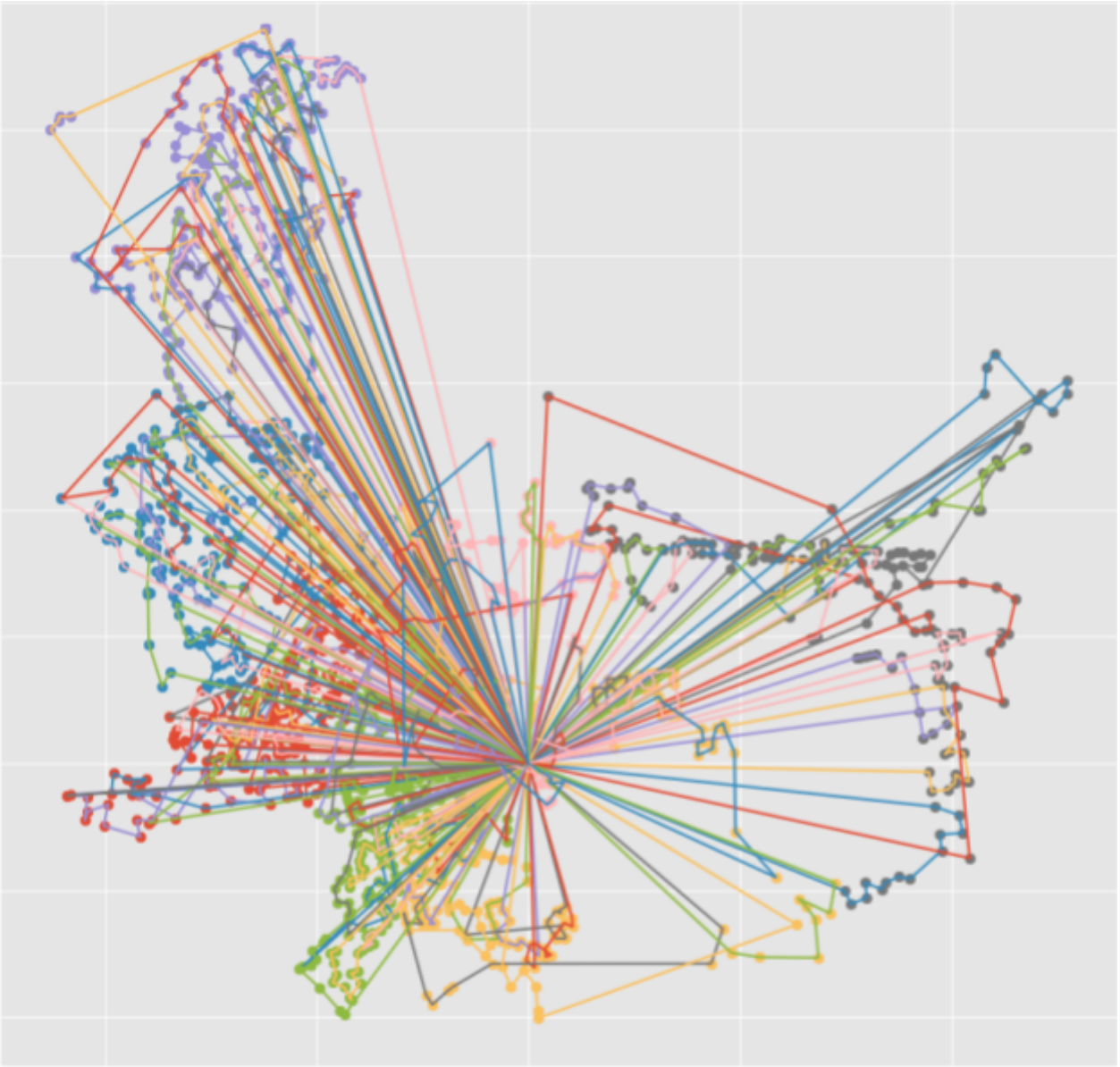
Алгоритм Кларка-Райта

Параметр: процент заполнения – 99%;

Длина, у.е.: 17021.5;

Время, сек: 9.898;

Количество маршрутов: 184.



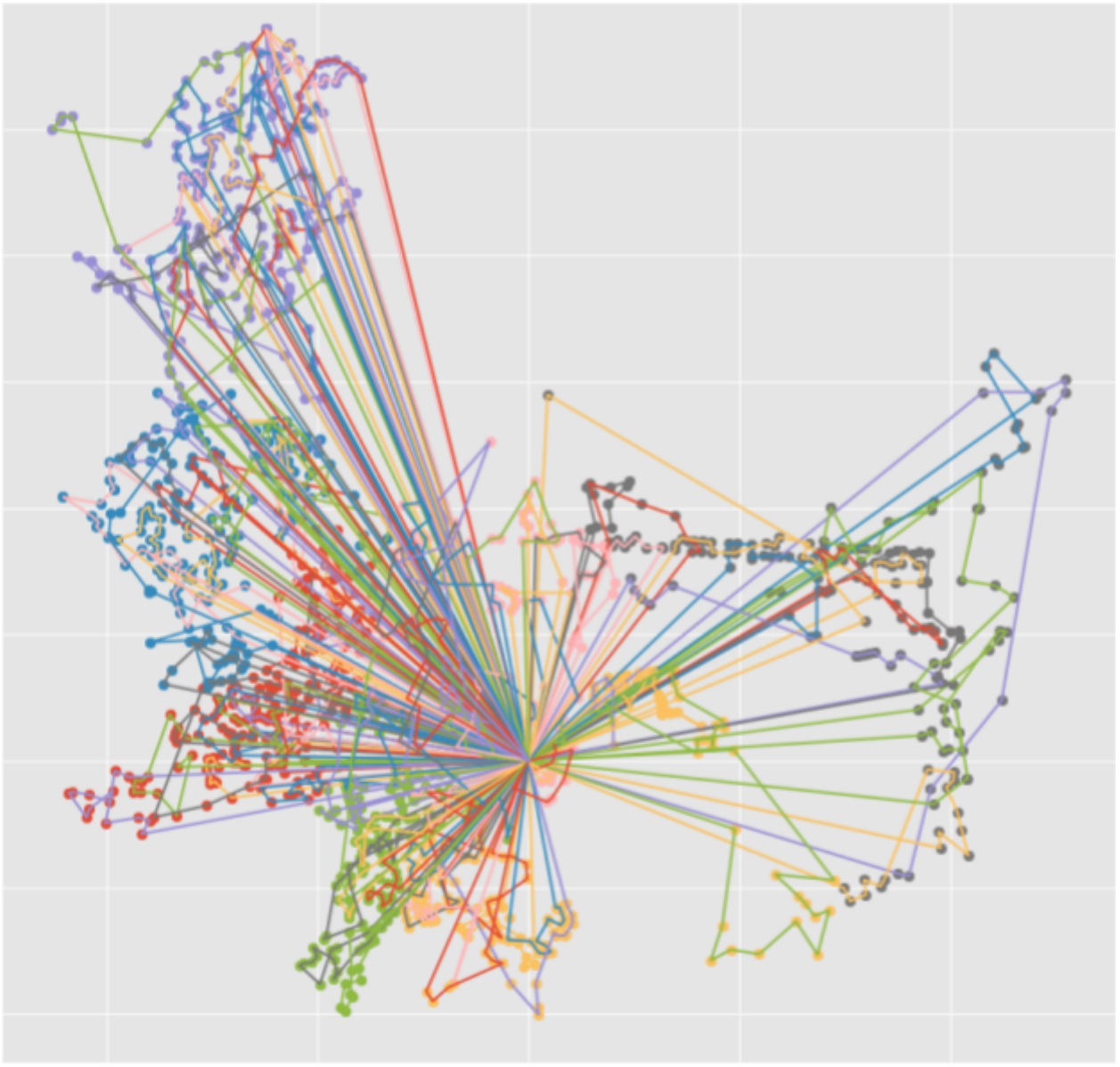
Жадный алгоритм

Параметр: процент заполнения – 99%;

Длина, у.е.: 15476.2;

Время, сек: 1.246;

Количество маршрутов: 156.



