Министерство образования и науки Российской Федерации

МОСКОВСКИЙ ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ

(государственный университет)

ФАКУЛЬТЕТ АЭРОФИЗИКИ И КОСМЧЕСКИХ ИССЛЕДОВАНИЙ

[«Логистические системы и технологии»](http://mipt.ru/dasr/upload/59b/10-arph4h8obci.pdf)

На правах рукописи

УДК 519.852.33

Подольский Антон Евгеньевич

Оптимизация количества посещения точек при ограничении продолжительности маршрута.

Выпускная квалификационная работа бакалавра

Направление подготовки 27.03.03 «СИСТЕМНЫЙ АНАЛИЗ И УПРАВЛЕНИЕ»

Заведующий кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.Н.Васильев

Научный руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ю. И. Смирнова

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. Е. Подольский

г. Долгопрудный

2015

**Оглавление**

Часть 1. 3

1.1 Введение. 3

1.1.1 Актуальность работы. 3

1.1.3 Методика исследования 5

1.1.4 Научная новизна работы. 5

1.2 Обзор алгоритмов решения ЗМТ 5

1.2.1 Терминология, используемая при обзоре алгоритмов и решении задачи: 5

1.2.1.1 Класс алгоритмов NP 5

1.2.1.2 Жадный алгоритм 7

1.2.2 Алгоритмы, применимые для решения задачи. 7

1.2.2.1 Симплекс-метод 7

1.2.2.2 Алгоритм Свира 11

1.2.2.3 Алгоритм Лепестков 13

1.2.2.4 Алгоритм Кларка-Райта 17

1.2.2.5 Алгоритм Фишера – Джекура. 19

1.2.2.6 Алгоритм Брамела-Симчи-Леви 20

1.2.2.7 Метод ветвей и границ. 21

1.2.2.8 Метаэвристики. 25

1.2.2.9 Алгоритм Османа поиска с исключениями 26

1.2.2.10 Алгоритм Генро-Герца-Лапорте 27

1.2.2.11 Генетический алгоритм. 29

1.3 Выводы 32

Часть 2. Решение задачи. 33

Список литературы 60

Часть 1.

1.1 Введение.

1.1.1 Актуальность работы.

Одним из способов оптимизации затрат на транспортировку грузов является оптимизация маршрутов и расписания передвижения транспортных средств. Оптимизация маршрутов и расписаний требует значительных вычислительных и временных ресурсов. Одной из ключевых функций систем принятия решений в области транспортной логистики является возможность расчета и построения эффективных с точки зрения стоимости объезда маршрутов. Работа посвящена исследованию задачи, состоящей в составлении расписания и оптимизации маршрута посещения транспортными средствами заданного количества точек с возвращением в начальное местоположение после окончания поездки. Данная задача актуальна для различных компаний, которые выполняют дистрибуцию грузов с некоторого склада до заданных точек. Данная задача решалась на примере ФГУП “Почта России”.

Математическая формулировка этой задачи широко известна как задача маршрутизации транспорта (ЗМТ). Существует ряд разновидностей ЗМТ с различными условиями, позволяющими учитывать характеристики ТС и другими факторами, которые позволяют еще более приблизить математическую модель к реальным условиям. ЗМТ является обобщением известной математической задачи коммивояжера. ЗМТ И ЗК принадлежат к классу задач дискретной оптимизации и являются NP-сложными. Не существует алгоритмов, позволяющих однозначно решить задачу и убедиться в оптимальности решения за полиноминальное время. Из-за чрезмерно быстрого роста времени вычислений большинство алгоритмов неприменимы для задач с более чем 15-20 вершинами.

Один из первых приближенных алгоритмов решения ЗМТ был предложен в 1964 г. Г. Кларком и Д. В. Райтом. В последующие годы исследования были продолжены и полученные результаты вошли в группу классических алгоритмов приближенному решению ЗМТ. С середины 90-х годов ХХ века исследования были сосредоточены на метаэвристических алгоритмов. Метаэвристические алгоритмы не являются полностью эвристическими, которые готовы для применения и решения реальных задач, а представляют набор методов для построения законченной эвристики для конкретной задачи. Вжаной их особенностью является способность к преодолению точки локального минимума для продолжения поиска, поэтому потенциально они способны находить более качественные решения по сравнению с классическими эвристиками.

Наибольший интерес представляют следующие методы или алгоритмы:

1. Поиск с исключениями
2. моделируемый и детерминированный отжиг
3. генетический алгоритм
4. Метод ветвей и границ

Обилие работ, которые были посвящены метаэвристикам, создало ситуацию, когда невозможно опрделить наилучший алгоритм для практического внедрения. Метаэвристики содержат дискретные и непрерывные параметры, управляющие их работой и требующие выполнения процедуры вариации значений для получения законченной эвристики. Подбор параметров необходимо выполнять не только для разных типов задач, но иногда и для каждого нового набора входных данных. Ни один из предложенных выше алгоритмов не позволяет однозначно определить глобальный оптимум маршрута. Актуальность данной работы заключается в определении лучшего алгоритма для поиска глобального минимума решения ЗМТ.

1.1.2 Цель работы.  
Целью данной работы является выбор лучшего алгоритма для решения задачи оптимизации маршрутов и расписания движения транспортных средств от дистрибуционных центров к отделениям Почты России.

1.1.3 Методика исследования

В ходе решения задачи использовался анализ существующей литературы по теме решения ЗМТ и ЗК, а также методы линейного программирования и существующие программные продукты

1.1.4 Научная новизна работы.

На данный момент не существует алгоритма, который позволяет однозначно решить задачу оптимизации маршрута и оптимально решить ЗМТ. В данной работе приводится анализ наиболее известных алгоритмов решения задачи и подбор необходимого для решения конкретной задачи ФГУП “Почта России”.

1.2 Обзор алгоритмов решения ЗМТ

1.2.1 Терминология, используемая при обзоре алгоритмов и решении задачи:

1.2.1.1 Класс алгоритмов NP

В теории алгоритмов классом NP (от англ. *non-deterministic polynomial*) называют множество задач распознавания (*англ.*), решение которых при наличии некоторых дополнительных сведений (так называемого *сертификата решения*) можно «быстро» (за время, не превосходящее полинома от размера данных) проверить на машине Тьюринга.

Эквивалентно класс NP можно определить как содержащий задачи, которые можно «быстро» решить на недетерминированной машине Тьюринга.

Класс сложности NP определяется для множества языков, то есть множеств слов над конечным алфавитом \Sigma. Язык *L* называется принадлежащим классу NP, если существуют двуместный предикат R(x,\, y) из класса P (то есть вычислимый за полиномиальное время) и константа c>0 такие, что для всякого слова *x* условие «*x* принадлежит *L*» равносильно условию «найдётся *y* длины меньше |x|^c такой, что верно R(x,\, y)» (где |*x*| — длина слова *x*). Слово называется **сертификатом** принадлежности *x* языку *L*. Таким образом, если у нас есть слово, принадлежащее языку, и ещё одно слово-свидетель ограниченной длины (которое бывает трудно найти), то мы быстро сможем удостовериться в том, что *x* действительно принадлежит *L*.

Эквивалентное определение можно получить, используя понятие недетерминированной машины Тьюринга (то есть такой машины Тьюринга, у программы которой могут существовать разные строки с одинаковой левой частью). Если машина встретила «развилку», то есть неоднозначность в программе, то дальше возможны разные варианты вычисления. Предикат R(x), который представляет данная недетерминированная машина Тьюринга, считается равным единице, если существует хоть один вариант вычисления, возвращающий 1, и нулю, если все варианты возвращают 0. Если длина вычисления, дающего 1, не превосходит некоторого многочлена от длины *x*, то предикат называется принадлежащим классу NP. Если у языка существует распознающий его предикат из класса NP, то язык называется принадлежащим классу NP. Это определение эквивалентно приведённому выше: в качестве свидетеля можно взять номера нужных веток при развилках в вычислении. Так как для *x* принадлежащему языку длина всего пути вычисления не превосходит многочлена от длины *x*, то и длина свидетеля также будет ограничена многочленом от длины *x*.

1.2.1.2 Жадный алгоритм

Жадный алгоритм (англ. *Greedy algorithm*) — алгоритм, заключающийся в принятии локально оптимальных решений на каждом этапе, допуская, что конечное решение также окажется оптимальным. Известно, что если структура задачи задается матроидом, тогда применение жадного алгоритма выдаст глобальный оптимум. Жадные алгоритмы не являются оптимальными для решения ЗК И ЗМТ и для всех NP-сложных задач.

