Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Новосибирский государственный технический университет»

Кафедра теоретической и прикладной информатики

**Отчет ПО ПРАКТИКЕ**

Учебная практика: ознакомительная практика

(наименование практики в соответствии с учебным планом)

Направление подготовки: 02.03.03 Математическое обеспечение и администрирование информационных систем

|  |  |
| --- | --- |
| Выполнил:  Студент Ковалевский Д.Е.  (Ф.И.О.)  Группа ПМИ-11  Факультет ПМИ.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись  «15» декабря 2023 г. | Проверил:  Руководитель от НГТУ Зайцев М.Г  (Ф.И.О.)  Балл: \_\_\_\_\_\_\_\_\_\_, ECTS\_\_\_\_\_\_\_\_\_\_\_\_\_,  Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «отлично», «хорошо», «удовлетворительно», «неуд.»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись  «27» декабря 2023 г. |
|  |  |

Новосибирск 2023

Оглавление

[Введение 3](#_Toc153307632)

[1.1. Задачи транспортной маршрутизации с одним депо 4](#_Toc153307633)

[1.2. Задачи транспортной маршрутизации с несколькими депо 5](#_Toc153307634)

[1.3. Задачи транспортной маршрутизации с временными окнами 6](#_Toc153307635)

[1.4. Задачи транспортной маршрутизации с раздельной доставкой 6](#_Toc153307636)

[1.5. Задачи транспортной маршрутизации с неоднородным автопарком 6](#_Toc153307637)

[Глава 2. Методы решения задачи транспортной маршрутизации с ограниченной грузоподъёмностью 7](#_Toc153307638)

[2.1. Метод K-means 8](#_Toc153307639)

[2.2. Метод K-means++ 9](#_Toc153307640)

[2.3. Метод поиска с запретами 11](#_Toc153307641)

[Глава 3. Решение задачи маршрутизации с ограниченной грузоподъёмностью и одним депо 13](#_Toc153307642)

[3.1. Постановка задачи 13](#_Toc153307643)

[3.2. Математическая модель задачи 13](#_Toc153307644)

[3.3. Эксперимент 15](#_Toc153307645)

[3.4. Сравнение методов k-means и k-means++ 15](#_Toc153307646)

[Заключение 31](#_Toc153307647)

[Литература 31](#_Toc153307648)

[Приложение 34](#_Toc153307649)

[Приложение 1. Функция метода k-средних 34](#_Toc153307650)

[Приложение 2. Функция метода k-средних++ 35](#_Toc153307651)

[Приложение 3. Функция метода поиска с запретами 35](#_Toc153307652)

# Введение

Данная работа посвящена математическому исследования задач оптимизации вопросов логистики. Под логистикой обычно понимают область деятельности предприятия, связанную с планированием, организацией, управлением и контролем движения материальных и информационных потоков в пространстве и во времени от их первичного источника до конечного потребителя, а также науку, изучающую эти процессы.

Основной целью в транспортной логистике является построение эффективных с точки зрения стоимости маршрутов объезда транспортными средствами пунктов-продавцов и пунктов-покупателей. В понятие стоимости в данном случае включаются любые затраты, возникающие в процессе объезда клиентов транспортным средством (ТС), которые могут определяться через длину маршрута, время в пути или количество потребляемого топлива. Правильная организация транспортировки может уменьшить экономические затраты на перевозку. Поэтому этот вопрос является одним из наиболее важных для многих предприятий.

Все задачи принадлежат классу NP вычислительной сложности задач, поэтому в тестовых примерах большой размерности задача становится трудно вычисляемой, т.е. не может быть решена методами с полным перебором вариантов решений т.к. простейшая задача с n точек будет иметь n! решений. Как правило, используют эвристические и метаэвристические алгоритмы. Они генерируют приближенные к оптимальному решению, но требуют меньшего времени по сравнению с точными методами.

В данной работе рассмотрен один из видов обобщения задачи коммивояжера – задача маршрутизации с вывозом и доставкой несколькими транспортными средствами учитывая грузоподъёмность транспорта. В главе 1 представлена классификация задач данного типа, а в главе 2 – методы решения. Глава 3 посвящена формулировке задачи транспортной маршрутизации с ограниченной грузоподъёмностью, а затем представлены решения задачи и анализ полученных решений.

Целью работы является решение задачи маршрутизации вывоза и доставки товара с несколькими транспортными средствами одинаковой грузоподъемности.

**Глава 1. Обзор задач транспортной маршрутизации с ограниченной грузоподъёмностью (Capacitated vehicle routing problem)**

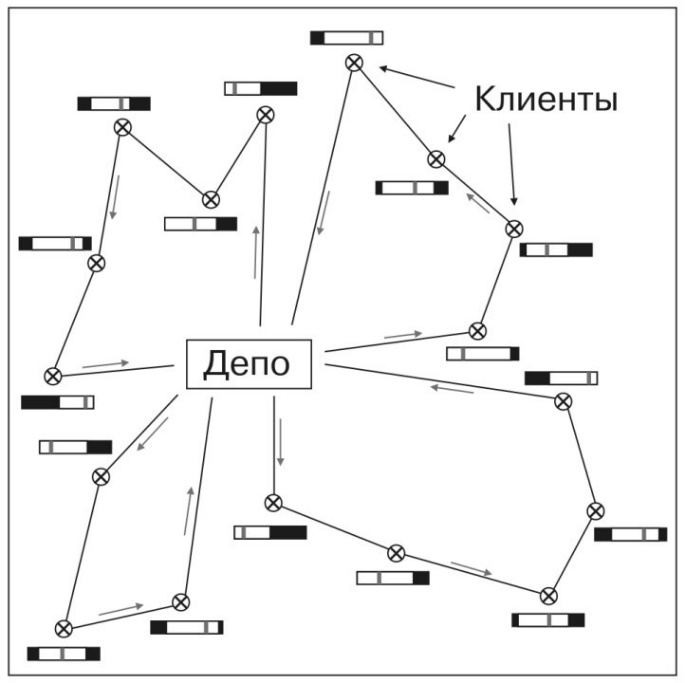
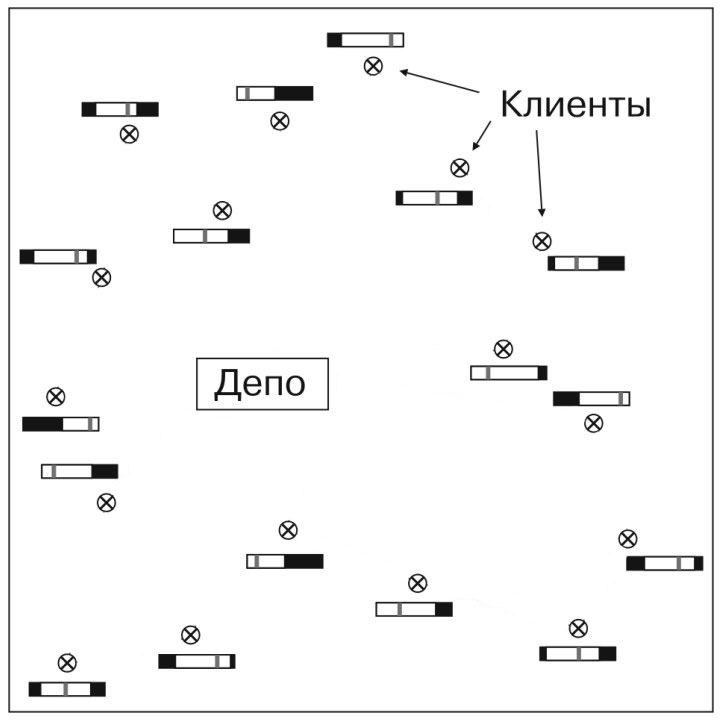
Задача транспортной маршрутизации с ограниченной вместимостью (Vehicle Routing Problems, VRP) – хорошо известная задача комбинаторной оптимизации в логистике и транспортировке. В CVRP существует набор клиентов с известными требованиями и парк транспортных средств ограниченной вместимости, которые используются для обслуживания этих клиентов. Цель состоит в том, чтобы найти оптимальный набор маршрутов для транспортных средств, чтобы минимизировать общую стоимость или пройденное расстояние, гарантируя при этом, что общая потребность каждого транспортного средства не превысит его вместимость.

Проблему можно представить в виде графика, где узлы представляют клиентов и склад (начальную и конечную точки), а ребра представляют маршруты между ними. У каждого клиента есть спрос на товары или услуги, которые необходимо доставить, и каждое транспортное средство имеет ограниченную вместимость для перевозки этих товаров.

Capacitated Vehicle Routing Problem (CVRP) можно разделить на несколько типов в зависимости от различных характеристик и вариаций. На CVRP с одним депо, CVRP с несколькими депо, CVRP с временными окнами, CVRP с раздельной доставкой и CVRP с гетерогенным автопарком.

## 1.1. Задачи транспортной маршрутизации с одним депо

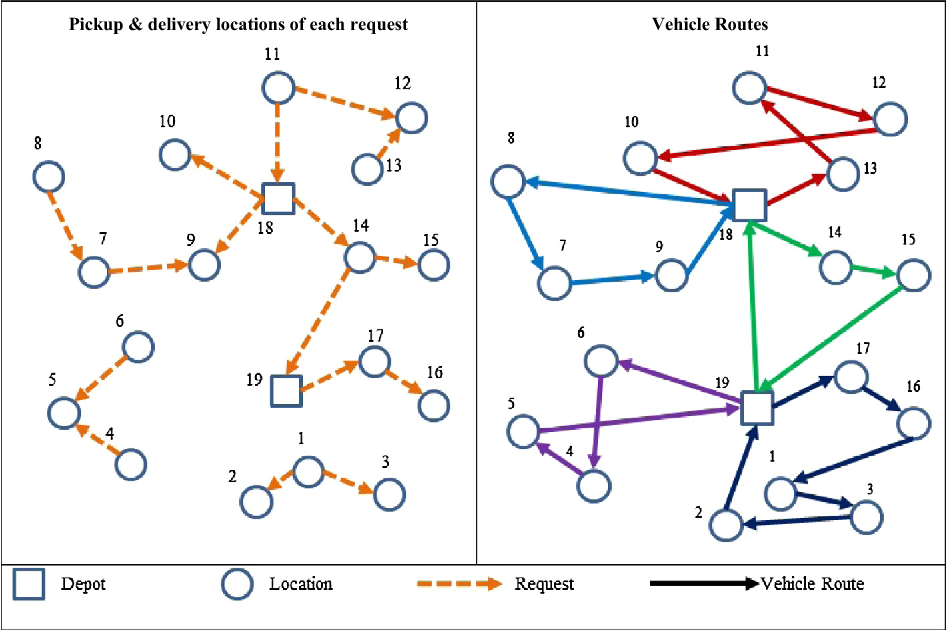
Этот тип CVRP включает в себя одно депо, с которого все транспортные средства отправляются и возвращаются. Цель состоит в том, чтобы найти оптимальные маршруты для транспортных средств, чтобы обслуживать группу клиентов, соблюдая при этом их требования и вместимость транспортного средства.