Алгоритм на муравьиных колониях

Параметры:

- $base\ pheromon$ – 0.2;
- $min\ pheromon$ – 0.1;
- $max\ pheromon$ – 0.85;
- α, β– 1, 4;
- $count\ iter$– 10;
- $time\ work$– 40 сек;
- $criterion\ start$ – запуск нового муравья того же типа, если максимальная по запросу вершина не может быть добавлена в маршрут текущего муравья;

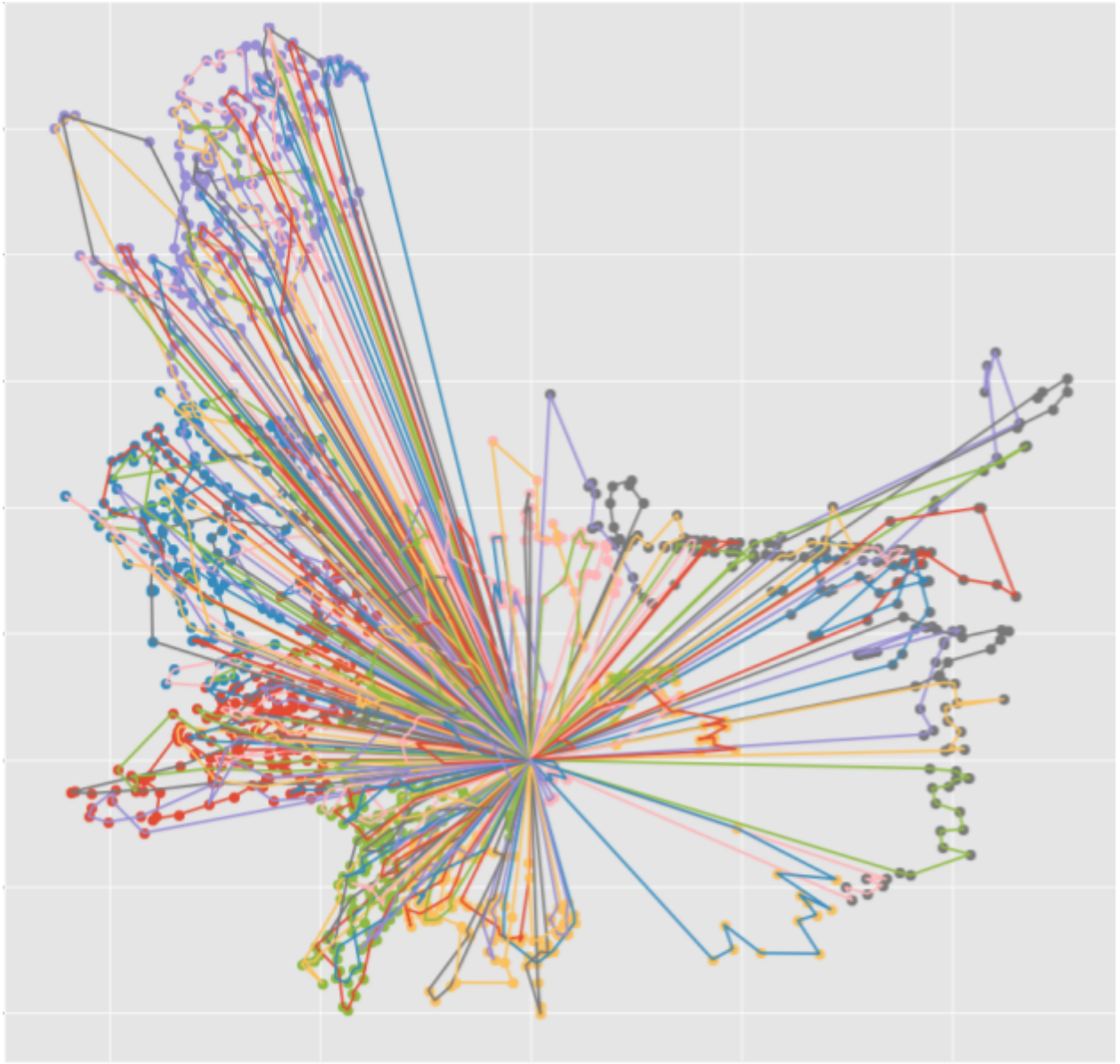
- $criterion\ end$ – возвращение муравья происходит, если минимальная по запросу вершина не может быть добавлена в маршрут текущего муравья;
- p– коэффициент испарения феромона, равен 0.8;
- Q – коэффициент добавления феромона, равен 5;

Длина, у.е.: 14493.2;

Время, сек: 282.251;

Количество маршрутов: 134.

Применение алгоритма Османа для первичных решений



Алгоритм Османа на базе алгоритма заметания

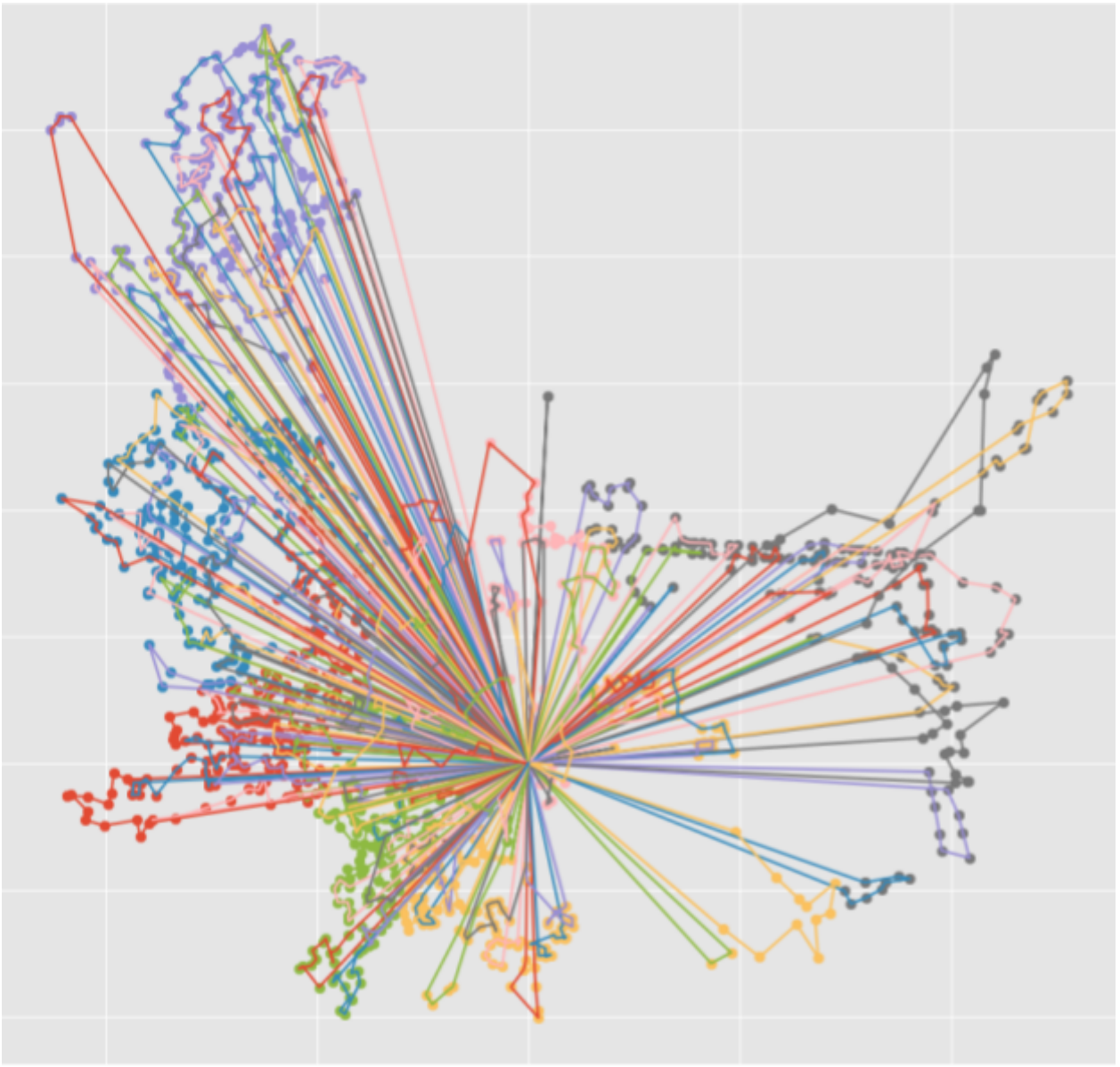
Параметры:

- γ – параметр отжига, равен 1.05;
- ts – длина списка запрета, равна $max(7.0,\ 9.6\ *\ log(count\ vertex\ *\ count\ rout)\ -\ 40)$;
- $time\ work$– 40 сек;

Длина, у.е.: 15476.2;

Время, сек: 1.246;

Количество маршрутов: 156.



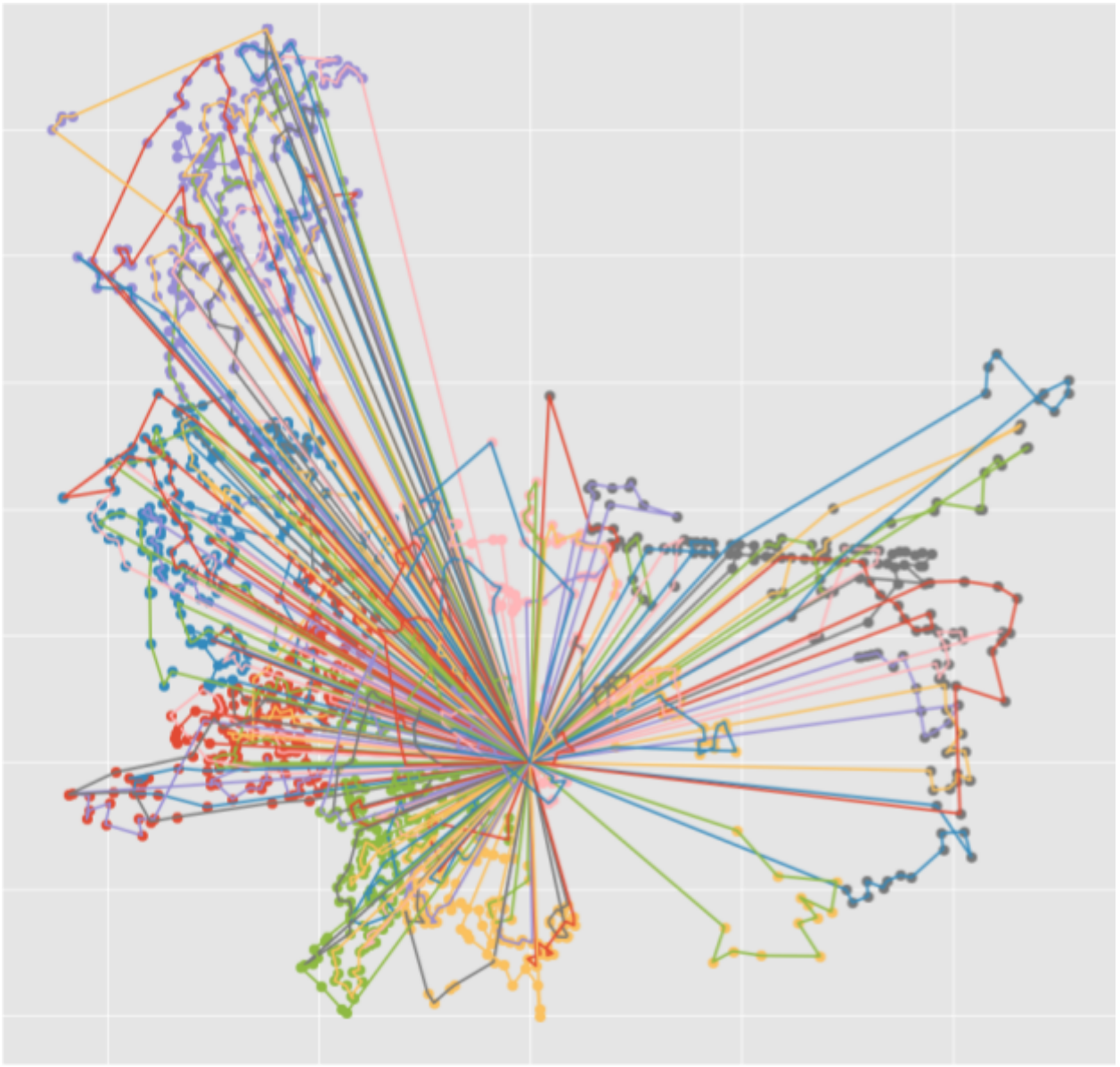
Алгоритм Османа на базе алгоритма Кларка-райта

Те же параметры;

Длина, у.е.: 16472.7;

Время, сек: 291.71;

Количество маршрутов: 184.