1.2.2 Алгоритмы, применимые для решения задачи.

1.2.2.1 Симплекс-метод

Симплекс-метод — алгоритм решения оптимизационной задачи линейного программирования путём перебора вершин выпуклого многогранника в многомерном пространстве.

Задача линейного программирования состоит в том, что необходимо максимизировать или минимизировать некоторыйлинейный функционал на многомерном пространстве при заданных линейных ограничениях.

Заметим, что каждое из линейных неравенств на переменные ограничивает полупространство в соответствующем линейном пространстве. В результате все неравенства ограничивают некоторый многогранник (возможно, бесконечный), называемый также полиэдральным комплексом. Уравнение *W*(*x*) = *c*, где *W*(*x*) — максимизируемый (или минимизируемый) линейный функционал, порождает гиперплоскость *L(c)*. Зависимость от *c* порождает семейство параллельных гиперплоскостей. Тогда экстремальная задача приобретает следующую формулировку — требуется найти такое наибольшее *c*, что гиперплоскость *L(c)* пересекает многогранник хотя бы в одной точке. Заметим, что пересечение оптимальной гиперплоскости и многогранника будет содержать хотя бы одну вершину, причём, их будет более одной, если пересечение содержит ребро или *k*-мерную грань. Поэтому максимум функционала можно искать в вершинах многогранника. Принцип симплекс-метода состоит в том, что выбирается одна из вершин многогранника, после чего начинается движение по его рёбрам от вершины к вершине в сторону увеличения значения функционала. Когда переход по ребру из текущей вершины в другую вершину с более высоким значением функционала невозможен, считается, что оптимальное значение *c* найдено.

Последовательность вычислений симплекс-методом можно разделить на две основные фазы:

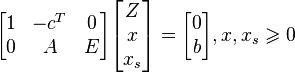
1. нахождение исходной вершины множества допустимых решений,
2. последовательный переход от одной вершины к другой, ведущий к оптимизации значения целевой функции.

При этом в некоторых случаях исходное решение очевидно или его определение не требует сложных вычислений, например, когда все ограничения представлены неравенствами вида «меньше или равно» (тогда нулевой вектор совершенно точно является допустимым решением, хотя и, скорее всего, далеко не самым оптимальным). В таких задачах первую фазу симплекс-метода можно вообще не проводить. Симплекс-метод, соответственно, делится на однофазный и двухфазный.

Рассмотрим следующую задачу линейного программирования:

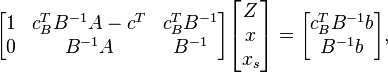
c^Tx \to \max, Ax\leqslant b, x\geqslant 0, b\geqslant 0.

Теперь поставим эту задачу в эквивалентной *усиленной* форме. Необходимо максимизировать *Z*, где:



Здесь *x* — переменные из исходного линейного функционала, *x*s — новые переменные, дополняющие старые таким образом, что неравенство переходит в равенство, *c* — коэффициенты исходного линейного функционала, *Z* — переменная, которую необходимо максимизировать. Полупространства x\geqslant 0 и x_s\geqslant 0 в пересечении образуют многогранник, представляющий множество допустимых решений. Разница между числом переменных и уравнений даёт нам число степеней свободы. Проще говоря, если мы рассматриваем вершину многогранника, то это число рёбер, по которым мы можем продолжать движение. Тогда мы можем присвоить этому числу переменных значение 0 и назвать их *«непростыми»*. Остальные переменные при этом будут вычисляться однозначно и называться *«простыми»*. Полученная точка будет вершиной в пересечении соответствующих непростым переменным гиперплоскостей. Для того, чтобы найти т. н. *начальное допустимое решение* (вершину, из которой мы начнём движение), присвоим всем изначальным переменным *x* значение 0 и будем их считать непростыми, а все новые будем считать простыми. При этом *начальное допустимое решение* вычисляется однозначно : \mathbf{x}_{s i}=\mathbf{b}_i.

Теперь приведём шаги алгоритма. На каждом шаге мы будем менять множества простых и непростых векторов (двигаться по рёбрам), и матрица будет иметь следующий вид:



где *c*B — коэффициенты вектора *c*, соответствующие простым переменным (переменным *x*s соответствуют 0), *B* — столбцы [\mathbf{A}\mathbf{E}], соответствующие простым переменным. Матрицу, образованную оставшимися столбцами обозначим *D*. Почему матрица будет иметь такой вид поясним в описании шагов алгоритма.

Первый шаг.

Выбираем начальное допустимое значение, как указано выше. На первом шаге *B* — единичная матрица, так как простыми переменными являются *x*s. *c*B — нулевой вектор по тем же причинам.

Второй шаг

Покажем, что в выражении (c^T_BB^{-1}A - c^T)x+(c_B^TB^{-1})x_s только непростые переменные имеют ненулевой коэффициент. Заметим, что из выражения Ax + x_s = b простые переменные однозначно выражаются через непростые, так как число простых переменных равно числу уравнений. Пусть x' — простые, а x'' — непростые переменные на данной итерации. Уравнение Ax + x_s = b можно переписать, как Bx' + Dx'' = b. Умножим его на B^{-1} слева: x'+B^{-1}Dx''=B^{-1}b. Таким образом мы выразили простые переменные через непростые, и в выражении B^{-1}Ax+B^{-1}x_s, эквивалентному левой части равенства, все простые переменные имеют единичные коэффициенты. Поэтому, если прибавить к равенству Z-c^Tx=0равенство c^T_BB^{-1}Ax+c_B^TB^{-1}x_s, то в полученном равенстве все простые переменные будут иметь нулевой коэффициент — все простые переменные вида *x* сократятся, а простые переменные вида *xs* не войдут в выражение c_B^TB^{-1}x_s.

Выберем ребро, по которому мы будем перемещаться. Поскольку мы хотим максимизировать *Z*, то необходимо выбрать переменную, которая будет более всех уменьшать выражение

(c^T_BB^{-1}A - c^T)x+(c_B^TB^{-1})x_s.

Для этого выберем переменную, которая имеет наибольший по модулю отрицательный коэффициент. Если таких переменных нет, то есть все коэффициенты этого выражения неотрицательны, то мы пришли в искомую вершину и нашли оптимальное решение. В противном случае начнём увеличивать эту непростую переменную, то есть перемещаться по соответствующему ей ребру. Эту переменную назовём *входящей*.

Третий шаг

Теперь необходимо понять, какая простая переменная первой обратится в ноль по мере увеличения входящей переменной. Для этого достаточно рассмотреть систему:

 \begin{bmatrix} B^{-1}A B^{-1} \end{bmatrix} \begin{bmatrix} x\\ x_s \end{bmatrix}=B^{-1}b 

При фиксированных значениях непростых переменных система однозначно разрешима относительно простых, поэтому мы можем определить, какая из простых переменных первой достигнет нуля при увеличении входящей. Эту переменную назовем *выходящей*. Это будет означать, что мы натолкнулись на новую вершину. Теперь входящую и выходящую переменную поменяем местами — входящая «войдёт» в простую, а выходящая из них «выйдет» в непростые. Теперь перепишем матрицу *B* и вектор *cB* в соответствии с новыми наборами простых и непростых переменных, после чего вернёмся ко второму шагу. *x''*

Поскольку число вершин конечно, то алгоритм однажды закончится. Найденная вершина будет являться оптимальным решением.

1.2.2.2 Алгоритм Свира

Составление кольцевых маршрутов в первом приближении может осуществляться методом, известным как алгоритм Свира или алгоритм дворника-стеклоочистителя. Зададим положение потребителя материального потока в полярной системе координат. Полюс системы - точку 0, разместим в месте дислокации распределительного склада. Выберем первоначальное, нулевое, положение полярной оси f= 0. Положение потребителя определяется расстоянием от центра и углом f, который образован полярной осью, т.е. лучом, исходящим из точки 0 и направленным на потребителя.