**Рисунок 1.1.** пример задачи транспортной маршрутизации с одним депо [3].

## 1.2. Задачи транспортной маршрутизации с несколькими депо

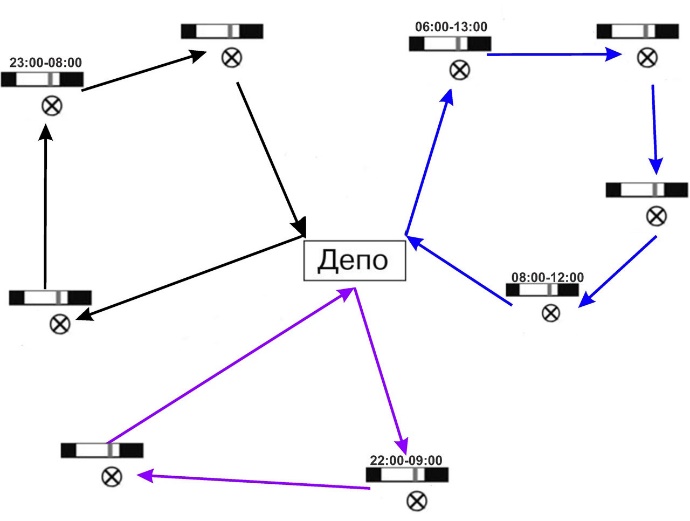
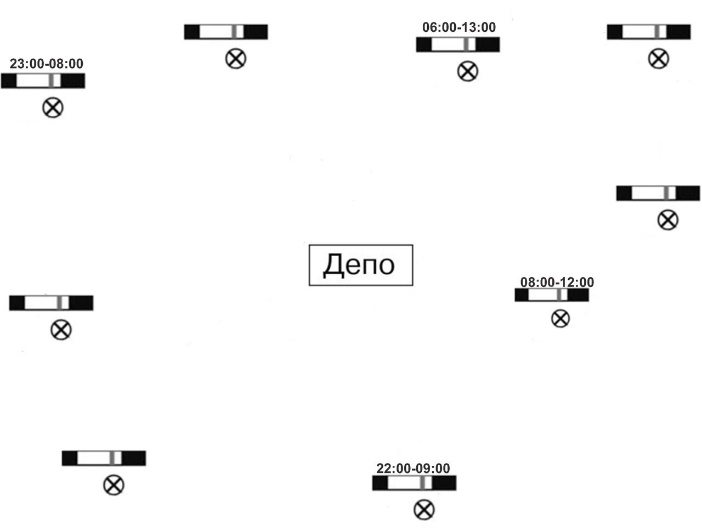
В этом варианте существует несколько депо, с которых транспортные средства отправляются и возвращаются. Каждое депо имеет свой собственный набор клиентов для обслуживания. Цель состоит в том, чтобы найти оптимальные маршруты для транспортных средств с разных складов, учитывая их соответствующие мощности и требования клиентов.



**Рисунок 1.2.** пример задачи транспортной маршрутизации с двумя депо [36].

## 1.3. Задачи транспортной маршрутизации с временными окнами

CVRP временного окна добавляет дополнительное ограничение временных окон для посещений клиентов. У каждого клиента есть определенное время, в течение которого он может быть обслужен. Цель состоит в том, чтобы найти маршруты, которые удовлетворяют ограничениям как по пропускной способности, так и по временному интервалу.



**Рисунок 1.3.** пример задачи транспортной маршрутизации с временными окнами.

## 1.4. Задачи транспортной маршрутизации с раздельной доставкой

В SDVRP, каждый заказ может быть разделен на части и доставлен различными транспортными средствами. Это позволяет более гибко планировать и оптимизировать маршруты доставки, учитывая ограничения по емкости и другие факторы. Основная цель SDVRP состоит в определении оптимального набора маршрутов для каждого транспортного средства, учитывая ограничения по емкости, времени и другие условия доставки.

## 1.5. Задачи транспортной маршрутизации с неоднородным автопарком

В HVRP, каждое транспортное средство имеет свою уникальную пропускную способность, скорость или другие характеристики, которые могут отличаться от других транспортных средств в автопарке. Задача состоит в оптимальном распределении заданий между транспортными средствами с учетом этих различий в характеристиках.

Решение HVRP обычно включает в себя определение оптимального набора маршрутов для каждого транспортного средства, учитывая их пропускные способности и другие ограничения, такие как ограничения на время и емкость.

# Глава 2. Методы решения задачи транспортной маршрутизации с ограниченной грузоподъёмностью

Для решения задачи транспортной маршрутизации с ограниченной грузоподъемностью (Vehicle Routing Problem with Capacity Constraints, VRPCC) существуют различные методы и алгоритмы, которые делятся на 3 группы: точные, метаэвристические и приближенного решения.

При небольших размерах задач используют точные методы:

* Метод ветвей и границ [22, 25];
* Метод ветвей с отсечением [26];
* Динамическое программирование [12].

Такие подходы перебирают решения, пока не будет найдено оптимальное. В худшем случае будут перебираться все варианты возможных решений. Время работы точных алгоритмов на практике велико, поэтому чаще применяются эвристические и метаэвристические. В этом случае ищутся приближенные решения, которые находятся достаточно быстро и достаточно точны для требуемых целей. В эвристических методах производится ограниченный поиск по пространству решений (т.е. нет перебора всевозможных вариантов).

В метаэвристических методах изучаются наиболее перспективные части пространства решений. Качество решений получается выше, чем у полученных классическими эвристиками. К этой группе относятся:

* Генетический алгоритм [23];
* Муравьиный алгоритм [10];
* Последовательный локальный поиск.

Эти методы могут быть комбинированы и настраиваться для конкретных требований и ограничений задачи транспортной маршрутизации с ограниченной грузоподъемностью. Каждый метод имеет свои преимущества и недостатки, и выбор оптимального метода зависит от конкретной ситуации и доступных ресурсов.

Методы приближенного решения представляют собой подходы к решению задач, когда точное решение недоступно или слишком сложно вычислить. Вместо этого, методы приближенного решения стремятся найти решение, которое близко к оптимальному, но не обязательно идеальное.

К этой группе относятся:

* Построение эвристического решения;
* Методы разбиения или кластеризации.

## 2.1. Метод K-means

Метод k-средних — наиболее популярный метод [кластеризации](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%82%D0%B5%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F). Под кластеризацией понимается обнаружение естественных группировок в данных. Обычно, когда нам даны данные, которые мы можем визуализировать (2- или 3-мерные данные), человеческий глаз может легко формировать различные кластеры. Но для машин это делать несколько сложнее. Но тут на помощь и приходят алгоритмы кластеризации, способные находить кластеры в пространстве с ещё большим количеством измерений, чего не может делать даже глаз человека.

Метод был изобретён в 1950-х годах математиком Гуго Штейнгаузом [29] и почти одновременно Стюартом Ллойдом. Особую популярность приобрёл после работы Маккуина.

Действие алгоритма таково, что он стремится минимизировать суммарное квадратичное отклонение точек кластеров от центров этих кластеров:

где �  — число кластеров, �� — полученные кластеры, �=1,2,…,�, а �� — центры масс всех векторов  � из кластера ��.

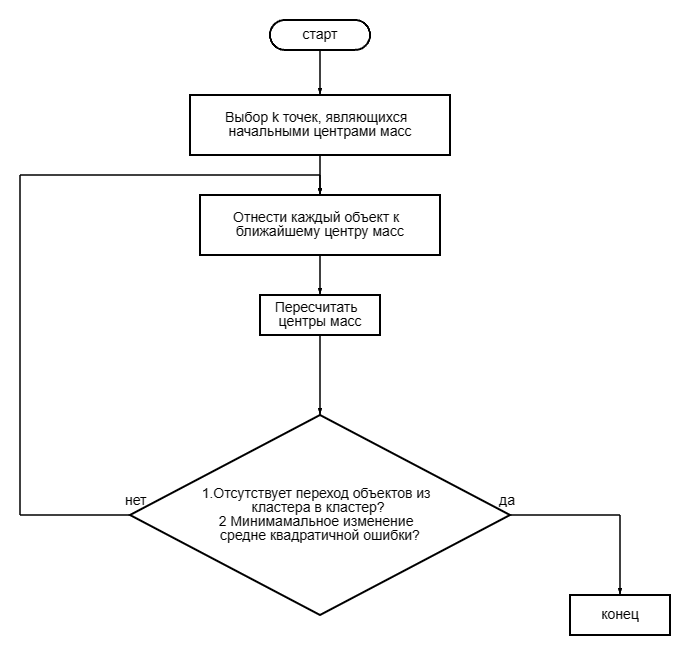
Основной задачей алгоритма является разделение m объектов из пространства на k кластеров, при этом каждый объект относится к тому кластеру, к центру которого он ближе всего. Центры кластеров называются центроидами. В качестве меры близости используется Евклидово расстояние:

Оценить качество каждого кластера можно методом WCSS (within-cluster sum of squares). Основная идея, лежащая в основе k-means, заключается в определении таких кластеров, что отклонения внутри каждого кластера минимальны. Чтобы оценить отклонение, мы вычислим такую величину, как WCSS (within-cluster sum of squares) – cумму квадратов внутрикластерных расстояний до центра кластера.

Где С – это кластерные центроиды, а d – значения данных в каждом кластере

Метод не предусматривает автоматического определения оптимального количества кластеров.

На рисунке 2.1 представлена блок-схема метода.