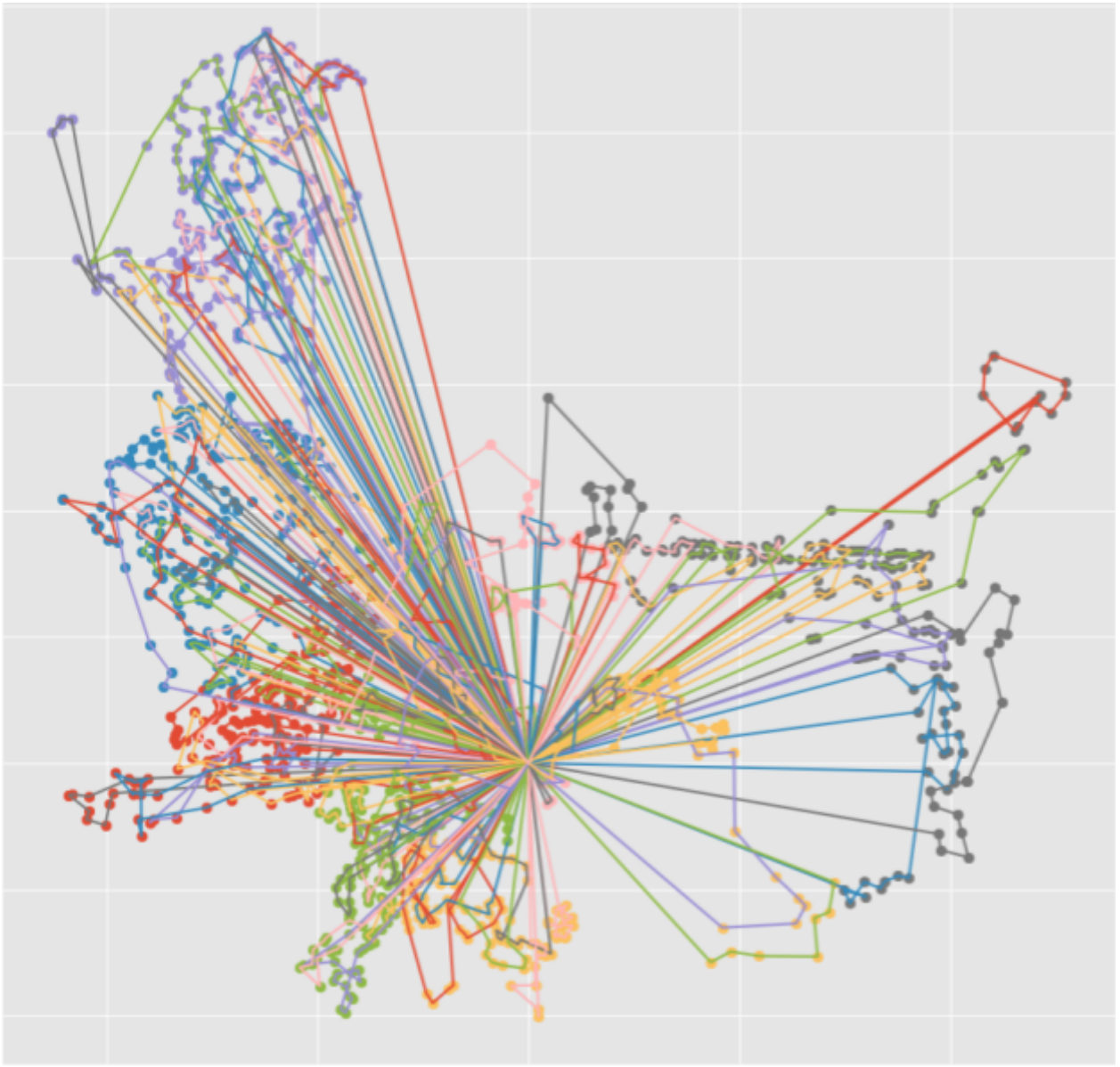
Алгоритм Османа на базе жадного алгоритма

Те же параметры;

Длина, у.е.: 14592.2;

Время, сек: 281.6;

Количество маршрутов: 156.



Алгоритм Османа на базе алгоритма на муравьиных колониях

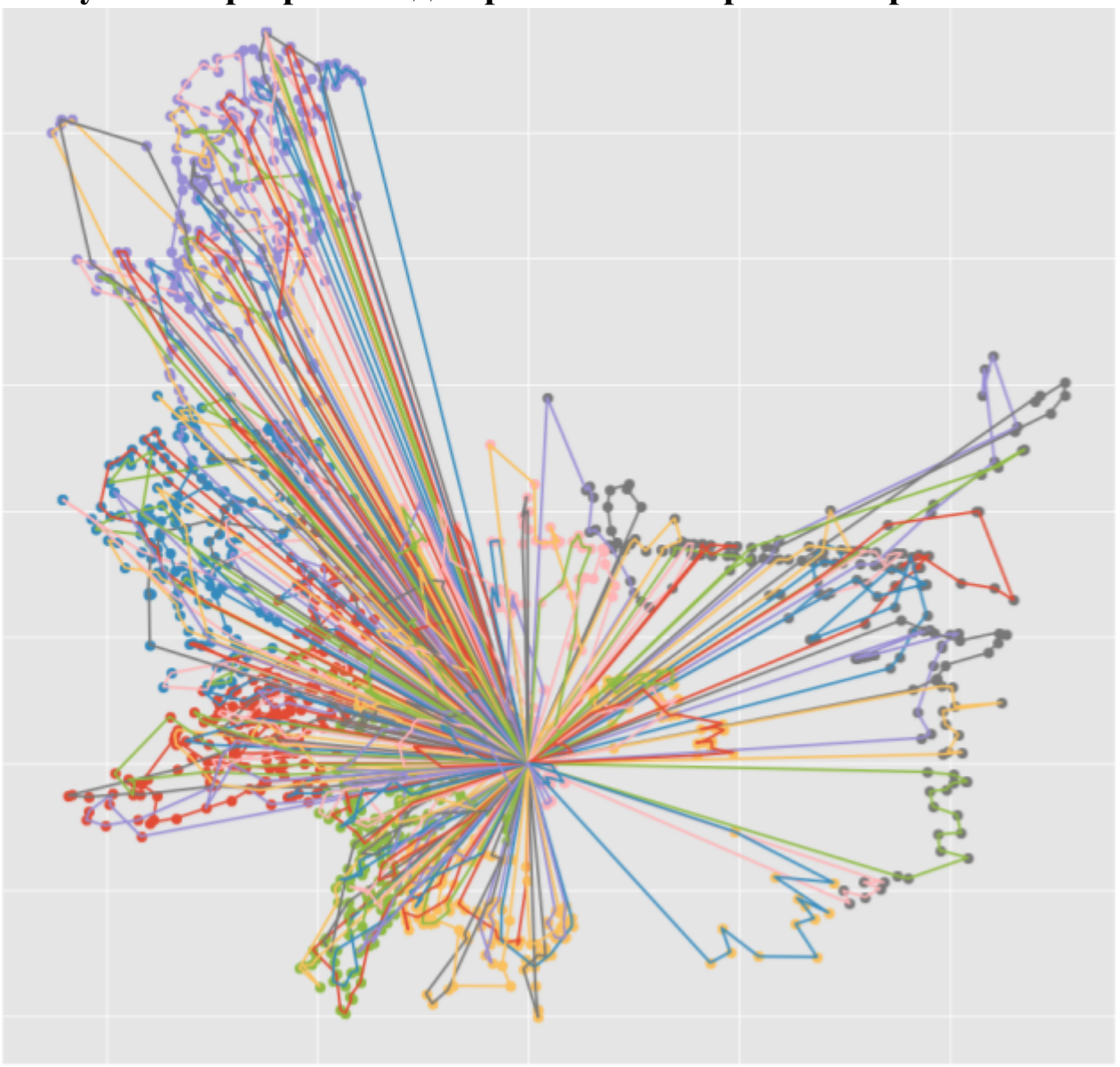
Те же параметры;

Длина, у.е.: 13863.5;

Время, сек: 562.581;

Количество маршрутов: 136.