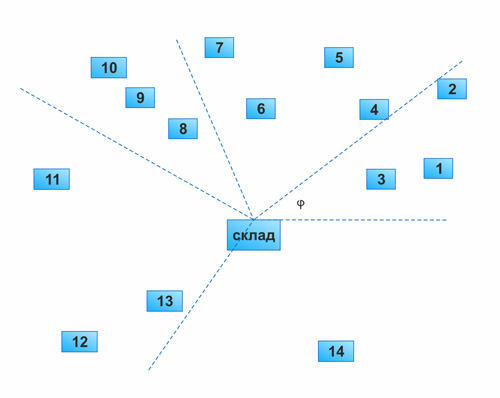
Суть алгоритма Свира заключается в том, что полярная ось, подобно щетке дворника-стеклоочистителя, начинает постепенно вращаться против (или по) часовой стрелки, "стирая" при этом с координатного поля изображенные на нем потребители материального потока. Как только сумма заказов "стертых" магазинов достигнет вместимости транспортного средства, фиксируется сектор, обслуживаемый одним кольцевым маршрутом, и намечается путь объезда потребителей.

Следует отметить, что данный метод дает хорошие результаты на евклидовой транспортной сети, т.е. в том случае, когда расстояние между узлами транспортной сети по существующим дорогам прямо пропорционально расстоянию по прямой.

На кольцевые маршруты кроме ограничений по вместимости могут накладываться дополнительные требования, например, ограничения по времени. Если окажется, что время движения по определенному кольцевому маршруту больше допустимого, необходимо этот сектор уменьшить, увеличив соответственно соседний сектор. Необходимые уменьшения сектора выполняются и при наличии других ограничений.

Построение следующего сектора начинается лишь после того, как в настоящем секторе будет получен допустимый кольцевой маршрут. Формирование кольцевых маршрутов завершается при полном обороте "стирающего" луча.

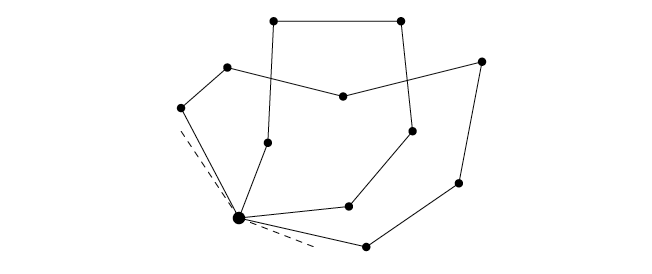
Алгоритм Свира позволяет разделить всю обслуживаемую зону на несколько секторов. В пределах каждого сектора составление кольцевого маршрута может осуществляться посредством решения задачи различных оптимизационных задач, в том числе и задачи коммивояжера.



1.2.2.3 Алгоритм Лепестков

Данный алгоритм – развитие идей предыдущего метода. В своей работе алгоритм первоначально строит избыточное множество маршрутов и выполняет выборку из него при помощи решения задачи о разбиении множества.

Основное понятие этого алгоритма – это понятие r-лепестка. Под r-лепестком здесь подразумевается множество из r маршрутов, которые в своей совокупности покрывают все вершины, расположенные на плоскости внутри некоторого сектора с центром в депо.



На первом шаге работы необходимо сгенерировать множества r-лепестков, которые должны покрывать все вершины исходного набора данных. Алгоритм, описываемый здесь для примера, генерирует только 1-лепески и 2- лепестки. Набор лепестков строится при помощи алгоритма Свира.

Процедура построения 1-лепестка на некотором множестве вершин S со включением в маршрут депо фактически является решением ЗК. Авторы алгоритма применяли для этих целей алгоритм . – трехфазовый алгоритм приближенного решения ЗК, который первоначально строит некоторую оболочку для всех вершин, а затем выполняет добавление вершин, не вошедших в полученное предварительное решение. Последней фазой работы является процедура оптимизации при помощи алгоритма 4-опт.

Процедура построения 2-лепестка на некотором множестве вершин S выглядит следующим образом: прежде всего выбираются две максимально удаленные вершины, которые используются для начального наполнения маршрутов, оставшихся вершины затем добавляются по очереди в один из них по принципу самой выгодной вставки и с соблюдением наложенных ограничений. После нового добавления запускается процедура 4-опт для оптимизации промежуточного решения. Возможно применение разных подходов для поиска двух начальных вершин и процедуры добавления.

Теперь рассмотрим процедуру, используемую для получения всех 1-лепестков и 2-лепестков. Без потери общности предположим, что все вершины пронумерованы в порядке увеличения угла в полярной системе координат с центром в депо. Определим, что , если j>n. Множество лепестков строится при помощи следующих действий:

1. Установим i=0.

2. Установим i=i+1. Если i>n, то завершаем работу.

3. Определим множество вершин , запомним его и вычислим стоимость соответсвующего начального маршрута . Установим j=i+1. Определим множество вершин и соответствующий маршрут. Если он не удовлетворяет наложенным ограничениям, то переходим на шаг 5. В противном случае запомним , соответсвующий маршрут и его стоимость .

4. Установим j=j+1 и . Если общее количество товара для развозки по превышает предел грузоподъемности , то переходим на шаг 5. В противном случае выполняем процедуру построения 1-лепестка при . Если не удачется найтии возможный 1-лепесток, то переходим на шаг 5. Если маршрут был найден, то запоминаем соответсвующий маршрут и его стоимость .

5. Если текущее количество товара для развозки по превосходит удвоенную максимальную грузоподъемность, то переходим на шаг 6. В противном случае выполним процедуру построения 2-лепестка при . Если не удалось найти допустимый 2-лепесток, то переходим на шаг 6. Если 2-лепесток был найден, запомним , соответсвующий маршрут и его стоимость . Установим j=j+1 и . Повторим этот шаг с начала.

6. Некоторые из лепестков, только что созданных, могут доминировать. Если j=2, то переходим на шаг 2. В противном случае для h=j,j-1,…,3 рассмотрим множество вершин и последнюю вершину , помещенную в . Пусть - стоимость соответствующего лепестка. Если , то удаление из должно приводить к лепестку, стоимость которого не превышает . Лепесток, определенный в , затем замещается лепестком, определенным над . Переходим на шаг 2.

Осталось рассмотреть процедуру выбора подмножества лепестков для окончательного решения задачи. После того, как все 1-лепестки и 2-лепестки определены, оптимальная их комбинация может быть найдена путем решении задачи о разбиении множества.

Минимизируем

соблюдая ограничения

где L - множество кандидатов r-лепестков (r = 1 или r = 2), стоимость лепестка l, и = 1 тогда и только тогда, когда принадлежит лепестку l. Поставленная задача о разбиении множества может быть решена за полиномиальное время.

В качестве дополнительной оптимизации можно провести проверку, нет ли возможности объединить некоторые маршруты, не нарушая наложенные ограничения. Если слияние некоторых маршрутов было проведено, то повторный запуск процедуры **1**-лепестка позволит перестроить решение на множестве вершин из маршрутов, вошедших в слияние.

1.2.2.4 Алгоритм Кларка-Райта

Алгоритм Кларка-Райта относится к классическим конструктивных алгоримов для решения ЗМТ. Его идея основана на процессе слияния мелких маршрутов в более крупные, производимого до тех пор, пока есть возможность уменьшить суммарную стоимость объезда. Особоую роль в этом алгоритме играет понятие «Сбережения» - снижения общей стоимости решения, получаемое при объединении двух маршрутов. Рассмотрим ситуацию, когда маршрут (0,…,i,0) и маршрут (0,…,j,0) могут быть совмещены в единую последовательность (0,…,I,j,…,0). Сбережением явлется изменение расстояния, равное , если оно больше нуля, где – расстояние между соответствующими вершинам. Алгоритм применяется в случаях, когда количество экипажей не определено заранее и его можно вычислять в ходе работы. Его можно использовать как для симметричных задач, так и для несимметричных, однако качество работы для симметричных случаев заметно ухудшается. Известны два варианта реализации: параллельный и последовательный. В обоих случаях для них требуется предварительное выполнение подготовительного этапа. Он заключается в

* + - 1. В вычислении сбережений для i,j = 1,…,n и
      2. Создании n маршрутов транспортных средств (0,i,0) для

i = 1,…,n.

* + - 1. Сортировке сбережений в порядке убывания.

Рассмотрим в начале параллельный вариант алгоритма:

1. Просматриваем построенный список сбережений с его начала и выполняем следующие действия:

2. Для текущего элемента списка определяем, существуют ли два маршрута, один из которых содержит дугу или ребро (0,j), второй или ребро (i,0), и которые могут быть соединены в один общий маршрут.