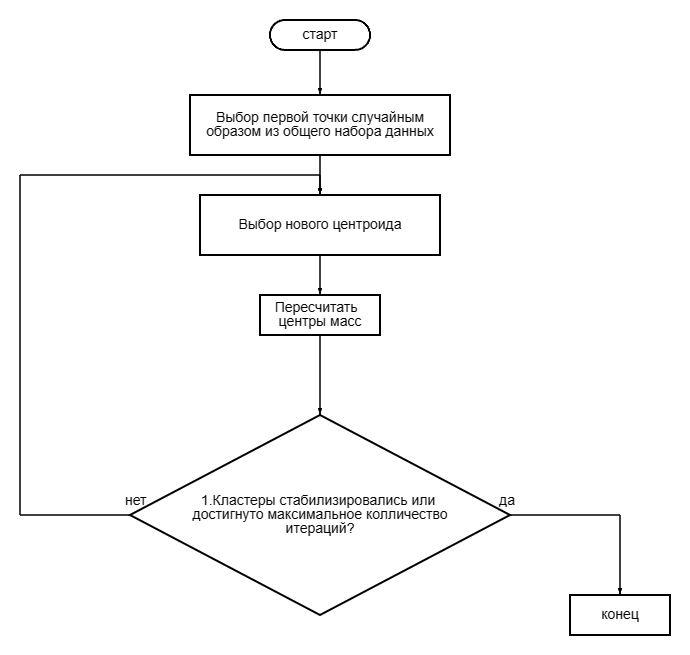
**Рисунок 2.1**. Блок-схема метода k-means.

## 2.2. Метод K-means++

k-means++ — улучшенная версия алгоритма кластеризации k-means. Суть улучшения заключается в нахождении более «хороших» начальных значений центроидов кластеров. Оригинальный k-means не регламентирует то, как выполняется этот этап алгоритма, и поэтому является нестабильным. Алгоритм предложен в 2007 году Дэвидом Артуром и Сергеем Вассильвитским. Также есть другие похожие методы, открытые другими учёными независимо [39].

Чтобы преодолеть вышеупомянутый недостаток, мы используем K-means ++. Этот алгоритм обеспечивает более разумную инициализацию центроидов и улучшает качество кластеризации. Помимо инициализации, остальная часть алгоритма такая же, как у стандартного алгоритма K-means. То есть K-means ++ - это стандартный алгоритм K-means в сочетании с более продуманной инициализацией центроидов.

Первым этапом инициализации является выбор первого центроида из точек данных. Далее для каждой точки данных вычисляется расстояние от ближайшего, ранее выбранного центроида. Следующий центроид выбирается таким образом, чтобы вероятность выбора точки в качестве центроида была прямо пропорциональна ее расстоянию от ближайшего, ранее выбранного центроида. (т.е. точка, имеющая максимальное расстояние от ближайшего центроида, с наибольшей вероятностью будет выбрана следующей в качестве центроида).

На рисунке 2.2 представлена блок-схема метода.

**Рисунок 2.2**. Блок-схема метода k-means++.

## 2.3. Метод поиска с запретами

Метод поиска с запретами (Tabu Search) является одним из наиболее эффективных метаэвристических методов. Он был предложен Ф. Гловером в 80-е годы [30]. Отличительной чертой этого метода является процесс введения и снятия некоторых искусственных ограничений задачи в процессе поиска решения.

Метод поиска с запретами и его вариации нашли широкое применение в решении разных оптимизационных задач NP-сложности: задач о составлении расписания, задач об упаковке, задач о выборе оптимального маршрута, задач о размещении и ряда других оптимизационных задач [37].

Метод поиска с запретами дает результаты, близкие к оптимальным, за приемлемое время, что позволяет использовать его в оптимизационных задачах наряду с другими метаэвристическими методами.

Основным недостатком метода локального поиска является его остановка при достижении локального оптимума. Под локально оптимальными понимаются такие решения, окрестность которых не содержит лучших по отношению к целевой функции решений, т.е. , где  – локальный оптимум. Нашим же искомым решением является глобальный оптимум. Очевидно, что глобальный оптимум является также и локальным, поэтому для успешного поиска решений мы должны как-то переходить от одного локального оптимума к другому.

В методе поиска с запретами с целью преодолеть вышеописанный недостаток вводится т.н. список запретов (Tabu List). Этот список хранит некоторое количество предыдущих решений, и при выборе нового решения запрещается выбирать из окрестности решения, содержащиеся в списке запретов.

Метод состоит в последовательном улучшении некоторого допустимого решения с целью максимизировать (минимизировать) целевую функцию. Основным инструментом для этого является список запретов Tabu(), строящийся на предыстории поиска (по нескольким предыдущим решениям) и запрещающий часть окрестности текущего решения N(𝑖𝑘). Тогда на каждом шаге ищется оптимальное решение для подзадачи:

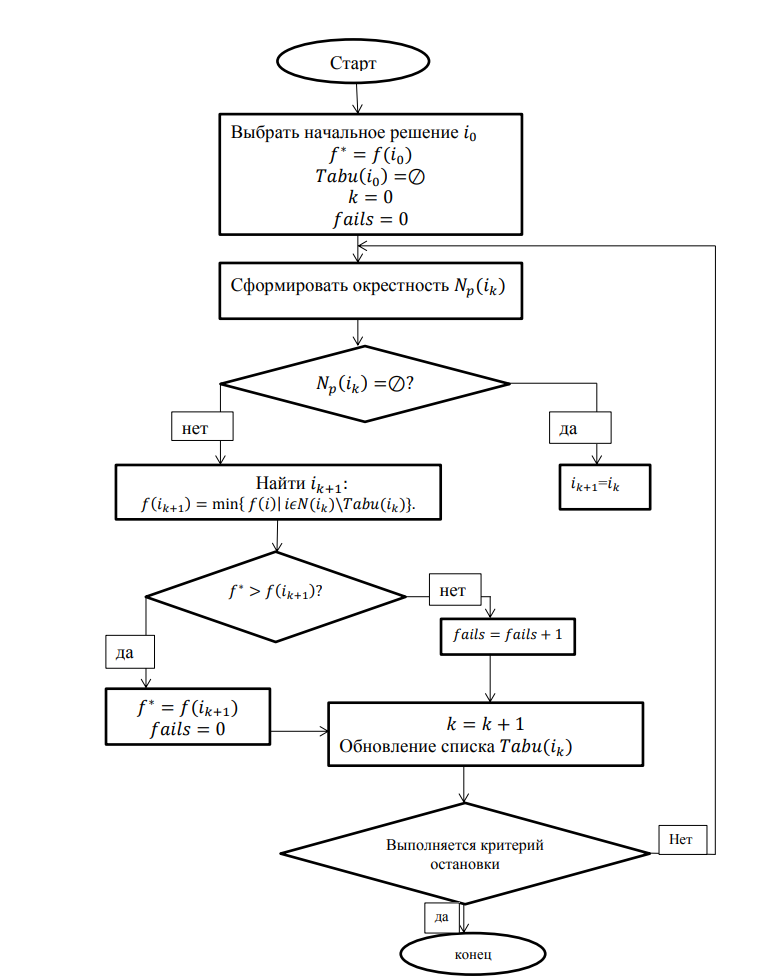
Список запретов запрещает использовать фрагмент решения (ребро графа, координат вектора), которые менялись на последних 𝑠 шагах. Константа 𝑠 является параметром метода, задающая длину списка Tabu. При 𝑠 = 1 получается стандартный локальный поиск. Вторым параметром метода выступает мощность окрестности N(𝑖𝑘) – 𝑝. При малом значении этого параметра метод не сможет перемещаться из одного локального оптимума в другой.

На рисунке 2.3 представлена блок схема метода:

f – значение целевой функции на некотором решении;

k – количество проделанных шагов;

fails – количество раз, когда алгоритм доходит до локального минимума без улучшения подряд.



**Рисунок 2.3**. Блок-схема метода поиска с запретами.

# Глава 3. Решение задачи маршрутизации с ограниченной грузоподъёмностью и одним депо

## 3.1. Постановка задачи

Для подробного рассмотрения выбрана статическая задача доставки товара с одним депо и несколькими ТС одинаковой грузоподъемности.

Имеется конечное множество потребителей. Каждому поставщику соответствует свой потребитель, поэтому количество поставщиков и потребителей одинаковое и равно n. Для решения задачи используется парк из k транспортных средств (ТС) одинаковой грузоподъемности, которую нельзя превышать. Все ТС начинают и заканчивают свой путь в депо (одно для всех ТС).

Перед выездом из депо имеется информация обо всех запросах на перевозку, координаты всех поставщиков и потребителей на плоскости.

Целью задачи является построение маршрутов для каждого ТС, при которых будут обслужены все поставщики и потребители. При этом минимизируются общие затраты на обслуживание всех клиентов. Затраты в данной задаче будут определяться расстоянием между вершинами.

## 3.2. Математическая модель задачи

Задача задается как множество точек, которые нужно посетить, задано как locations, где каждая точка представлена координатами (x, y). Информация о депо хранится в depot, где точка представлена координатами (x, y). Грузоподъемность каждой машины равна Q. Вес груза для каждой точки задан в виде списка Q\_delivery, где Q\_delivery[i] - вес груза для i-ой точки. Количество машин равно K.

Выезд всех транспортных средств осуществляется из депо, сюда же они должны вернуться после окончания маршрута. С каждой вершиной 𝑖 ассоциировано некоторое количество груза 𝑞𝑖 . Если 𝑖 ∈ 𝑃, то 𝑞𝑖 ≥ 0, и если 𝑖 ∈ 𝐷, то 𝑞𝑖 ≤ 0. При этом выполняется равенство 𝑞𝑖 = −𝑞𝑖+𝑛. Для вершин соответствующих депо: 𝑞0 = 𝑞2𝑛+1 = 0.

В задаче рассматривается однородный груз. Накладываются ограничения на порядок обхода: каждый пункт посещается только один раз и только одним ТС.

Для решения задачи, используется парк из 𝑘 ∈ 𝐾 ТС одинаковой грузоподъемности Q. На любом этапе маршрута нельзя допустить переполнения ТС и перевозки отрицательного значения груза.

Целью задачи является построение замкнутых маршрутов для каждого ТС, при которых будут обслужены все поставщики и все потребители. При этом минимизируются общие затраты на обслуживание всех клиентов.

Для распределения ТС по маршрутам используется кластеризация k-means++. Алгоритм случайным образом генерируем `K` центроидов с помощью алгоритма k-means++ и вычисляет метки кластеров для каждой точки.

Создание маршрута начинается с установления текущей локации равной точки депо и установлением грузоподъёмности Q. Далее создается список запретных локаций tabu\_list для недавно посещенных и запрещенных локаций. В данном коде используется простая форма метода поиска с запретами, где используется "табу-лист" (tabu list) для хранения последних посещенных локаций. В каждой итерации алгоритма, когда выбирается следующая доступная локация, проверяется, присутствует ли эта локация в табу-листе или уже была посещена ранее. Если это так, то эта локация будет пропущена и алгоритм продолжит поиск следующей доступной локации.