Результат программы для различных первичных решений



Итоговый результат на базе алгоритма заметания

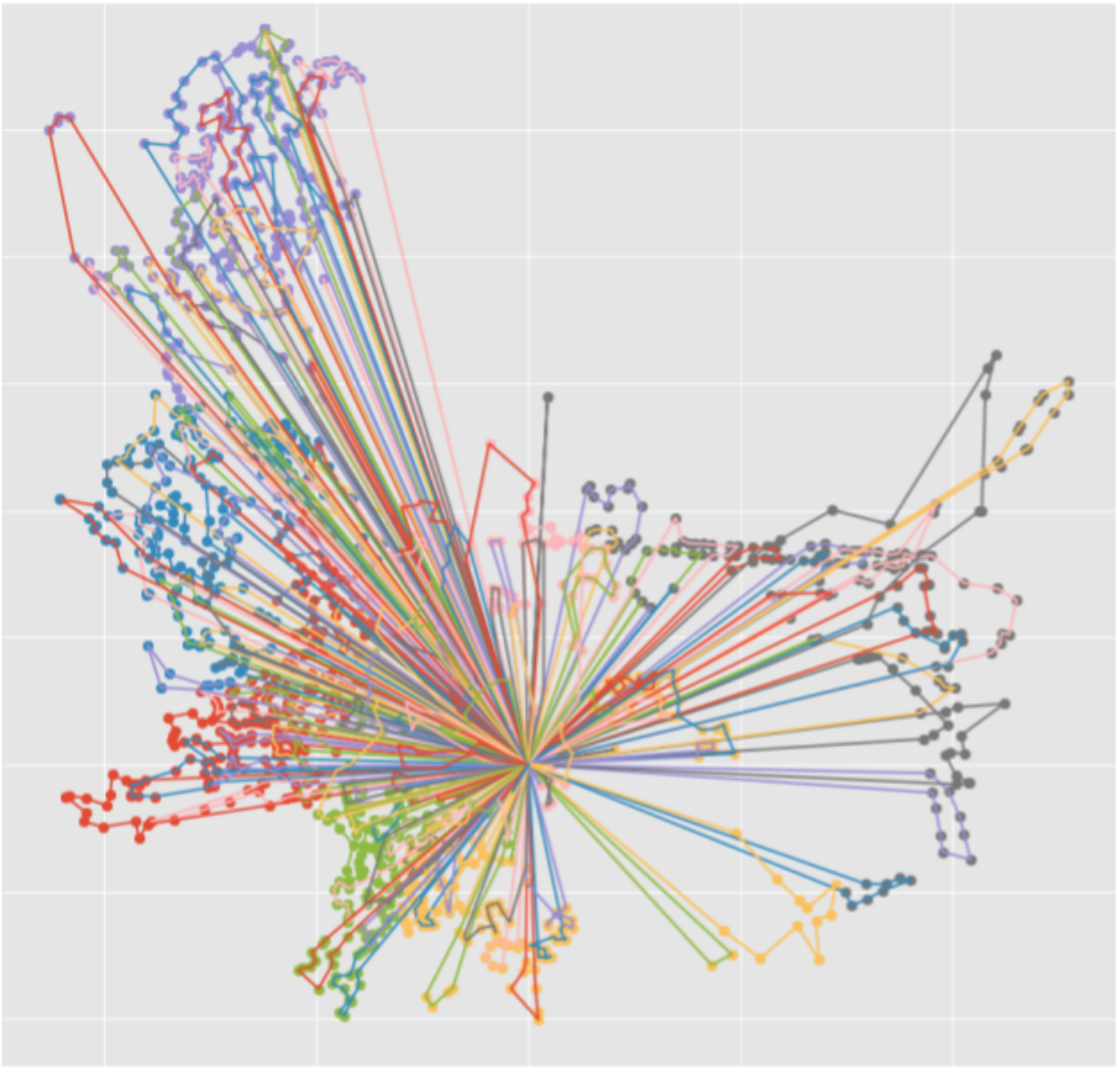
Параметры локальной оптимизации:

- γ – параметр отжига, равен 1.09;
- $prob\ neighborhood\ choose$ – начальная вероятность выбора окрестностей: 0.5 для 2-opt, равновероятно для всех остальных;
- $time\ work$– 20 сек;

Длина, у.е.: 15633.1;

Время, сек: 421.838;

Количество маршрутов: 164.



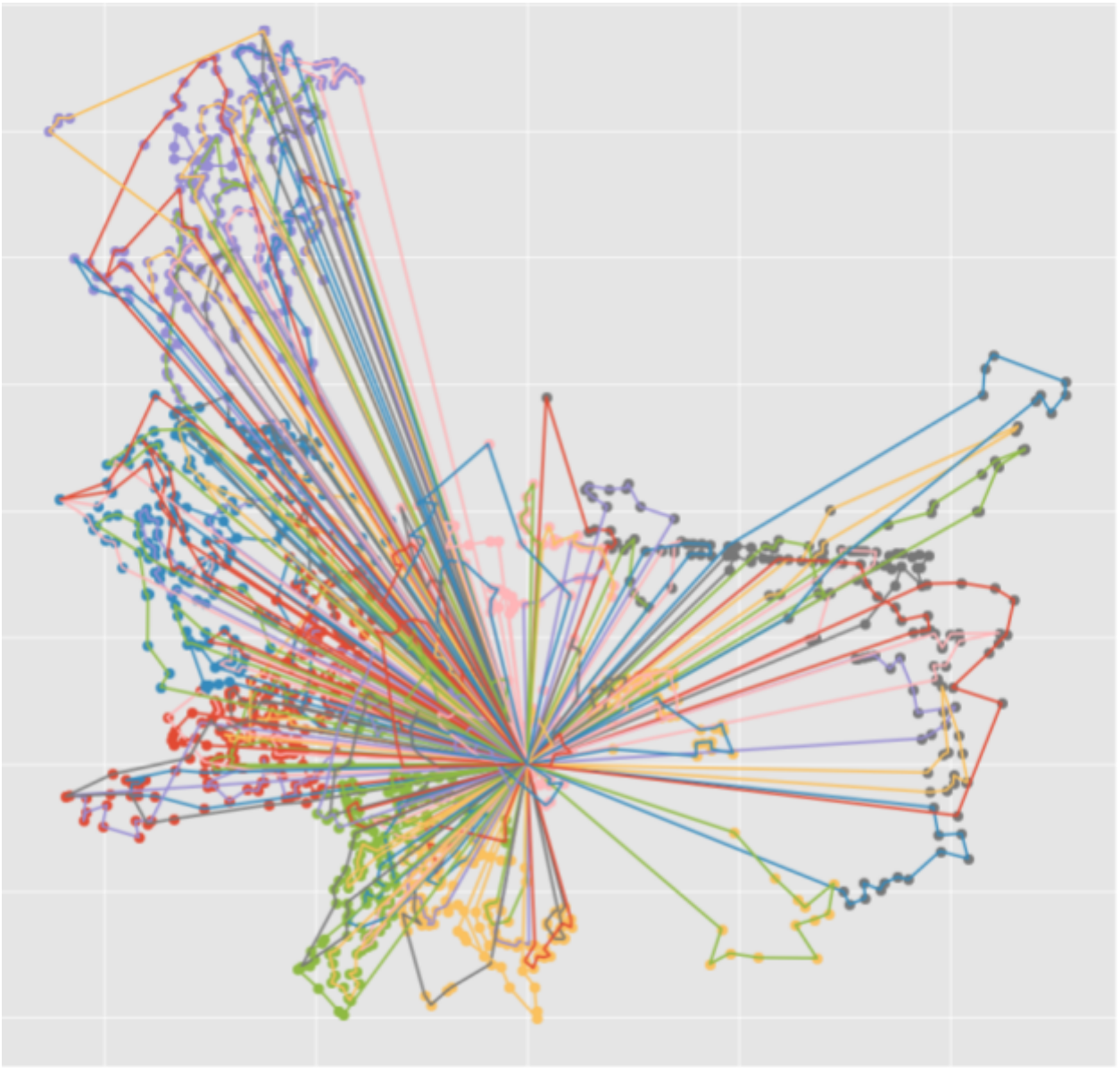
Итоговый результат на базе алгоритма Кларка-Райта

Те же параметры;

Длина, у.е.: 16357.4;

Время, сек: 431.155;

Количество маршрутов: 184.