3. Если такие маршруты найдены, то выполняем их объединение, удаляя (0,j) и (i,0), а затем добавляя (i,j).

Последовательный вариант можно представить следующим образом:

1. Для всех маршрутов (0,i,…,j,0) выполним следующие действия:

2. Проведем поиск элемента в списке сбережений или , который может быть использован для объединения текущего маршрута с некоторым, содержащим дугу (ребро) (k,0) или (0,l).

3. Если элемент был найден, то выполним объединение и применим процедуру еще раз к новому маршруту.

4. Если сбережение найти не удалось, то перейдем к следующему маршруту и продолжим работу.

5. Завершим работу, когда объединения более невозможны.

Вычислительные результаты сравнения показывают, что параллельный вариант алгоритма дает результаты лучше, чем последовательный.

Одним из недостатков алгоритма Кларка-Райта заключается в том, что эффективность его работы падает по мере приближения к концу вычислений. Так же алгоритм Кларка-Райта может оказаться неэффективным с точки зрения времени исполнения, т.к. во всех вариантах все выигрыши должны быть вычислены, сохранены и отсортированы. Различные авторы предпринимали попытки предложить расширения, уменьшающие время исполнения и использования памяти, необходимые для работы этого алгоритма. Большинство из них выполнено в 70-е годы, когда многие исследователи еще работали с вычислительно техникой, значительно менее производительной, чем в текущий момент.

При реализации стратегии, основанной на понятии сбережения, необходимо уделять внимание двум важным аспектам: методу нахождения элемента наибольшего сбережения в списке и требованиям к хранению данных в памяти. Здесь поиск наибольшего сбережения – самый трудоемкий процесс. Для решения задачи применяются три основных подхода:

1. Использования полного алгоритма сортировки. Алгоритм сортировки подбирается исходя из конкретных условий задачи.

2. Использование ограниченной сортировки при помощи построения кучи.

3. Использование специального итеративного алгоритма для вычисления максимального сбережения.

Алгоритм Кларка-Райта утратил свое значение как самостоятельного подхода, т.к. предложено много более эффективных методов, но часто используется как способ нахождения начального решения для последующего улучшения.

1.2.2.5 Алгоритм Фишера – Джекура.

Данный алгоритм использует для формирования кластеров не геометрический метод, а обощенную задачу о назначениях. Количество транспортных средств k в этом методе предполагается заданным явно. Опишем работу алгоритма:

1. Выберем по одной вершине из множества всех вершин V для начального заполнения каждого кластера k.

2. Вычисляем стоимость назначения каждой вершины i каждому кластеру как

3. Решаем обобщенную задачу о назначениях со стоимостями , с учетом весов клиентов и грузоподъемности транспортных средств Q.

4. Решаем ЗК для каждого кластера на основе результатов, найденных при помощи обобщенной задачи о назначениях.

1.2.2.6 Алгоритм Брамела-Симчи-Леви

Алгоритм Бармела-Симчи-Леви – один из классических двухфазовых алгоритмов. В его описании используется понятие «Ядра кластера», под которым подразумевается некоторая вершина, используемая для начального наполнения кластера.

В алгоритме Брамела-Симчи-Леви ядра кластеров определяются путем решения задачи размещения с ограничениями на мощности. Авторы предлагают определить К ядер кластеров из n местоположений клиентов. Последующее распределение оставшихся клиентов предлагается выполнять так, чтобы суммарное расстояние от всех клиентов до близжайших ядер кластеров было бы минимальным, при этом отслеживая, чтобы общая потребность в товаре для каждого ядра не превышала установленную константу Q.

Рассмотри частичный маршрут k, описываемый вектором . Пусть и – длина точного решения ЗК на множестве . Тогда стоимость добавления неиспользованной вершины i в маршрут k равна . Точное вычисление может оказаться очень трудоемким, поэтому предлагаются две аппроксимации :

1. Прямая стоимость .

2. Стоимость ближайшей вставки

Авторы алгоритма показали, что алгоритм с использованием первого варианта получается асимптотически оптимальным.

1.2.2.7 Метод ветвей и границ.

Задачи дискретной оптимизации имеют конечное множество допусти- мых решений, которые теоретически можно перебрать и выбрать наилуч- шее (дающее минимум или максимум целевой функции). Практически же зачастую это бывает неосуществимо даже для задач небольшой размерно- сти.

В методах неявного перебора делается попытка так организовать пере- бор, используя свойства рассматриваемой задачи, чтобы отбросить часть допустимых решений. Наибольшее распространение среди схем неявного перебора получил метод ветвей и границ, в основе которого лежит идея последовательного разбиения множества допустимых решений. На каждом шаге метода элементы разбиения (подмножества) подвергаются анализу – содержит ли данное подмножество оптимальное решение или нет. Если рассматривается задача на минимум, то проверка осуществляется путем сравнения нижней оценки значения целевой функции на данном подмно- жестве с верхней оценкой функционала. В качестве оценки сверху ис- пользуется значение целевой функции на некотором допустимом решении. Допустимое решение, дающее наименьшую верхнюю оценку, называют рекордом. Если оценка снизу целевой функции на данном подмножестве не меньше оценки сверху, то рассматриваемое подмножество не содержит решения лучше рекорда и может быть отброшено. Если значение целевой функции на очередном решении меньше рекордного, то происходит смена рекорда. Будем говорить, что подмножество решений просмотрено, если установлено, что оно не содержит решения лучше рекорда.

Если просмотрены все элементы разбиения, алгоритм завершает работу, а текущий рекорд является оптимальным решением. В противном случае среди непросмотренных элементов разбиения выбирается множество, являющееся в определенном смысле перспективным. Оно подвергается разбиению (ветвлению). Новые подмножества анализируются по описан- ной выше схеме. Процесс продолжается до тех пор, пока не будут про- смотрены все элементы разбиения.

Формальное описание.

Пусть рассматривается задача вида , где - вещественная функция, а D – конечное множество допустимых решений. Пусть . Функцию b(d), ставящую в соответствие множеству d разбиение его на подмножества , будем называть ветвлением.

Вещественная функция H(d) называется нижней границей для d, если

1.

2. На одноэлементном множестве {x} верно равенство H({x})=f(x)

Алгоритм, реализующий метод ветвей и границ, состоит из последовательности однотипных шагов. На каждом шаге известен рекорд и подмножества непросмотренных решений. В начале работы алгоритма - произвольный элемент множества D или пустое множество (на пустом множестве положим значение фунционала равным бесконечности).

На каждом шаге алгоритм начинает работу с проверки элементов разбиения. Пусть проверяется множество . Множество отсекается в одном из двух последовательно проверяемых случаев:

1. Если

2. Если и найден такой элемент

Во втором случае происходит смена рекорда .

Пусть - неотсеченные множества. В случае М=0 алгоритм заканчивает работу, и в качестве решения задачи принимается рекорд . При М1 среди множеств выбирается множество для нового ветвления. Пусть таковым является множество . Тогда осуществляется ветвление , в результате которого получаем список множеств . Эти множества нумеруются числами от 1 до L, и начинается новый шаг алгоритма.

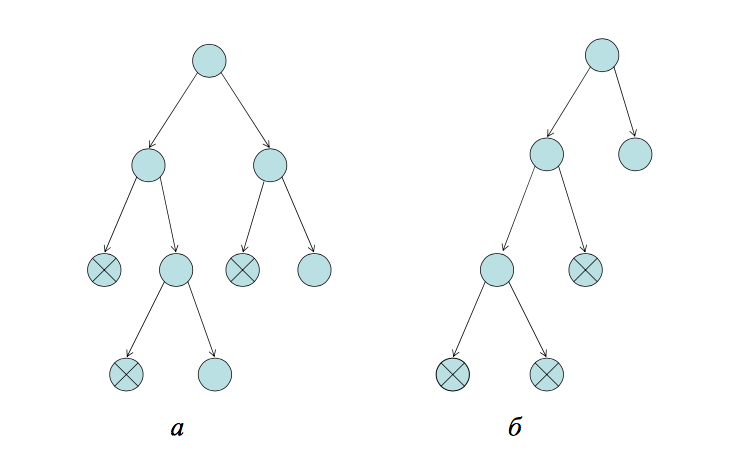
Нетрудно убедиться в том, что описанный алгоритм находит оптимальное решение за конечное число шагов.