При этом, если размер табу-листа превышает заданное значение (tabu\_size), самая старая запись в табу-листе удаляется, чтобы освободить место для новых записей.

Метод поиска с запретами позволяет избегать повторных посещений локаций и просматривать более разнообразные варианты маршрутов. Он может помочь алгоритму найти лучшее глобальное решение, учитывая различные ограничения и цели задачи.

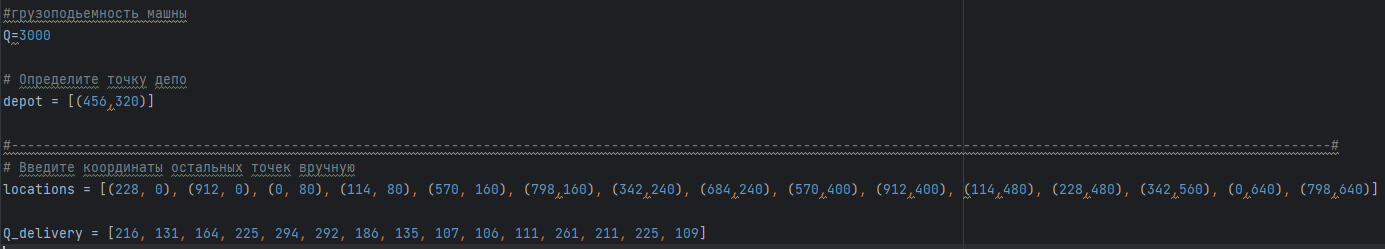
Создание маршрутов:

* Ищем ближайшую доступную локацию с учетом грузоподъемности и добавляем ее в маршрут.
* Обновляем текущую локацию, грузоподъемность и добавляем посещенную локацию в запретные локации.
* Если размер запретных локаций превышает заданный порог, удаляем самую старую локацию из запретных локаций.
* Если маршрут не пустой, добавляем его в список маршрутов.

В случае если грузоподъёмность полностью не использована, создаем новый маршрут от точки депо к оставшимся локациям и добавляем его в список маршрутов. Полученные маршруты представляют оптимальное решение доставки грузов с учетом грузоподъёмности машин и маршрутизации.

## 3.3. Эксперимент

Для проведения эксперимента был рассмотрен тестовый набор точек самостоятельно подготовленный заранее а так же набор точек сгенерированных случайно. Тест состоит из n точек, заданных на координатной плоскости. Для каждого теста проводилась серия экспериментов с разным количеством транспортных средств и разной грузоподъемностью. На рисунке 5.1 подставлен формат тестового набора



**Рисунок 5.1.** Фрагмент файла для тестовой задачи 1P1.DAT из библиотеки NEO.

Решение задачи маршрутизации с несколькими ТС разбивается на два этапа:

1. Разбиение вершин по ТС;

2. Построение маршрутов для каждого ТС.

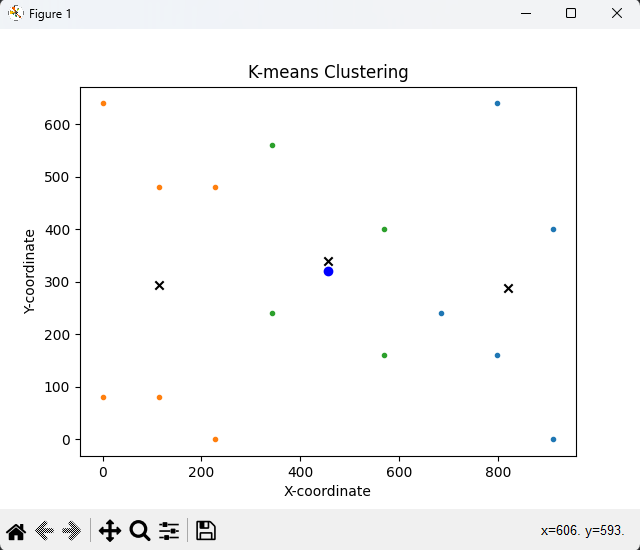
На первом этапе воспользуемся методом кластерного анализа, а именно, методом k-средних (пункт 2.1. главы 2.) и k-средних++ (пункт 2.2 главы 2.)

На втором этапе формируем маршруты метаэвристическими методами: метод писка с запретами (пункт 2.3 главы 2).

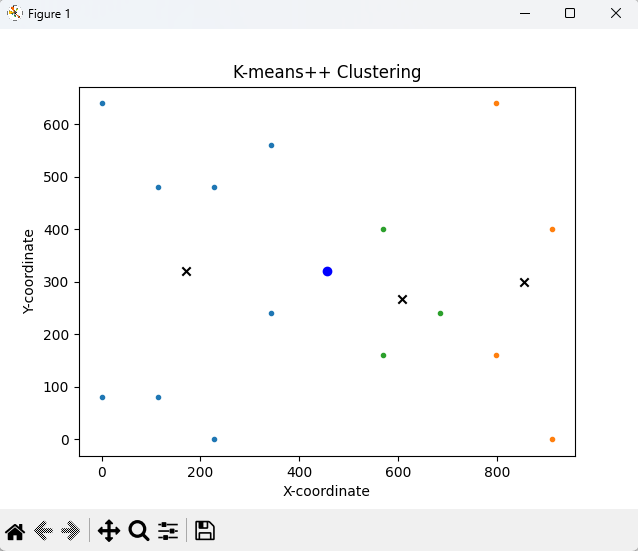
## 3.4. Сравнение методов k-means и k-means++

Методы будут сравниваться по точности и временным затратам из чего будет строиться общая оценка каждого метода кластеризации. Для примера будут использоваться визуальные отображения кластеризации на разном количестве точек и будет прилагаться таблица с данными о времени работы каждого метода.

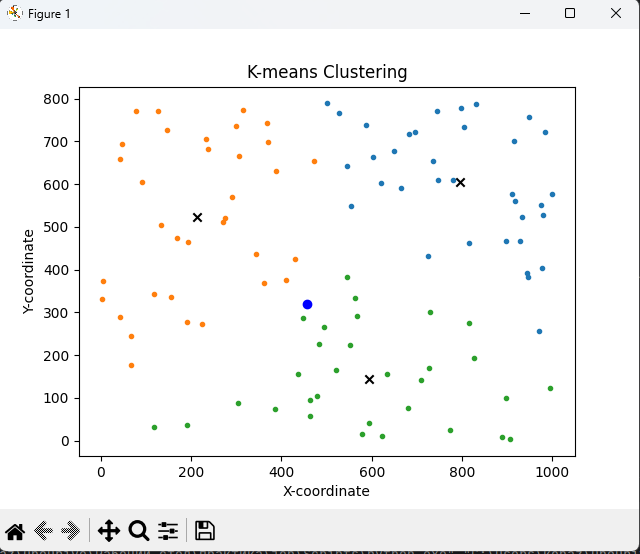
Возьмем количество точек равное n, тогда n=15,100,500,1000,10000,1000000. Количество транспорта K=3.



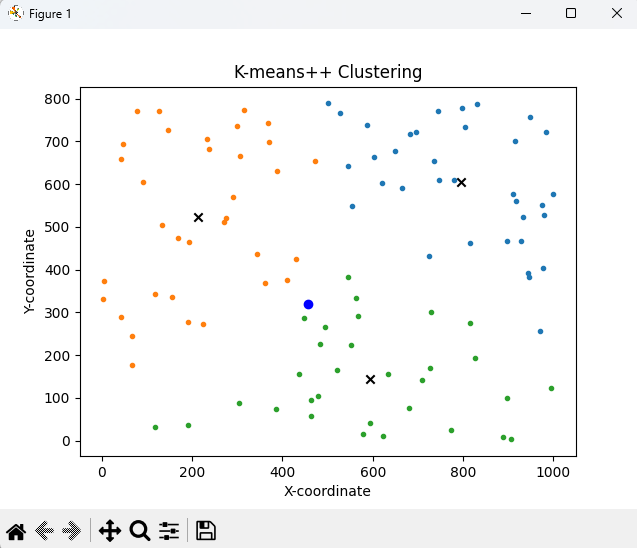
**Рисунок 3.1.** Кластеризация методом k-means для n=15.



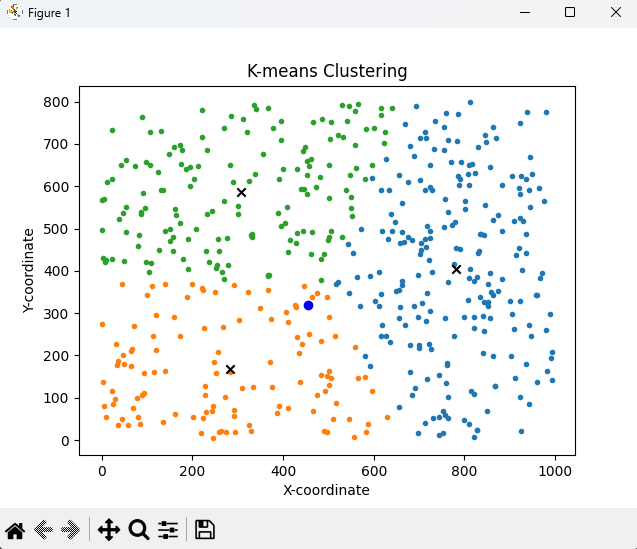
**Рисунок 3.2.** Кластеризация методом k-means++ для n=15.



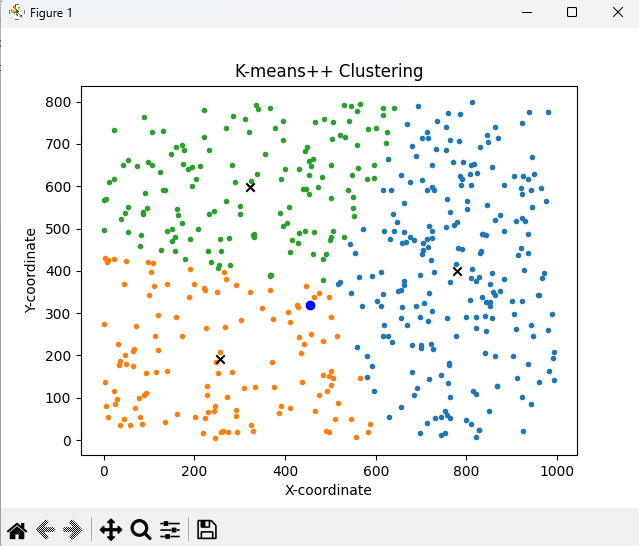
**Рисунок 3.3.** Кластеризация методом k-means для n=100.