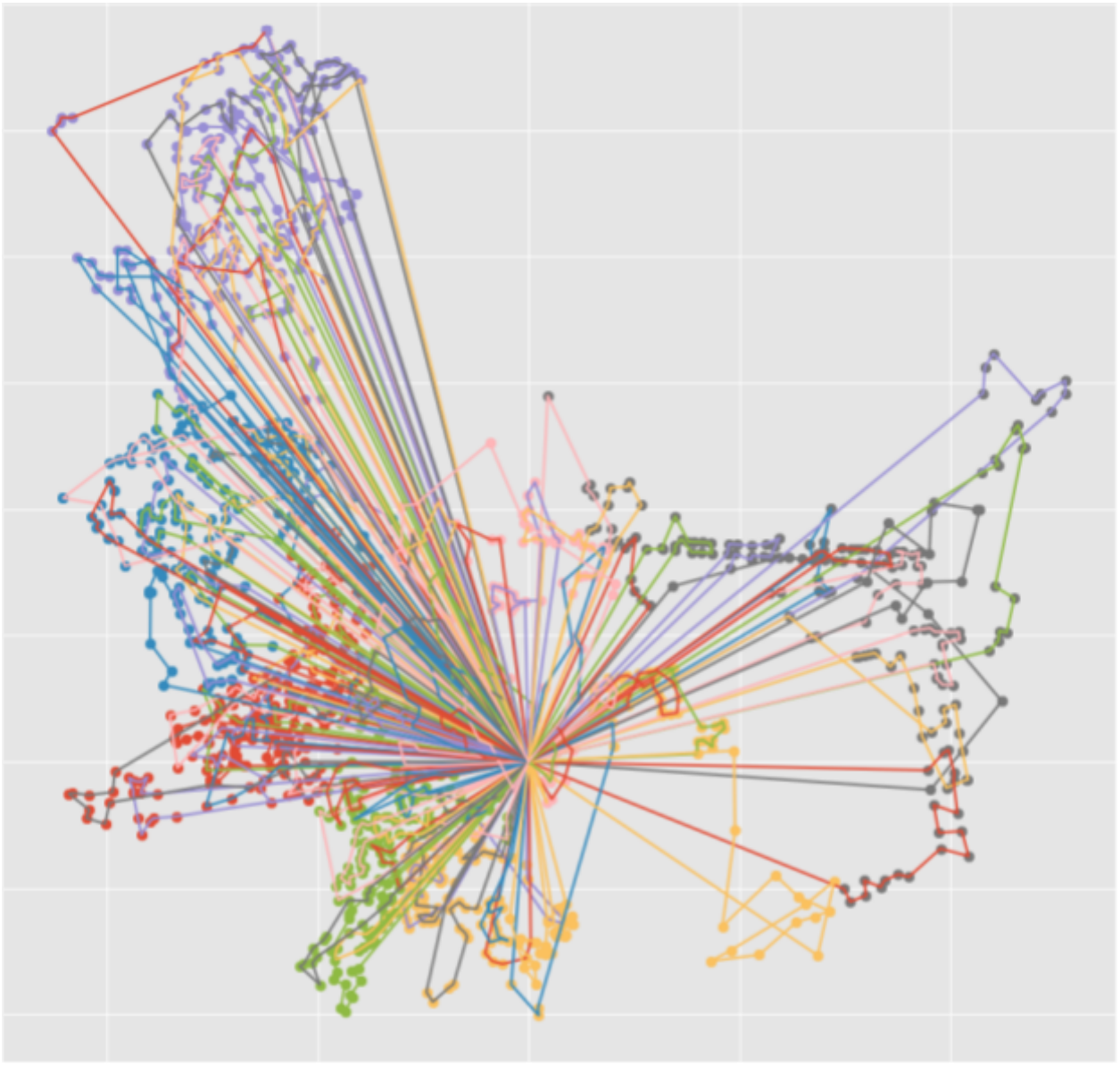
Итоговый результат на базе жадного алгоритма

Те же параметры;

Длина, у.е.: 14482.9;

Время, сек: 421.601;

Количество маршрутов: 156.



Итоговый результат на базе алгоритма на муравьиных колониях

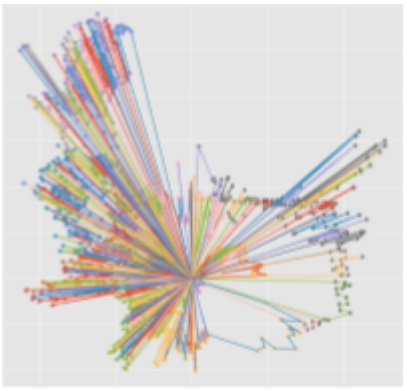
Те же параметры;

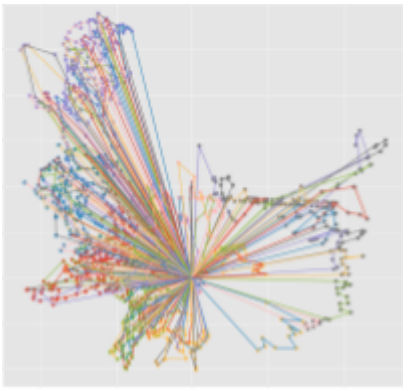
Длина, у.е.: 13774.2;

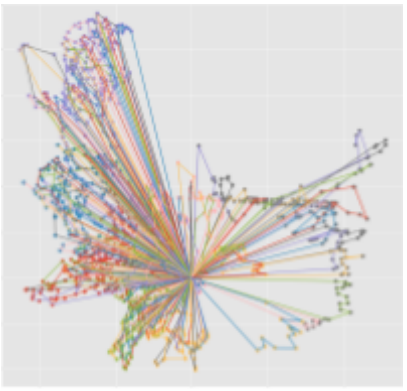
Время, сек: 702.693;

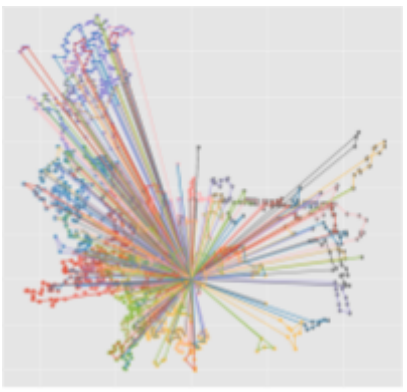
Количество маршрутов: 139.