Описанная последовательность действий является общей схемой метода ветвей и границ для решения задач на минимум. При решении конкретной задачи следует указать способы построения нижней и верхней оценок, метод ветвления, а также правило выбора перспективного множества для разбиения.

При определении перспективного элемента разбиения в основном применяются две схемы: одновременного и одностороннего ветвления. При одновременном ветвлении функция b может быть применена к любому элементу разбиения. Часто в качестве такого элемента выбирается подмножество c минимальной нижней границей

При одностороннем ветвлении номер разбиваемого подмножества известен заранее. В этом случае, не ограничивая общности, можно считать, что перспективным является подмножество . Отметим, что при односторонней схеме ветвления нет необходимости запоминать все элементы разбиения, достаточно иметь информацию о первом элементе разбиения и объединении остальных элементов.

Разбиения множеств решений удобно представлять в виде дерева решений. На рис приведены примеры одновременной и односторонней схем ветвления. Каждая вершина дерева соответствует некоторому подмножеству решений. Дуги, исходящие из вершины, означают, что на некотором шаге это подмножество отсечь не удалось и оно было разбито на подмножества. Вершины, в которые входят эти дуги, соответствуют подмножествам, полученным в результате ветвления. Зачеркнутые висячие вершины означают отсеченные подмножества. Не зачёркнутые висячие вершины означают отсеченные подмножества. Не зачёркнутые висячие вершины соответствуют не просмотренным множествам, среди которых на следующем шаге алгоритма выбирается подмножество для дальнейшего ветвления.



В схеме одностороннего ветвления выбирается первая вершина на нижнем уровне, а для схемы одновременного ветвления такой вершиной может быть любая. Алгоритм заканчивает работу, если зачеркнуты все висячие вершины дерева ветвлений.

Графическое представление метода ветвей и границ иллюстрирует его суть – отсечение ветвей дерева поиска, которое осуществляется на основании сравнения нижней границы и значения фунциионала на рекорде. Это объясняет название метода.

При использовании описанной выше схемы для решения конкретной задачи необходимо ее уточнение, учитывающее особенности рассматриваемой задачи.

1.2.2.8 Метаэвристики.

Ниже рассматриваются наиболее интересные примеры так называемых метаэвристических алгоритмов. Все они, кроме поиска с исключениями, появились в ходе исследовании различных явлений живой и неживой природы. Организация муравьиных колоний, наблюдения за размножением особей в некоторой популяции и изучение процессов при отжиге металлов послужили основой целой серии идей, применяемых для решения ЗМТ. Отличительной особенностью и в некоторой мере слабой стороной метаэвристических алгоритмов является наличие в их описании большого количества параметров и необходимость подбора их значений при практическом применении. В некоторых случаях процедуру подбора значений параметров необходимо выполнять непосредственно для каждого нового набора данных. Это заметно усложняет использование подобных алгоритмов для решения реальных задач на практике, т. к. требует очень высокой квалификации пользователя программных пакетов на их основе.

Сильной стороной рассматриваемых методов является возможность преодоления локального минимума в процессе поиска. При удачных обстоятельствах это позволяет просмотреть несколько точек локальных минимумов и выбрать из них наилучшую. Такая способность алгоритмов улучшает качество получаемых решений, хотя это приводит к росту требуемого процессорного времени. Описываемые метаэвристические алгоритмы вызывают интерес с разных точек зрения. Поиск с исключениями является одним из самых исследованных подходов, но это не делает остальные идеи менее привлекательными. Согласно мнению исследователей, о чём ещё пойдёт речь ниже, такие методы, как генетические алгоритмы и алгоритмы на основе муравьиных колоний, в настоящий момент ещё относительно слабо изучены, хотя содержат большой простор для дальнейшего поиска улучшений, потенциально способных привести к заметному росту качества. Генетический алгоритм, в частности, использует в своей работе очень мало информации о структуре задачи, и это позволяет строить на его основе методы для решения ЗМТ с очень сложным набором ограничений, с которым не смогут справиться алгоритмы, дающие сейчас наилучшие результаты.

1.2.2.9 Алгоритм Османа поиска с исключениями

Поиск с исключениями (Taboo search) - это стратегия для решения задач

комбинаторной оптимизации. Она основана на общих идеях, описанных. Осман (Osman) удачно её применил для решения ЗМТ. Поиск с исключениями, как и моделируемый отжиг, способен преодолевать точку локального минимума. Метод, позволяющий это делать, основан на использовании так называемой оценочной функции, дающей характеристику планируемого хода с учётом значений исходной целевой функции и содержимого списка исключений. На каждом шаге выбирается такое решение , которое даёт максимальное значение оценочной функции, даже если используется вариант без реального улучшения качества. При выборе ходов без улучшений появляется вероятность возврата к ранее обработанным решениям, наличие списка исключений делает это невозможным. Для любой реализации поиска с исключениями требуется определить следующее:

1. Запрещающую стратегию, которая задаёт, что именно помещается в список исключений.

2. Освобождающую стратегию, которая задаёт, что должно удаляться

из списка исключений.

3. Стратегию, объединяющую поведение двух предыдущих пунктов, включая механизм определения направления поиска и правило выбора следующего решения из на основе стратегии “наилучший подходящий” или “первый подходящий”.

4. Критерий остановки.

В некоторых случаях требуется определение дополнительных стратегий, связанных с особенностями конкретной задачи.

1.2.2.10 Алгоритм Генро-Герца-Лапорте

Алгоритм Генро-Герца-Лапорте, также известный под названием Taburoute, основан на предшествующих попытках применения поиска с исключениями для решения задач ЗМТ, но содержит ряд серьёзных улучшений. Структура окрестности решений в этом алгоритме содержит все решения, которыемогут быть получены из текущего путём удаления вершины из маршрута, где она находилась, и переносе её в другой, который содержит одного из её p ближайших соседей, используя обобщённую процедуру вставки, разработанную Генро, Герцем и Лапортедля решения ЗК. Применение этой процедуры может приводить к удалению существующего маршрута или появлению нового. Другая особенность этого алгоритма заключается в том, что ему позволяется находить решения, нарушающие ограничения грузоподъёмности или длины маршрутов. И тем не менее эти условия не игнорируются полностью. Говоря более точно, здесь целевая функция включает в себя два штрафных компонента: один для штрафования превышений порога грузоподъёмности, а другой \_ для штрафования превышений времени объезда. Они обновляются каждые десять итераций: делятся на **2**, если все десять итераций не нарушали ограничений, или умножаются на **2**, если все десять нарушали его.

Алгоритм Генро-Герца-Лапорте использует неявную форму представления списка исключений. В ситуациях, когда вершина перемещена из маршрута r в маршрут s на итерации t, её возвращение в маршрут r запрещено до итерации t **+** θ, где θ целое число, случайно выбираемое из интервала от **[1**, **2]**. Алгоритм имеет дополнительный механизм, штрафующий частое перемещение одних и тех же вершин. Он позволяет выбирать такие решения, в которых задействованы давно не используемые вершины. Начальное решение определяется процедурой, которая строит несколько вариантов и выбирает наилучший из них.

Ниже приводится краткое описание алгоритма Генро-Герца-Лапорте.

В описании использованы следующие обозначения: W \_ множество вершин кандидатов для перемещения в другие маршруты, q ≤ **|**W**|** -количество вершин из W, с которыми выполнялось пробное перемещение, k - число смежных итераций, на протяжении которых не было найдено улучшений.

1. Сгенерируем начальных решений и выполним поиск с исклю-

чениями с q **= 5** и k **= 50**. Использованное значение q

даёт вероятность выбора одной вершины из каждого маршрута равной

по крайней мере **90%**.

2. Начиная с наилучшего решения, полученного на шаге 1, выполняем поиск с исключениями при q **= 5** и k **= 50**n.

3. Используя наилучший результат, найденный на шаге 2, выполним поиск с исключениями при k **= 50**. Здесь – множествовершин, которые наиболее часто перемещались на шагах 1 и 2. При поиске q выбирается равным **|**W**|**.

Описанный алгоритм на практике оказывается одним из самых эффективных по стоимости решений, но при этом его результаты сложно сравнивать с другими алгоритмами, т. к. он использует систему мягких ограничений при штрафовании вместо более традиционной жёсткой.\_

1.2.2.11 Генетический алгоритм.