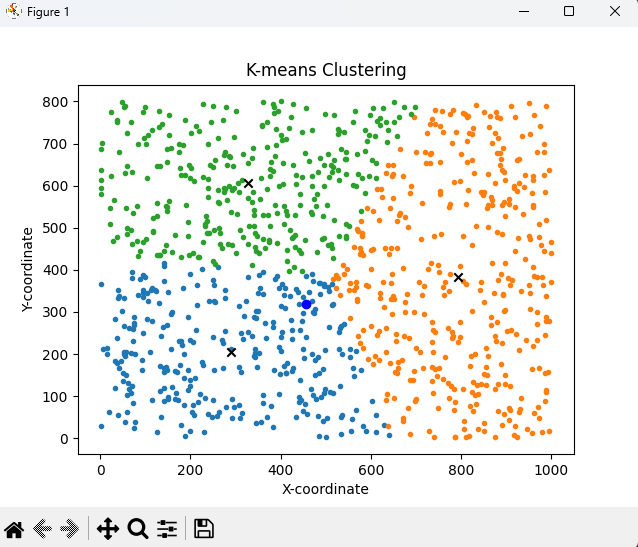
**Рисунок 3.4.** Кластеризация методом k-means++ для n=100.



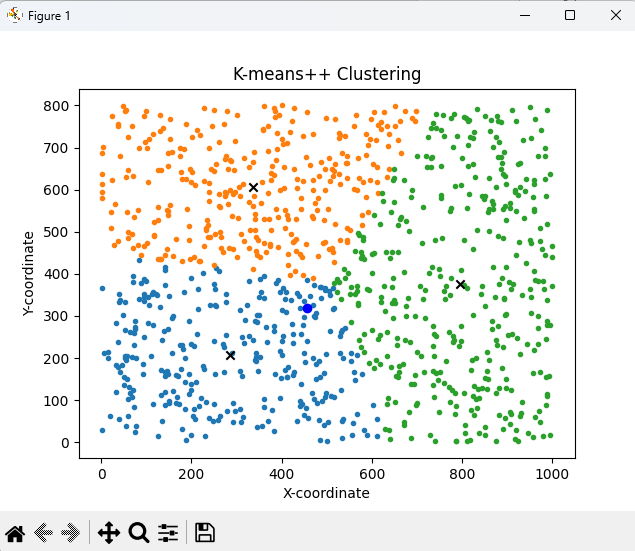
**Рисунок 3.5.** Кластеризация методом k-means для n=500.



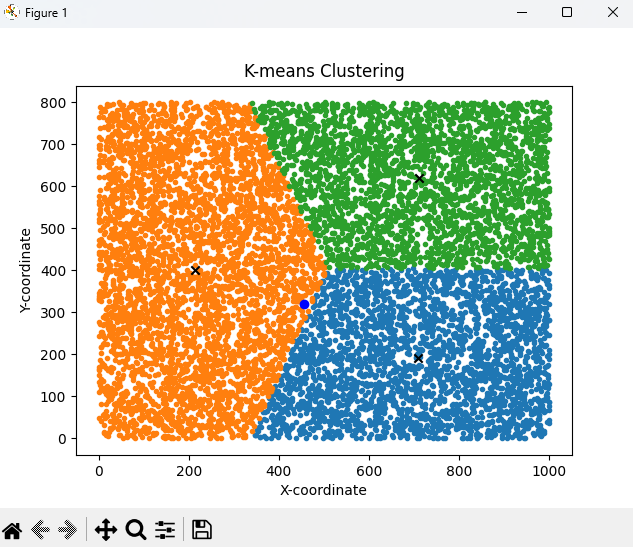
**Рисунок 3.6.** Кластеризация методом k-means++ для n=500.



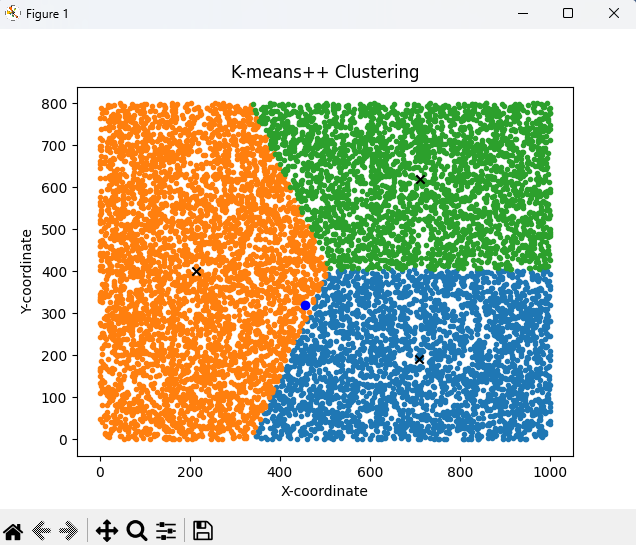
**Рисунок 3.7.** Кластеризация методом k-means для n=1000.



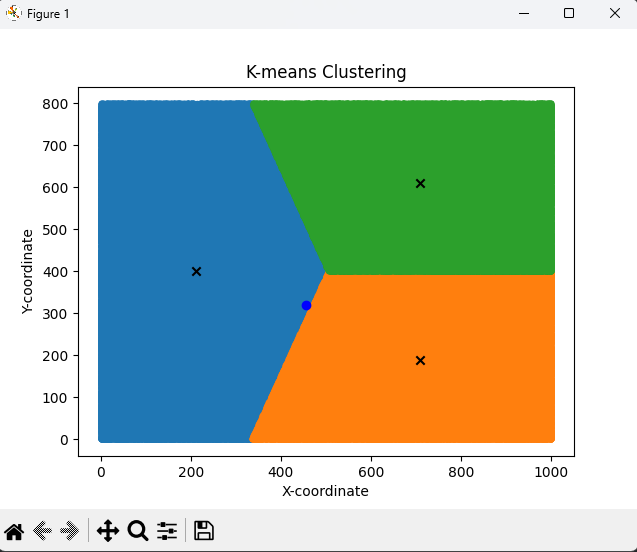
**Рисунок 3.8.** Кластеризация методом k-means++ для n=1000.



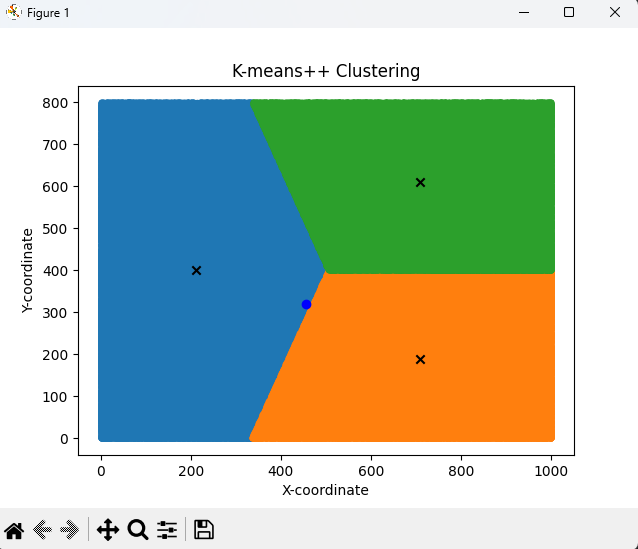
**Рисунок 3.9**. Кластеризация методом k-means для n=10000.



**Рисунок 3.10.** Кластеризация методом k-means++ для n=1000.

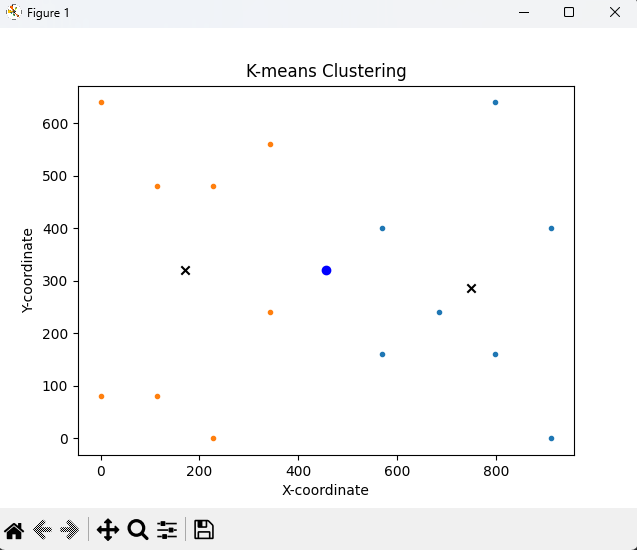


**Рисунок 3.11.** Кластеризация методом k-means для n=1000000.

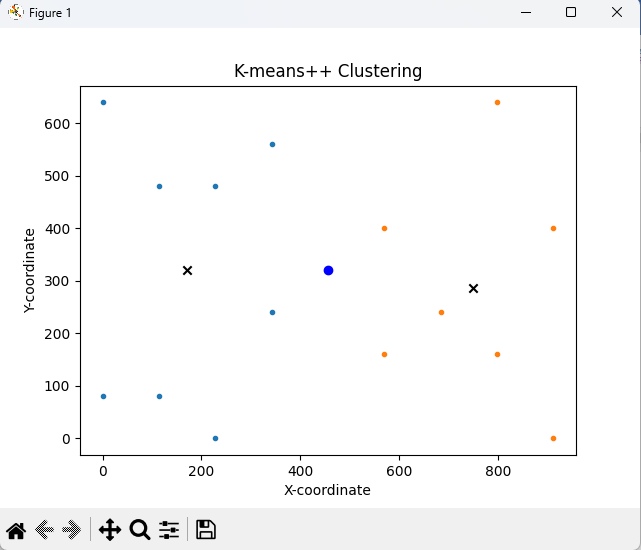


**Рисунок 3.12.** Кластеризация методом k-means++ для n=1000000.

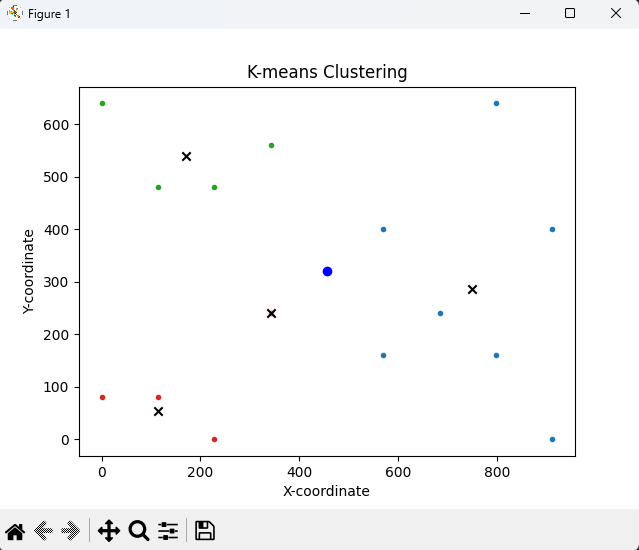
Далее рассмотрим примеры на 15 и 150 точка при различных K.