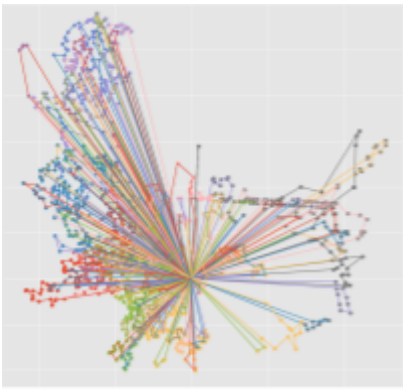
Сравнение улучшения для различных первичных решений

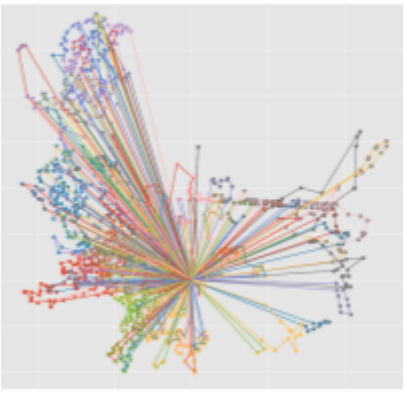


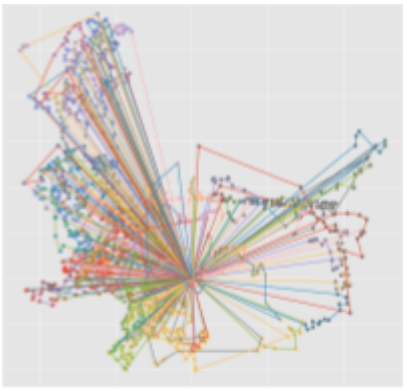


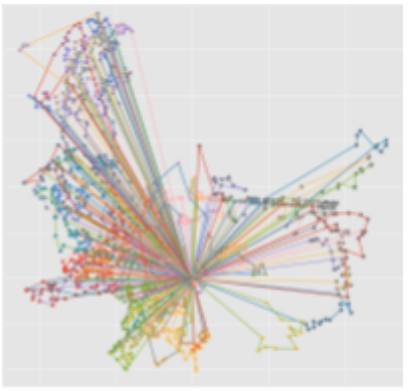


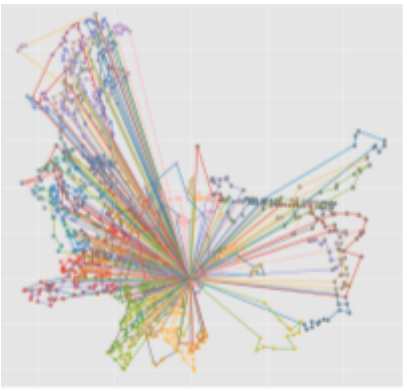


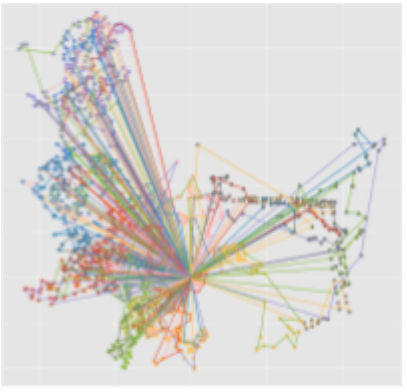


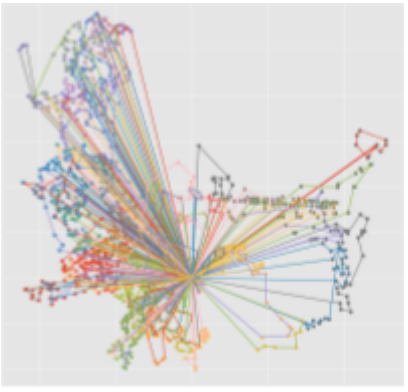


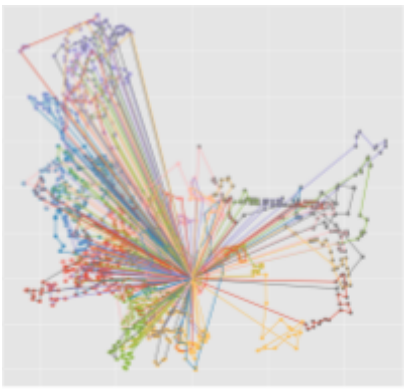












Анализ результатов

Длина, у. е.

Этап решения	Алгоритм			
	заметания	Кларк-Райт	жадный	муравьиный
первичное	20492.6	17021.5	15476.2	14493.2
Осман	15633.1	16472.7	14592.2	13863.5
поиск в окрестностях	15633.1	16357.4	14482.9	13774.2

Время, сек

Этап решения	Алгоритм			
	заметания	Кларк-Райт	жадный	муравьиный
первичное	0.615	9.898	1.246	282.251
Осман	281.885	291.71	281.6	562.581
поиск в окрестностях	421.838	431.155	421.601	702.693

Количество маршрутов

Этап решения	Алгоритм			
	заметания	Кларк-Райт	жадный	муравьиный
первичное	164	184	156	134
Осман	164	184	156	136
поиск в окрестностях	164	184	156	139

Выбор способа построения первичного базового решения оказывает существенное влияние на возможности дальнейшего улучшения. Так чем более совершенный алгоритм применялся на первом этапе, тем лучше было

качество решения на последующих, за исключением пары алгоритмов заметания и Кларка-Райта. А также решение алгоритма Османа на базе заметания не смогла быть улучшено последующим методом.

Поэтому можно видеть, что на каждом этапе работы решение, построенное за счет муравьиного алгоритма оказывается наилучшим.

Стоит упомянуть, что алгоритмы Османа, локальной оптимизации в чередующихся окрестностях и муравьиный работают в течении указанного времени, без иного критерия остановки. Из чего следует, что реальное эффективное время работы может быть меньше на 0-25%.

Также реализованные улучшающие алгоритмы не способны добавлять новые маршруты, а только изменять имеющиеся или удалять их. Однако из таблицы видно, что у муравьиного алгоритма выросло количество маршрутов. Это связано с тем, что сам муравьиный алгоритм в процессе своей работы создает множество пустых маршрутов, в которые потом и добавляют вершины улучшающие.

Из чего следует, что возможно улучшение других алгоритмов, строящих первичные решения путем добавления в них пустых маршрутов.

Ссылка на гитхаб с кодом проекта:
https://github.com/EvgenyOleshkevich/transopt_routing

Заключение

Задача маршрутизации транспорта — NP-трудная задача, имеющая множество различных формулировок и постановок: с множеством депо, с требованиями на длину, вместимость или сбалансированность маршрутов, с раздельной или ограниченной по времени доставкой и другие.

Основной подход к решению ЗМТ эвристические и метаэвристические алгоритмы. Для каждой конкретной формулировки ЗМТ подбираются определенные алгоритмы, которые не просто способны найти допустимое решение, но будут достаточно эффективны.

В процессе работы была сформулирована постановка задачи, отвечающая требованиям предметной области, а именно построение маршрутов для сбора и вывоза ТБО, а также разработана программа, реализующая и совмещающая известные алгоритмы. За счет общего интерфейса алгоритмов, данная программа достаточно гибко и легко поддается модификации: добавлению новых алгоритмов, изменению старых, а также внесению изменений в логику программы.

Результат работы был протестирован на данных о расположении площадок ТБО в городе Краснодар и проведен сравнительный анализ решений, построенных разными алгоритмами.