Генетический алгоритм (genetic algorithm) \_ методика решения некоторых задач путём имитации процессов, наблюдаемых в ходе эволюции естественной природы.

В чистом виде генетический алгоритм - это общая методика решения задач, для реализации которой требуется относительно небольшое количество информации о предметной области. Как следствие этот подход может быть использован для достаточно большого числа плохо структурированных задач, где специализированные методы не показывают удачных результатов.

В своей работе генетический алгоритм пытается развивать популяцию битовых строк (bitstrings) или “хромосом”, где каждая хромосома кодирует некоторое решение задачи в некоторой её конкретной постановке. Такая эволюция реализуется путём применения нескольких операторов, имитирующих феномены живой природы (воспроизведение, видоизменение и. т. д.). Ниже приводится общее описание генетического алгоритма, а затем будет описано, как применить подобную стратегию к ЗМТ.

Опишем основную схему работы генетического алгоритма. Он начинает выполнение с некоторой начальной популяции хромосом , полученной при помощи датчика случайных чисел. На каждой итерации t = 1,…,T выполняются k раз шаги с 1 по 3, приведённые ниже (k ≤ N/2),

затем выполняется шаг 4.

1. Выбираем две родительские хромосомы из .

2. Порождаем два потомка на основе двух родительских хромосом при помощи оператора скрещивания (crossover operator).

3. Выполняем операцию мутации для каждого потомка (с малой вероятностью).

4. Создаём из путём удаления 2k наихудших решений в и замены их 2k новых потомков.

В приведённом алгоритме параметр T количество поколений, и k ко-

личество выборок в поколении. Наилучшее решение, полученное за T поколений, является окончательным результатом работы алгоритма. На шаге 1 выбор родительских хромосом вероятностно склоняется к выбору наилучших вариантов. На шаге 2 потомок генерируется скрещиванием путем сочетания битовых строк, найденных у родителей. Каждый потомок может быть затем немного модифицирован на шаге 3 путём изменения значения битов в битовой строке с малой вероятностью для каждой битовой позиции. Ожидается, что первоначальное случайно построенное множество будет улучшено в ходе описанного процесса. Такого вывода придерживаются некоторые теоретические исследования.

Литература по разработке алгоритмов для решения ЗМТ на основе генетического алгоритма довольно скудна в отличие от его применения для решения ЗК или от более сложных вариантов ЗМТ, где учитываются временные окна или есть ограничения предшествования. Отставание по сравнению с ЗК объясняется тем, что ЗК\_ это хорошо изученная каноническая комбинаторная задача, хорошо подходящая для исследования и пробного применения новых идей вообще и генетического алгоритма в частности, а сложные ограничения тяжело обрабатывать общими алгоритмами.

Генетический алгоритм хорошо подошёл для случаев со сложными ограничениями, т. к. он требует относительно небольшое количество информации о природе самой задачи. В настоящий момент можно говорить, что генетический алгоритм имеет хороший потенциал для создания быстрых методов поиска решений ЗМТ в сочетании с обработкой сложных видов ограничений.

Это подтверждают существующие предложенные методы применения генетического алгоритма для решения ЗМТ в окнах времени. Выполнена работа в области решения ЗМТ с ограничениями по грузоподъёмности, где уделялось особое внимание влиянию различных параметров на качество получаемых результатов.

В ЗМТ решение содержит несколько маршрутов (в отличии от ЗК), поэтому представления цепочек для генетического алгоритма расширено и депо включается в них многократно. Каждое включение депо служит разделителем между двумя соседними маршрутами.

Применяемый классический оператор скрещивания, называемый PMX и оператор мутации RAR были адаптированы для использования в новыхусловиях. На каждой итерации они применяются до тех пор, пока не будет получено требуемое количество решений, удовлетворяющих ограничениям, а остальные потомки отбрасываются. Автор применял специальный оператор локального спуска (local descent operator), основанный на четырёх различных типов изменения порядка выполняемых шагов, и вызывал его только для наилучших решений в текущей популяции. В работе показано, что применение оператора локального спуска даёт заметное улучшение производительности алгоритма. В целом, наилучшие решения, получаемые при помощи генетического алгоритма, моделируемого отжига и поиска с исключениями, сравнимы по качеству. Генетический алгоритм требует несколько больше процессорного времени для работы, чем два других подхода. Пока не приведено сравнительных результатов качества работы генетического алгоритма относительно

алгоритма на основе муравьиных колоний и нейронных сетей.

Интересный вариант генетического алгоритма приведён в [4]. Он может быть использован в задачах с ограничениями на окна времени. Интересная особенность этого алгоритма в том, что в нём используется классический подход с разрезанием общего маршрута, полученным после решения ЗК на общем множестве вершин (см. разд. 1.5.5), что позволяет использовать традиционную схему представления решения без разделителей в виде многократного включения депо. Для разделения на маршруты каждого экипажа применяется процедура заметания, начиная работу с первой вершины, находящейся на первом месте в генетической цепочке. Переход к следующему маршруту происходит, когда превышается ограничение на длину маршрута или грузоподъёмности. Таким образом генетическая цепочка может быть всегда преобразована в допустимое решение ЗМТ.

Реализация алгоритма, предложенная в [5], использует оператор скрещивания OX и оператор мутации, который основан на обмене местами двух случайно выбранных вершин. Для повышения качества работы этот оператор в дальнейшем был заменён на использование алгоритма 2-опт, запускаемый для каждой вершины с вероятностью **0**,**15**.

В [6] предложена схема кодирования, основанная на случайных ключах. В такой схеме каждый элемент задачи связывается со случайно сгенерированным ключом. Преобразование таких ключей в решение задачи можно выполнить при помощи операции сортировки. Непосредственно в ЗМТ каждая вершина-клиент связывается со случайным целым числом, обозначающим транспортное средство, которое будет обслуживать эту вершину, плюс число с плавающей точкой в интервале **(0**, **1)**. При сортировке этих чисел можно получить последовательности вершин для каждого маршрута. Подобное применение случайных чисел для решения ЗМТ приводится скорее для иллюстрации такой возможности, но оно не исследовалось достаточно детально.

1.3 Выводы

Сделаны обзоры ключевых алгоритмов, которые позволяют решить ЗМТ. Так как конечным результатом данной работы должно быть расписание движения транспортных средств для ФГУП “Почта России”, то необходимо подобрать алгоритм лучше всего удовлетворяющий данной задаче. Рабочий день водителя составляет 720 минут, мы имеем первое ограничение по длине маршрута. Нам необходимо подобрать алгоримт, который позволит нам построить маршруты удовлетворяющие следующим условиям:

1. Возможно установить ограничения по длине маршрута.

2. Маршрут является оптимальным

3. Решение получается за приемлимое время.

В результате сравнительного анализа всех указанных выше алгоритмов было принято решения использовать комбинацию из алгоритмов Свира и метода ветвей и границ для решения данной задачи. Метод ветвей и границ позволяет найти математически обоснованный глобальный оптимум, а алгоритм Свира позволяет разбить имеющиеся в задачи точки на сектора по максимальной длине маршрута.

Часть 2. Решение задачи.

Исходя из условий задачи мы имеем точки, которые необходимо посетить транспортному средству. Вот список точек, которые необходимо посетить в течении 720 минут.

|  |
| --- |
| ММПО |
| Москва EMS СЦ 130214 |
| МПКО-Восток |
| МРАСЦ |
| СЦ Мытищи |
| АОПП Внуково |
| АОПП Домодедово |
| МПКО-Юг |
| ПЖДП при Павелецком вокзале |
| АОПП Шереметьево |
| АОПП Шереметьево (ММПО) |
| ММПО |
| МПКО-Север |
| ПЖДП при Казанском вокзале |
| ПЖДП при Казанском вокзале (Цех МПО) |
| ПЖДП при Ярославском вокзале |
| Рем.цех |

Первой группой, которая получилась на основе алгоритма Свира получилась следующая группа точек:

|  |
| --- |
| 1. АОПП Шереметьево |
| 2. АОПП Шереметьево (ММПО) |
| 3. ММПО |
| 4. МПКО-Север |
| 5. ПЖДП при Казанском вокзале |
| 6. ПЖДП при Казанском вокзале (Цех МПО) |
| 7. ПЖДП при Ярославском вокзале |
| 8. Рем.цех |

Обозначим ее группой “север”.