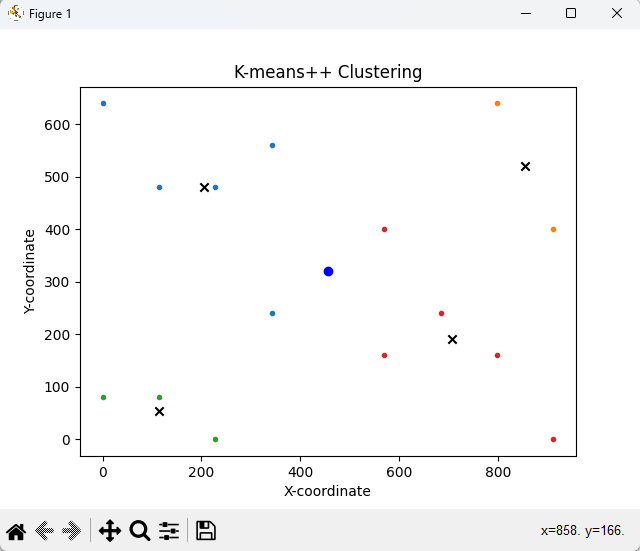
**Рисунок 3.13.** Кластеризация методом k-means для n=15 и K=2.



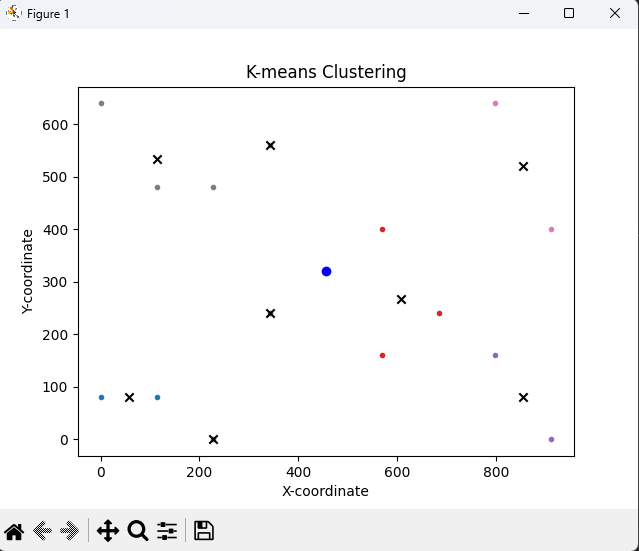
**Рисунок 3.14.** Кластеризация методом k-means++ для n=15 и K=2.



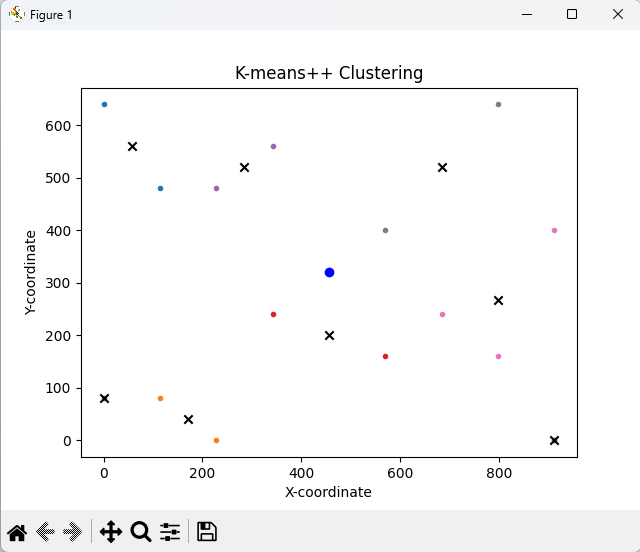
**Рисунок 3.15.** Кластеризация методом k-means для n=15 и K=4.



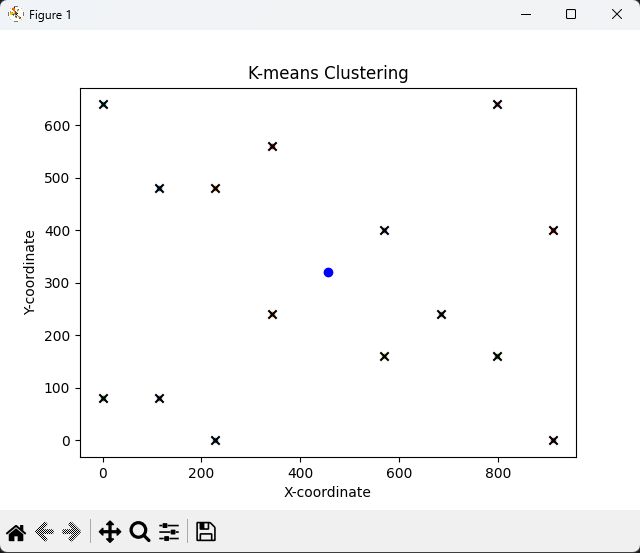
**Рисунок 3.16.** Кластеризация методом k-means++ для n=15 и K=4.



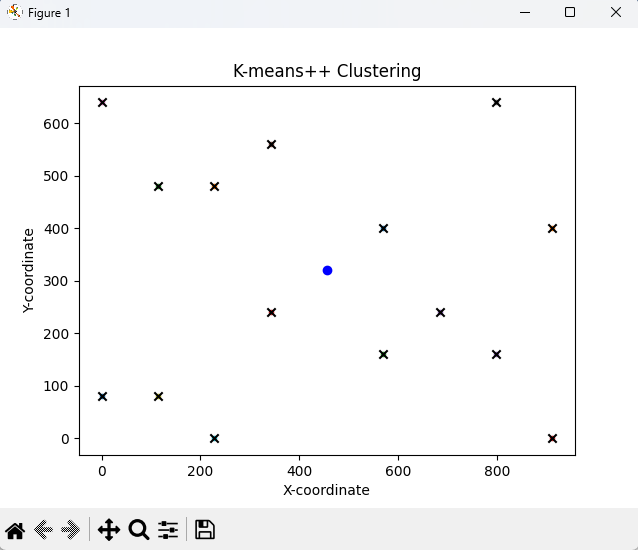
**Рисунок 3.17.** Кластеризация методом k-means для n=15 и K=8.



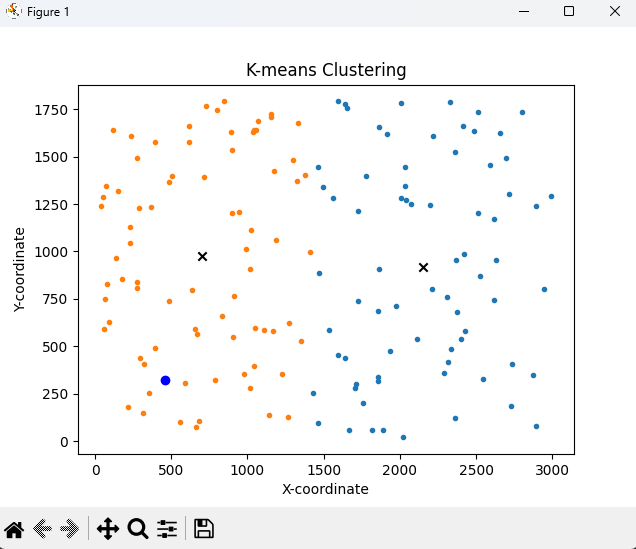
**Рисунок 3.18.** Кластеризация методом k-means++ для n=15 и K=8.



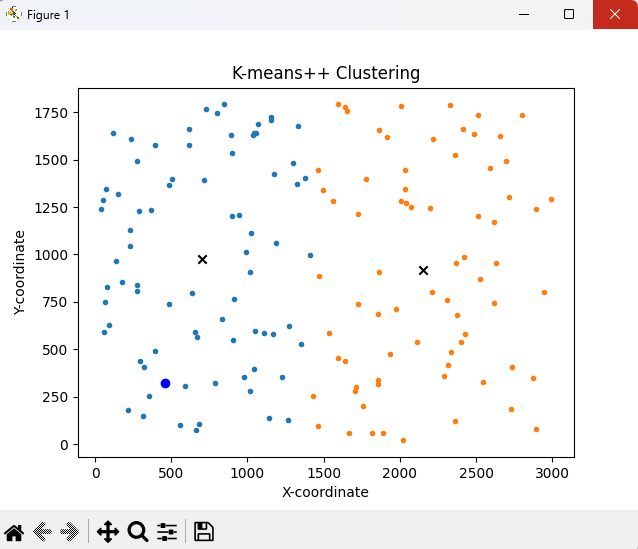
**Рисунок 3.19.** Кластеризация методом k-means для n=15 и K=15.



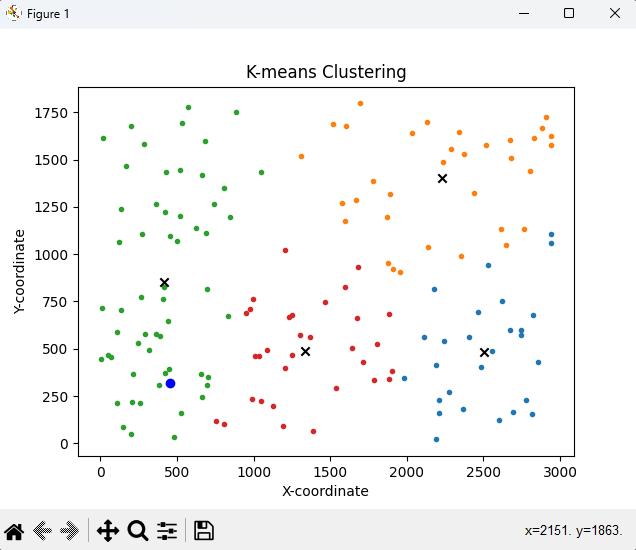
**Рисунок 3.20.** Кластеризация методом k-means++ для n=15 и K=15.