Список использованных источников

1. Пожидаев М. С. Алгоритмы решения задачи маршрутизации транспорта: дис. … канд. техн. наук / М.С. Пожидаев; Том. гос. ун-т. – Томск, 2010. – 134 с.
2. Манукян Г. Некоторые конструктивные классические алгоритмы для решения задачи маршрутизации транспорта // Сб. науч. трудов НАН РА Междунар. науч.-образов. центр. 2018. С. 33–41.
3. Григорьев В. П., Киселев К. А. Маршрутизация доставки розничной продукции в городской дорожной сети на основе генетического алгоритма // Известия ТПУ. 2007. №2. URL: https://cyberleninka.ru/article/n/marshrutizatsiya-dostavki-roznichnoy-produktsii-v-gorodskoy-dorozhnoy-seti-na-osnove-geneticheskogo-algoritma (дата обращения: 01.03.2021).
4. Просов С.Н., Кузьменко Е.А. “Декомпозиция задачи маршрутизации по эвристикам метода Кларка–Райта”, 2019;
5. Васильев Ю. М. Компьютерные системы принятия решений в задачах маршрутизации транспорта на графах // Известия СПбГЭУ. 2018. №1 (109). URL: https://cyberleninka.ru/article/n/ kompyuternye-sistemy-prinyatiya-resheniy-v-zadachah-marshrutizatsii-transporta-na-grafah (дата обращения: 01.03.2021).
6. Широких В. А. Динамическая адаптация эвристических алгоритмов в задачах маршрутизации транспорта: вып. квалиф. работа / В. А. Широких; СПбГУ – СПб, 2018. – 62 с. URL: http://hdl.handle.net/11701/11963
7. Места (площадки) накопления твёрдых коммунальных отходов [Электронный ресурс] // Сайт Администрация и городская Дума Краснодара. URL: https://krd.ru/podrazdeleniya/administratsii-krasnodara/departament-gorodskogo-khozyaystva-i-toplivno-energeticheskogo-kompleksa/tbo/ (дата обращения: 27.04.2021).

8. OpenStreetMap [Электронный ресурс] // URL: https://www.openstreetmap.org/ (дата обращения: 15.02.2021).
9. BBBike.org: [Электронный ресурс] // URL: https://extract.bbbike.org/ 27.04.2021).
10. Сеидова А. С. Использование пакета WIZWHY для формирования базы знаний экспертных систем / А. С. Сеидова // МНСК-2017: Информационные технологии : Материалы 55-й Международной научной студенческой конференции, Новосибирск, 17–20 апреля 2017 года. – Новосибирск: Новосибирский национальный исследовательский государственный университет, 2017. – С. 124.
11. Старикова А. В. Создание подсистемы принятия решений в медицинских информационных системах./А.В.Старикова, О.Г.Берестнева, Г.Е.Шевелев, К.А. Шаропин, Л.И. Кабанова// Известия Томского политехнического университета. 2010. Т.317. №5. С.194–197.
12. К. В. Воронцов, Лекции по логическим алгоритмам классификации, 2007, адрес статьи: http://www.ccas.ru/voron/download/LogicAlgs.pdf, С.33–35;
13. Fisher M. L., Jaikumar R. A generalized assignment heuristic for vehicle routing // Networks. 1981. Т. 11. № 2. С. 109–124. DOI: https://doi.org/10.1002/net.3230110205
14. Bramel J., Simchi-Levi D. A Location Based Heuristic for General Routing Problems // Oper. Res. 1995. Т. 43. № 4. С. 649–660. DOI: http://doi.org/10.1287/opre.43.4.649
15. Renaud J., Boctor F. F., Laporte G. An Improved Petal Heuristic for the Vehicle Routeing Problem // J. Oper. Res. Soc. 1996. Т. 47. № 2. С. 329–336. DOI: https://doi.org/10.1057/jors.1996.29
16. Захаров, В. В. Динамическая адаптация генетического алгоритма маршрутизации транспорта на больших сетях / В. В. Захаров, А. В. Мугайских // Управление большими системами: сборник трудов. – 2018. – № 73. – С. 108-133. DOI: https://doi.org/10.25728/ubs.2018.73.6

17. Мугайских А. В. Задачи маршрутизации транспорта на сети мегаполиса: магистерская диссертация. СПбГУ – СПб., 2017. – 59 с. URL: http://hdl.handle.net/11701/11629
18. Glover F., Laguna M. Tabu Search. Boston, MA: Springer US, 1997. DOI: http://doi.org/10.1007/978-1-4615-6089-0
19. Gendreau M., Hertz A., Laporte G. A Tabu Search Heuristic for the Vehicle Routing Problem // Manage. Sci. 1994. Т. 40. № 10. С. 1276–1290. DOI: http://doi.org/10.1287/mnsc.40.10.1276
20. Taillard É. Parallel iterative search methods for vehicle routing problems // Networks. 1993. Т. 23. № 8. С. 661–673. DOI: https://doi.org/10.1002/net.3230230804
21. Xu J., Kelly J. P. A Network Flow-Based Tabu Search Heuristic for the Vehicle Routing Problem // Transp. Sci. 1996. Т. 30. № 4. С. 379–393. DOI: https://doi.org/10.1287/trsc.30.4.379
22. Кубил В. Н. Исследование и разработка методов решения многокритериальных задач маршрутизации транспорта на основе муравьиного алгоритма: дис. … канд. техн. наук / В.Н. Кубил; Южно-Российский гос. политех. ун-т (НПИ) им. М. И. Платова. – Новочеркасск, 2019. – 184 с.
23. Robust F., Daganzo C. F., Souleyrette R. R. Implementing vehicle routing models // Transp. Res. Part B Methodol. 1990. Т. 24. № 4. С. 263–286. DOI: https://doi.org/10.1016/0191-2615(90)90002-G
24. Renaud J., Boctor F. F., Laporte G. An Improved Petal Heuristic for the Vehicle Routeing Problem // J. Oper. Res. Soc. 1996. Т. 47. № 2. С. 329–336. DOI: https://doi.org/10.1287/trsc.29.2.143
25. Solomon M. M. Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints // Oper. Res. 1987. Т. 35. № 2. С. 254–265. DOI: https://doi.org/10.1287/opre.35.2.254

26. Parragh S. N., Doerner K. F., Hartl R. F. A survey on pickup and delivery problems // J. für Betriebswirtschaft. 2008. Т. 58. № 2. С. 81–117. DOI: https://doi.org/10.1007/s11301-008-0036-4
27. Хмелев А. В. Алгоритмы локального поиска для задач маршрутизации транспортных средств: дис. … канд. физ.-мат. наук / А. В. Хмелев; НГУ – Новосибирск, 2015. – 119 с.
28. Библиотека Boost [Электронный ресурс] // URL: https://www.boost.org/ (дата обращения: 15.02.2021).
29. Лукинский В. С., Модели и методы теории логистики, 2 издание, Лукинский В.С., Лукинский В.В., Малевич Ю.В., Пластуняк И.А., Плетнева Н.Г., СПБГЭУ, 2007;
30. Гончарова Ю. А. Оптимизация доставки однородного груза различным клиентам на базе алгоритма муравьиной колонии, основанного на популяции: дис. … канд. техн. наук / Ю. А. Гончарова; Уфимский гос. авиац. техн. ун-т. – Уфа, 2017. – 152 с.
31. Dantzig G. B., Ramser J. H. The Truck Dispatching Problem // Manage. Sci. 1959. Т. 6. № 1. С. 80–91. DOI: https://doi.org/10.1287/mnsc.6.1.80
32. Rego C., Roucairol C. A Parallel Tabu Search Algorithm Using Ejection Chains for the Vehicle Routing Problem // Meta-Heuristics. Boston, MA: Springer US, 1996. С. 661–675. DOI: https://doi.org/10.1007/_978-1-4613-1361-8_40