Исходная матрица времени достижения точек выглядит следующим образом:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 | - | 13,92 | 126,13 | 49,69 | 126,43 | 114,33 | 132,22 | 109,18 |
| 2 | 40,00 | - | 155,18 | 45,00 | 110,00 | 103,09 | 117,14 | 120,00 |
| 3 | 134,44 | 134,88 | - | 51,90 | 53,17 | 57,09 | 54,00 | 81,36 |
| 4 | 50,86 | 45,00 | 54,51 | - | 42,76 | 10000 | 51,43 | 54,00 |
| 5 | 118,13 | 115,00 | 56,61 | 44,54 | - | 10000 | 20,30 | 40,52 |
| 6 | 118,40 | 117,29 | 59,60 | 10000 | 26,25 | - | 15,94 | 30,00 |
| 7 | 93,57 | 99,00 | 67,71 | 45,00 | 22,99 | 14,17 | - | 37,47 |
| 8 | 108,79 | 120,00 | 70,00 | 66,00 | 42,53 | 30,00 | 40,47 | - |

Несуществующие ребра графа были установлены как 10000 минут. Алгоритм воспринимает эти ребра как неоптимальные и отбрасывает в ходе решения задачи.

Возьмем в качестве произвольного маршрута:

X0 = (1,2);(2,3);(3,4);(4,5);(5,6);(6,7);(7,8);(8,1)

Тогда F(X0) = 13.92 + 155.18 + 51.90 + 42.76 + 10000.00 + 15.94 + 37.47 + 108.79 = 10425.96

Для определения нижней границы множества воспользуемся операцией редукции или приведения матрицы по строкам, для чего необходимо в каждой строке матрицы D найти минимальный элемент.

di = min(j) dij

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i j | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | di |
| 1 | M | 13.92 | 126.1 | 49.69 | 126.4 | 114.3 | 132.2 | 109.1 | 13.92 |
| 2 | 40 | M | 155.1 | 45 | 110 | 103.0 | 117.1 | 120 | 40 |
| 3 | 134.4 | 134.8 | M | 51.9 | 53.17 | 57.09 | 54 | 81.36 | 51.9 |
| 4 | 50.86 | 45 | 54.51 | M | 42.76 | 10000 | 51.43 | 54 | 42.76 |
| 5 | 118.1 | 115 | 56.61 | 44.54 | M | 10000 | 20.3 | 40.52 | 20.3 |
| 6 | 118.4 | 117.2 | 59.6 | 10000 | 26.25 | M | 15.94 | 30 | 15.94 |
| 7 | 93.57 | 99 | 67.71 | 45 | 22.99 | 14.17 | M | 37.47 | 14.17 |
| 8 | 108.79 | 120 | 70 | 66 | 42.53 | 30 | 40.47 | M | 30 |

Затем вычитаем di из элементов рассматриваемой строки. В связи с этим во вновь полученной матрице в каждой строке будет как минимум один ноль.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i j | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 | M | 0 | 112.21 | 35.77 | 112.51 | 100.41 | 118.3 | 95.26 |
| 2 | 0 | M | 115.18 | 5 | 70 | 63.09 | 77.14 | 80 |
| 3 | 82.54 | 82.98 | M | 0 | 1.27 | 5.19 | 2.1 | 29.46 |
| 4 | 8.1 | 2.24 | 11.75 | M | 0 | 9957.24 | 8.67 | 11.24 |
| 5 | 97.83 | 94.7 | 36.31 | 24.24 | M | 9979.7 | 0 | 20.22 |
| 6 | 102.46 | 101.35 | 43.66 | 9984.06 | 10.31 | M | 0 | 14.06 |
| 7 | 79.4 | 84.83 | 53.54 | 30.83 | 8.82 | 0 | M | 23.3 |
| 8 | 78.79 | 90 | 40 | 36 | 12.53 | 0 | 10.47 | M |

Такую же операцию редукции проводим по столбцам, для чего в каждом столбце находим минимальный элемент:

dj = min(i) dij

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i j | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 | M | 0 | 112.21 | 35.77 | 112.51 | 100.41 | 118.3 | 95.26 |
| 2 | 0 | M | 115.18 | 5 | 70 | 63.09 | 77.14 | 80 |
| 3 | 82.54 | 82.98 | M | 0 | 1.27 | 5.19 | 2.1 | 29.46 |
| 4 | 8.1 | 2.24 | 11.75 | M | 0 | 9957.24 | 8.67 | 11.24 |
| 5 | 97.83 | 94.7 | 36.31 | 24.24 | M | 9979.7 | 0 | 20.22 |
| 6 | 102.46 | 101.35 | 43.66 | 9984.06 | 10.31 | M | 0 | 14.06 |
| 7 | 79.4 | 84.83 | 53.54 | 30.83 | 8.82 | 0 | M | 23.3 |
| 8 | 78.79 | 90 | 40 | 36 | 12.53 | 0 | 10.47 | M |
| dj | 0 | 0 | 11.75 | 0 | 0 | 0 | 0 | 11.24 |

После вычитания минимальных элементов получаем полностью редуцированную матрицу, где величины di и dj называются константами приведения.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i j | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 | M | 0 | 100.46 | 35.77 | 112.51 | 100.41 | 118.3 | 84.02 |
| 2 | 0 | M | 103.43 | 5 | 70 | 63.09 | 77.14 | 68.76 |
| 3 | 82.54 | 82.98 | M | 0 | 1.27 | 5.19 | 2.1 | 18.22 |
| 4 | 8.1 | 2.24 | 0 | M | 0 | 9957.24 | 8.67 | 0 |
| 5 | 97.83 | 94.7 | 24.56 | 24.24 | M | 9979.7 | 0 | 8.98 |
| 6 | 102.46 | 101.35 | 31.91 | 9984.06 | 10.31 | M | 0 | 2.82 |
| 7 | 79.4 | 84.83 | 41.79 | 30.83 | 8.82 | 0 | M | 12.06 |
| 8 | 78.79 | 90 | 28.25 | 36 | 12.53 | 0 | 10.47 | M |

Сумма констант приведения определяет нижнюю границу H:

H = ∑di + ∑dj

H=13.92+40.00+51.90+42.76+20.30+15.94+14.17+30.00+0+0+11.75+0+0+0+0+11.24 = 251.98

Элементы матрицы dij соответствуют расстоянию от пункта i до пункта j.

Поскольку в матрице n точек, то D является матрицей nxn с неотрицательными элементами dij ≥ 0

Каждый допустимый маршрут представляет собой цикл, по которому ТС посещает каждую точку только один раз и возвращается в исходную точку.

Длина маршрута определяется выражением:

F(Mk) = ∑dij

Причем каждая строка и столбец входят в маршрут только один раз с элементом dij.

Шаг №1.

Определяем ребро ветвления и разобьем все множество маршрутов относительно этого ребра на два подмножества (i,j) и (i\*,j\*).

С этой целью для всех клеток матрицы с нулевыми элементами заменяем поочередно нули на М(бесконечность) и определяем для них сумму образовавшихся констант приведения, они приведены в скобках.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i j | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | di |
| 1 | M | 0(38.0) | 100.46 | 35.77 | 112.51 | 100.41 | 118.3 | 84.02 | 35.77 |
| 2 | 0(13.1) | M | 103.43 | 5 | 70 | 63.09 | 77.14 | 68.76 | 5 |
| 3 | 82.54 | 82.98 | M | 0(6.27) | 1.27 | 5.19 | 2.1 | 18.22 | 1.27 |
| 4 | 8.1 | 2.24 | 0(24.5) | M | 0(1.27) | 9957.24 | 8.67 | 0(2.82) | 0 |
| 5 | 97.83 | 94.7 | 24.56 | 24.24 | M | 9979.7 | 0(8.98) | 8.98 | 8.98 |
| 6 | 102.46 | 101.35 | 31.91 | 9984.6 | 10.31 | M | 0(2.82) | 2.82 | 2.82 |
| 7 | 79.4 | 84.83 | 41.79 | 30.83 | 8.82 | 0(8.82) | M | 12.06 | 8.82 |
| 8 | 78.79 | 90 | 28.25 | 36 | 12.53 | 0(10.4) | 10.47 | M | 10.47 |
| dj | 8.1 | 2.24 | 24.56 | 5 | 1.27 | 0 | 0 | 2.82 | 0 |

d(1,2) = 35.77 + 2.24 = 38.01;

d(2,1) = 5 + 8.1 = 13.1;

d(3,4) = 1.27 + 5 = 6.27;

d(4,3) = 0 + 24.56 = 24.56;

d(4,5) = 0 + 1.27 = 1.27;

d(4,8) = 0 + 2.82 = 2.82;

d(5,7) = 8.98 + 0 = 8.98;

d(6,7) = 2.82 + 0 = 2.82;

d(7,6) = 8.82 + 0 = 8.82;

d(8,6) = 10.47 + 0 = 10.47;

Наибольшая сумма констант приведения равна (35.77 + 2.24) = 38.01 для ребра (1,2), следовательно, множество разбивается на два подмножества (1,2) и (1\*,2\*).