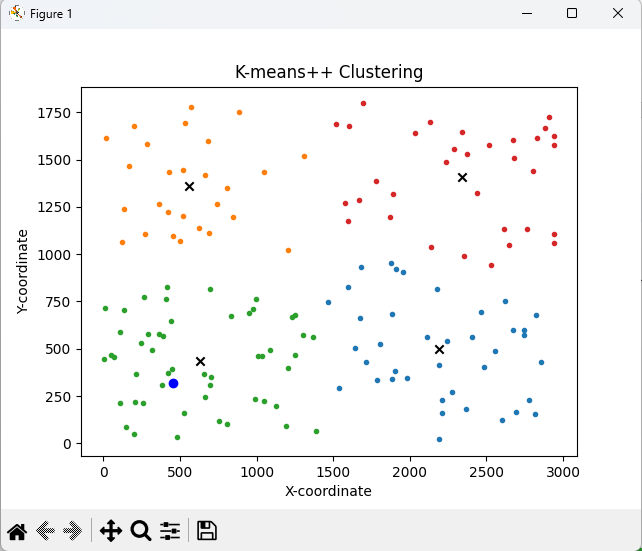
**Рисунок 3.21.** Кластеризация методом k-means для n=150 и K=2.



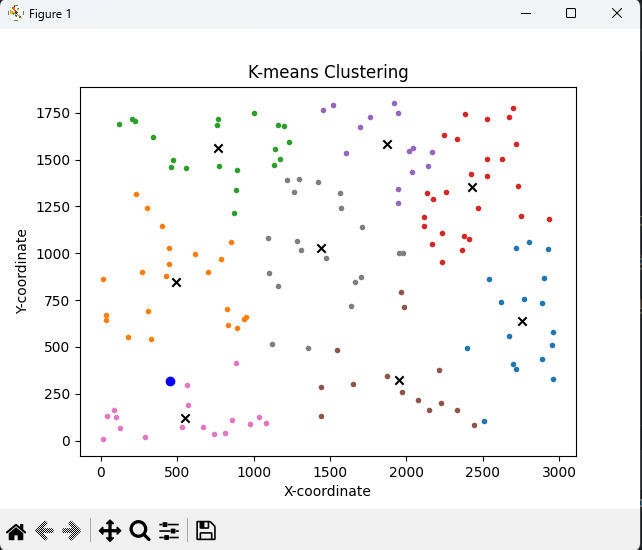
**Рисунок 3.22.** Кластеризация методом k-means++ для n=150 и K=2.



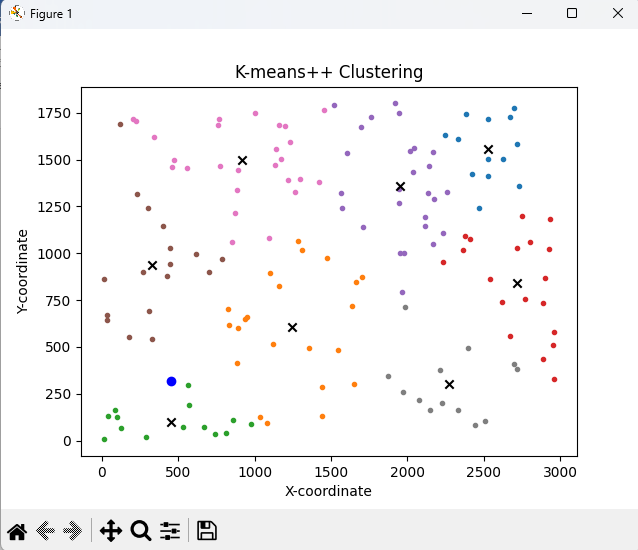
**Рисунок 3.23.** Кластеризация методом k-means для n=150 и K=4.



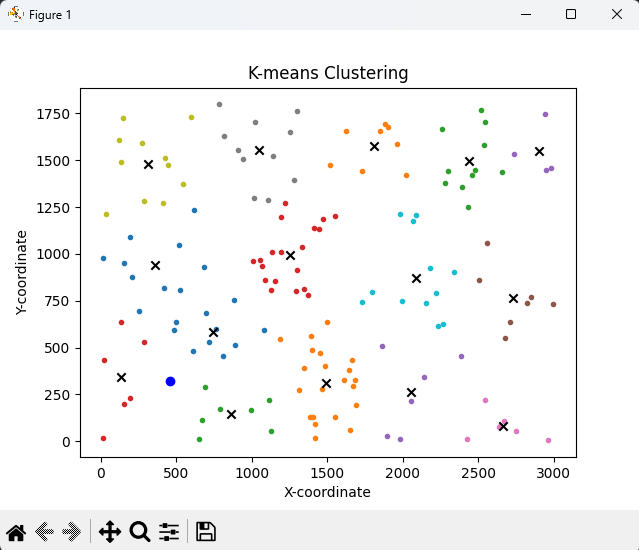
**Рисунок 3.24.** Кластеризация методом k-means++ для n=150 и K=4.



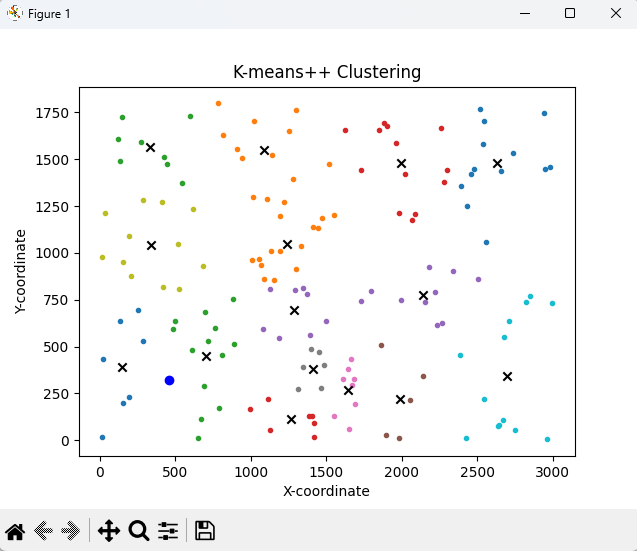
**Рисунок 3.25.** Кластеризация методом k-means для n=150 и K=8.



**Рисунок 3.26.** Кластеризация методом k-means++ для n=150 и K=8.



**Рисунок 3.27.** Кластеризация методом k-means для n=150 и K=15.



**Рисунок 3.28.** Кластеризация методом k-means++ для n=150 и K=15.

|  |  |  |  |
| --- | --- | --- | --- |
| K | n | k-means++ | k-means |
| 4 | 15 | 1.000ms | 2.002ms |
| 8 | 15 | 1.000ms | 1.407ms |
| 15 | 15 | 1.074ms | 1.001ms |
| 4 | 100 | 1.386ms | 1.101ms |
| 8 | 100 | 4.052ms | 2.056ms |
| 15 | 100 | 2.996ms | 2.050ms |
| 4 | 500 | 2.069ms | 4.817ms |
| 8 | 500 | 6.000ms | 6.001ms |
| 15 | 500 | 15.213ms | 6.219ms |
| 4 | 1000 | 10.068ms | 6.999ms |
| 8 | 1000 | 17.077ms | 9.999ms |
| 15 | 1000 | 25.865ms | 17.532ms |
| 4 | 5000 | 51.872ms | 21.003ms |
| 8 | 5000 | 72.002ms | 55.054ms |
| 15 | 5000 | 89.594ms | 82.597ms |
| 4 | 10000 | 136.844ms | 92.610ms |
| 8 | 10000 | 289.301ms | 287.628ms |
| 15 | 10000 | 519.837ms | 486.898ms |
| 4 | 50000 | 489.503ms | 675.104ms |
| 8 | 50000 | 1227.456ms | 1699.153ms |
| 15 | 50000 | 3276.012ms | 2647.407ms |
| 4 | 100000 | 1057.981ms | 1710.361ms |
| 8 | 100000 | 3841.908ms | 3153.755ms |
| 15 | 100000 | 5806.751ms | 5358.012ms |
| 4 | 500000 | 4444.845ms | 8056.886ms |
| 8 | 500000 | 13874.216ms | 14583.002ms |
| 15 | 500000 | 26600.476ms | 23988.471ms |
| 4 | 1000000 | 17700.689ms | 25350.910ms |
| 8 | 1000000 | 29023.706ms | 36778.105ms |
| 15 | 1000000 | 50451.912ms | 50540.761ms |

**Таблица 1.** Результаты работы алгоритмов для теста

На рисунках 3.1. – 3.12. сравнивается доставка разным количествам клиентов при одинаковом количестве ТС, следовательно и кластеров. Как можно заметить в данном случае оба метода выдают примерно одинаковый результат. Однако на рисунках 3.13. – 3.28. при различном количестве машин можно заметить, что k-means++ находит более точное решение.

В таблице 1 приведены результаты замеров времени k-means и k-mean++ при различных n и K. В среднем k-means++ оказался быстрее k-means на 10.4%. k-means++ тратит в среднем 5301,4ms на разбиение на класетры, а k-means 5854,3ms.

**Выводы**

Для задачи маршрутизации с вывозом и доставкой с несколькими транспортными средствами ограниченной грузоподъемности на языке программирования Python был реализован двухэтапный подход решения задачи. Для первого этапа – метод k-средних и метод k-средних++, для второго – метод поиска с запретами. Таблица с результатами дает основания сделать следующие выводы:

1. K-means++ дает более точное решение при разбиении на кластеры
2. K-means++ тратит меньше времени на разбиения в отличие от k-means

# Заключение

В данной работе был рассмотрен класс задач транспортной маршрутизации и подробно исследована одна из задач - задача маршрутизации с вывозом и доставкой с несколькими транспортными средствами ограниченной грузоподъемности. Были изучены подходы к решению данного типа задач. Был выбран двухэтапный метод решения задач: на первом этапе было распределение вершин по ТС, а на втором - формирование маршрутов для каждого ТС. На языке программирования Python был реализован метод k-средних и k-средних++ для первого этапа. Для второго этапа был реализован метод поиска с запретами т.к. на текущем этапе работы не требовалось нахождения более точного маршрута.

Далее были сравнены методы k-means++ и k-means на точность и затраты по времени. Как показал результат k-means++ быстрее на 10.4%, а так же точнее поэтому в ходе дальнейшей работы будет использоваться именно этот метод кластеризации.

# Литература

1. Трифонов Ю.В., Громницкий В.С., Золотов М.Ю. Формирование оптимальных маршрутов доставки товаров автотранспортом // Вестник Нижегородского университета им. Н.И.Лобачевского. 2010. № 6. С. 236-240.
2. Ишкова Е.С. Матричный метод решения задачи маршрутизации с несколькими транспортными средствами с учетом ограничений на массу и объем перевозимого груза // Вестник ИНЖЭКОНА серия: Экономика. 2011. № 1(44). С. 329-333.
3. Маршрутизация с ограничением по грузоподъемности (Capacitated VRP, CVRP) / [Электронный ресурс] // Лекции.ком : [сайт]. — URL: https://lektsii.com/1-119780.html (дата обращения: 12.12.2023).
4. Объясните так, как будто мне 10 лет: простое описание популярного алгоритма кластеризации k-средних / [Электронный ресурс] // proglib : [сайт]. — URL: https://proglib.io/p/obyasnite-tak-kak-budto-mne-10-let-prostoe-opisanie-populyarnogo-algoritma-klasterizacii-k-srednih-2022-12-07?ysclid=lptan0qmkh684150699 (дата обращения: 12.12.2023).
5. Diving Deep into K-Means Clustering: A Scikit-Learn Guide / [Электронный ресурс] // Scicoding : [сайт]. — URL: https://www.scicoding.com/diving-deep-into-k-means-clustering-a-scikit-learn-guide/#:~:text=K%2DMeans%20Clustering%20with%20Scikit%2DLearn.%20Scikit%2DLearn,integrate%20with%20other%20Python%20libraries (дата обращения: 12.12.2023).
6. Vehicle routing problem / [Электронный ресурс] // Википедия : [сайт]. — URL: https://en.wikipedia.org/wiki/Vehicle\_routing\_problem (дата обращения: 12.12.2023).
7. Vehicle Routing Problem / [Электронный ресурс] // google OR-Tools : [сайт]. — URL: https://developers.google.com/optimization/routing/vrp?hl=en#python (дата обращения: 12.12.2023).
8. Dantzig G. B., Ramser J. H. The Truck Dispatching Problem // Management Science. 1959. Vol. 6, No 1. P. 80–91.
9. Berbeglia G., Cordeau J., Laporte G. Dynamic pickup and delivery problems // European journal of operational research. 2010. No 202. P. 8-15.
10. Berbeglia G., Cordeau J., Gribkovskaia I., Laporte G. Static pickup and delivery problems: classification scheme and survey // Springer. 2007. No 15. P. 1-31.
11. Psaraftis Harilaos N. Analysis of an 𝑂(𝑁 2 ) heuristic for the single vehicle many-to-many Euclidean dial-a-ride problem // Transpn Res.-B. 1983. Vol. 17B, No 2. P. 133–145.
12. Harilaos N. A dynamic programming solution to the single vehicle many-tomany immediate request dial-a-ride problem // Transportation science. 1980. Vol. 14. No. 2. P. 385-392.
13. Gribkovskaia I., Laporte G. One-to-one Single Vehicle Pickup and Delivery Problems // Single Vehicle Pickup and Delivery Problems. 2008. P. 359-377.
14. Cordeau J., Laporte G., Ropke S. Recent Models and Algorithms for One-toOne Pickup and Delivery Problems // The vehicle routing problem: latest advances and new challenges. 2008 P. 329-359
15. Hernandez-Perez H., Salazar-Gonzalez J. The multi-commodity one-to-one pickup-and-delivery traveling salesmen problem // European journal of operation research. 2009. Vol. 196. P. 987-995.
16. Anily S, Hassin The swapping problem // Networks. 1992. No. 22. P. 419- 433. 10. Gribkovskaia I., Laporte G., Shlopak A. A tabu search heuristic for a routing problem arising in servicing of offshore oil and gas platforms // Journal of the operational research society. 2008. No 59. P. 1449-1459.
17. Cordeau J., Laporte G. A tabu search heuristic for the static multi-vehicle 33 dial-a-ride problem // Transportation research. 2003. Part B. Vol. 37. P.579- 594.
18. Powell W.B., Bouzanene-Ayari B., Simpo H.P. Dynamic models for freight transportation // Handbooks in operation research and management science: transportation. 2007. Vol. 14. P.285-365.
19. Abdulkader M., Gajpal Y., ElMekkawy T. Hybridized ant colony algorithm for the Multi Compartment Vehicle // Applied Soft Computing. 2015. No 37. P. 196–203.
20. Grandinetti L., Guerrieo F., Pezzella F., Pisacane O. The multi-objective multi-vehicle pickups and delivery problem with time windows // Procediasocial and behavioral sciences. 2014. Vol. 111. P. 203-212.
21. Nagy G., Salhi S. Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries // European Journal of Operational Research. 2005. Vol. 162. No. 1. P. 126-141.
22. Dror M., Laporte G., Trudeau P. Vehicle routing with split deliveries // Discrete applied mathematics. 1994. Vol. 50. No. 3. P.239-254.
23. Vidal T., Crainic T.C., Gendreau M. A Hybrid Genetic Algorithm for Multidepot and Periodic Vehicle Routing Problems // Operations Research. 2012. Vol. 60. No. 3. P. 721-738.
24. Heilporn G., Cordeau J.-F., Laporte G. An Integer L-shaped Algorithm for the Dial-a-Ride Problem with Stochastic Customer Delays // CIRRELT. 2010. No.29.
25. Hernandez-Perez H., Salazar-Gonzalez J. A branch-and-cut algorithm for a traveling salesmen problem with pickup and delivery // Discreet applied mathematics. 2004. Vol. 145. P. 126-139.
26. Hu T.-Y., Chang C.-P. A revised branch-and-price algorithm for dial-a-ride problems with consideration of time-dependent travel cost // Journal of advanced transportation. 2015. Vol. 49. P.700-723.
27. Corominas A., Garcia-Villoria A., Pastor R. Improving parametric Clarke and Wright algorithms by means of iterative empirically adjusted greedy 34 heuristics // SORT. 2014. No 1. P. 3-12.
28. Desrochers M., Verhoog T.W. A Matching Based Savings Algorithm for the Vehicle Routing Problem // GERAD. 1989. No 4. P. 1-19.
29. Steinhaus H. Sur la division des corp materiela en parties // Bull Acd. Polon. Sci. 1956. C1 III Vol IV P. 801-804.
30. Glover F. Tabu search – Part I // ORSA Journal on Computing. 1989. No. 1. P. 190-206.
31. Holland J.H. Adaptation in natural and artificial systems (1975) // The university of Michigan press. 1992.
32. Беллман Р. Динамическое программирование // Изд-во иностранной литературы. 1960. 400 С.
33. Shirokikh V.A., Zakharov V.V. Dynamic adaptive large neighborhood search for inventory routing problem // Advances in intelligent systems and computing. 2015. Vol. 359. P. 231-243.
34. Захаров В.В. Методы и модели прикладной математической логики // Процессы управления и устойчивость. 2015. Т. 2(18). С. 742-776.
35. Networking and emerging optimization [Электронный ресурс]: URL: http://neo.lcc.uma.es/vrp/ (дата обращения: 12.12.2023).
36. Figure 1 / [Электронный ресурс] // burp : [сайт]. — URL: https://www.burp-online.it/content/vehicle-routing-problem-with-pickup-and-delivery-time-windows-k.html (дата обращения: 12.12.2023).
37. Скаков Е.С. Е.С. Скаков - Метод поиска с запретами для оптимизационных задач / Скаков Е.С. [Электронный ресурс] // masterst.donntu : [сайт]. — URL: https://masters.donntu.ru/2018/fknt/solonitsyn/library/article8.htm?ysclid=lpu1w6efv2784333877 (дата обращения: 12.12.2023).
38. ML | K-means++ Algorithm / [Электронный ресурс] // : [сайт]. — URL: https://www.geeksforgeeks.org/ml-k-means-algorithm/ (дата обращения: 12.12.2023).
39. k-means++ / [Электронный ресурс] // Википедия : [сайт]. — URL: https://ru.wikipedia.org/wiki/K-means%2B%2B (дата обращения: 12.12.2023).

# Приложение

## Приложение 1. Функция метода k-средних

def kmeans(X, k, max\_iters=100):

centroids = X[np.random.choice(range(len(X)), k, replace=False)]

for \_ in range(max\_iters):

labels = np.argmin(np.linalg.norm(X[:, np.newaxis] - centroids, axis=-1), axis=-1)

new\_centroids = np.array([X[labels == i].mean(axis=0) for i in range(k)])

if np.all(centroids == new\_centroids):

break

centroids = new\_centroids

return centroids, labels

## Приложение 2. Функция метода k-средних++

def kmeans\_plusplus(X, k, max\_iters=100):

centroids = [X[np.random.choice(range(len(X)))]]

for \_ in range(k - 1):

distances = np.min(np.linalg.norm(X[:, np.newaxis] - centroids, axis=-1), axis=-1)

probabilities = distances / np.sum(distances)

new\_centroid = X[np.random.choice(range(len(X)), p=probabilities)]

centroids.append(new\_centroid)

for \_ in range(max\_iters):

labels = np.argmin(np.linalg.norm(X[:, np.newaxis] - centroids, axis=-1), axis=-1)

new\_centroids = []

for i in range(k):

cluster\_points = X[labels == i]

new\_centroids.append(np.mean(cluster\_points, axis=0))

if np.allclose(centroids, new\_centroids):

break

centroids = new\_centroids

return centroids, labels

## Приложение 3. Функция метода поиска с запретами

def find\_feasible\_routes(depot, locations, centroids, labels, Q, Q\_delivery, tabu\_size=5):

routes = []

visited = set()

for i, centroid in enumerate(centroids):

route = [depot[0]]

current\_location = depot[0]

current\_capacity = Q

tabu\_list = []

while True:

min\_distance = float('inf')

next\_location = None

for j, location in enumerate(locations):

if labels[j] != i:

continue

if location in visited or location in tabu\_list:

continue

if Q\_delivery[j] > current\_capacity:

continue

distance = np.linalg.norm(np.array(current\_location) - np.array(location))

if distance < min\_distance:

min\_distance = distance

next\_location = location

if next\_location is None:

break

route.append(next\_location)

visited.add(next\_location)

current\_location = next\_location

current\_capacity -= Q\_delivery[locations.index(next\_location)]

tabu\_list.append(next\_location)

if len(tabu\_list) > tabu\_size:

tabu\_list.pop(0)

if len(route) > 1:

routes.append(route)

if current\_capacity < Q:

remaining\_locations = set(locations) - visited

remaining\_route = [depot[0]]

for location in remaining\_locations:

remaining\_route.append(location)

routes.append(remaining\_route)

return routes