Исключение ребра (1,2) проводим путем замены элемента d12 = 0 на M, после чего осуществляем очередное приведение матрицы расстояний для образовавшегося подмножества (1\*,2\*), в результате получим редуцированную матрицу.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i j | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | di |
| 1 | M | M | 100.4 | 35.77 | 112.5 | 100.4 | 118.3 | 84.02 | 35.77 |
| 2 | 0 | M | 103.4 | 5 | 70 | 63.09 | 77.14 | 68.76 | 0 |
| 3 | 82.54 | 82.98 | M | 0 | 1.27 | 5.19 | 2.1 | 18.22 | 0 |
| 4 | 8.1 | 2.24 | 0 | M | 0 | 9957 | 8.67 | 0 | 0 |
| 5 | 97.83 | 94.7 | 24.56 | 24.24 | M | 9979 | 0 | 8.98 | 0 |
| 6 | 102.4 | 101.3 | 31.91 | 9984 | 10.31 | M | 0 | 2.82 | 0 |
| 7 | 79.4 | 84.83 | 41.79 | 30.83 | 8.82 | 0 | M | 12.06 | 0 |
| 8 | 78.79 | 90 | 28.25 | 36 | 12.53 | 0 | 10.47 | M | 0 |
| dj | 0 | 2.24 | 0 | 0 | 0 | 0 | 0 | 0 | 38.01 |

Нижняя граница гамильтоновых циклов этого подмножества:

H(1\*,2\*) = 251.98 + 38.01 = 289.99

Включение ребра (1,2) проводится путем исключения всех элементов 1-ой строки и 2-го столбца, в которой элемент d21 заменяем на М, для исключения образования негамильтонова цикла.

В результате получим другую сокращенную матрицу (7 x 7), которая подлежит операции приведения.

После операции приведения сокращенная матрица будет иметь вид:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i j | 1 | 3 | 4 | 5 | 6 | 7 | 8 | di |
| 2 | M | 103.43 | 5 | 70 | 63.09 | 77.14 | 68.76 | 5 |
| 3 | 82.54 | M | 0 | 1.27 | 5.19 | 2.1 | 18.22 | 0 |
| 4 | 8.1 | 0 | M | 0 | 9957.24 | 8.67 | 0 | 0 |
| 5 | 97.83 | 24.56 | 24.24 | M | 9979.7 | 0 | 8.98 | 0 |
| 6 | 102.46 | 31.91 | 9984.06 | 10.31 | M | 0 | 2.82 | 0 |
| 7 | 79.4 | 41.79 | 30.83 | 8.82 | 0 | M | 12.06 | 0 |
| 8 | 78.79 | 28.25 | 36 | 12.53 | 0 | 10.47 | M | 0 |
| dj | 8.1 | 0 | 0 | 0 | 0 | 0 | 0 | 13.1 |

Сумма констант приведения сокращенной матрицы:

∑di + ∑dj = 13.1

Нижняя граница подмножества (1,2) равна:

H(1,2) = 251.98 + 13.1 = 265.08 ≤ 289.99

Поскольку нижняя граница этого подмножества (1,2) меньше, чем подмножества (1\*,2\*), то ребро (1,2) включаем в маршрут с новой границей H = 265.08

Шаг №2.

Определяем ребро ветвления и разобьем все множество маршрутов относительно этого ребра на два подмножества (i,j) и (i\*,j\*).

С этой целью для всех клеток матрицы с нулевыми элементами заменяем поочередно нули на М(бесконечность) и определяем для них сумму образовавшихся констант приведения, они приведены в скобках.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i j | 1 | 3 | 4 | 5 | 6 | 7 | 8 | di |
| 2 | M | 98.43 | 0(58.09) | 65 | 58.09 | 72.14 | 63.76 | 58.09 |
| 3 | 74.44 | M | 0(1.27) | 1.27 | 5.19 | 2.1 | 18.22 | 1.27 |
| 4 | 0(70.69) | 0(24.56) | M | 0(1.27) | 9957.24 | 8.67 | 0(2.82) | 0 |
| 5 | 89.73 | 24.56 | 24.24 | M | 9979.7 | 0(8.98) | 8.98 | 8.98 |
| 6 | 94.36 | 31.91 | 9984.06 | 10.31 | M | 0(2.82) | 2.82 | 2.82 |
| 7 | 71.3 | 41.79 | 30.83 | 8.82 | 0(8.82) | M | 12.06 | 8.82 |
| 8 | 70.69 | 28.25 | 36 | 12.53 | 0(10.47) | 10.47 | M | 10.47 |
| dj | 70.69 | 24.56 | 0 | 1.27 | 0 | 0 | 2.82 | 0 |

d(2,4) = 58.09 + 0 = 58.09; d(3,4) = 1.27 + 0 = 1.27; d(4,1) = 0 + 70.69 = 70.69; d(4,3) = 0 + 24.56 = 24.56; d(4,5) = 0 + 1.27 = 1.27; d(4,8) = 0 + 2.82 = 2.82; d(5,7) = 8.98 + 0 = 8.98; d(6,7) = 2.82 + 0 = 2.82; d(7,6) = 8.82 + 0 = 8.82; d(8,6) = 10.47 + 0 = 10.47;

Наибольшая сумма констант приведения равна (0 + 70.69) = 70.69 для ребра (4,1), следовательно, множество разбивается на два подмножества (4,1) и (4\*,1\*).

Исключение ребра (4,1) проводим путем замены элемента d41 = 0 на M, после чего осуществляем очередное приведение матрицы расстояний для образовавшегося подмножества (4\*,1\*), в результате получим редуцированную матрицу.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i j | 1 | 3 | 4 | 5 | 6 | 7 | 8 | di |
| 2 | M | 98.43 | 0 | 65 | 58.09 | 72.14 | 63.76 | 0 |
| 3 | 74.44 | M | 0 | 1.27 | 5.19 | 2.1 | 18.22 | 0 |
| 4 | M | 0 | M | 0 | 9957.24 | 8.67 | 0 | 0 |
| 5 | 89.73 | 24.56 | 24.24 | M | 9979.7 | 0 | 8.98 | 0 |
| 6 | 94.36 | 31.91 | 9984.06 | 10.31 | M | 0 | 2.82 | 0 |
| 7 | 71.3 | 41.79 | 30.83 | 8.82 | 0 | M | 12.06 | 0 |
| 8 | 70.69 | 28.25 | 36 | 12.53 | 0 | 10.47 | M | 0 |
| dj | 70.69 | 0 | 0 | 0 | 0 | 0 | 0 | 70.69 |

Нижняя граница гамильтоновых циклов этого подмножества:

H(4\*,1\*) = 265.08 + 70.69 = 335.77

Включение ребра (4,1) проводится путем исключения всех элементов 4-ой строки и 1-го столбца, в которой элемент d14 заменяем на М, для исключения образования негамильтонова цикла.

В результате получим другую сокращенную матрицу (6 x 6), которая подлежит операции приведения.

После операции приведения сокращенная матрица будет иметь вид:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| i j | 3 | 4 | 5 | 6 | 7 | 8 | di |
| 2 | 98.43 | 0 | 65 | 58.09 | 72.14 | 63.76 | 0 |
| 3 | M | 0 | 1.27 | 5.19 | 2.1 | 18.22 | 0 |
| 5 | 24.56 | 24.24 | M | 9979.7 | 0 | 8.98 | 0 |
| 6 | 31.91 | 9984.06 | 10.31 | M | 0 | 2.82 | 0 |
| 7 | 41.79 | 30.83 | 8.82 | 0 | M | 12.06 | 0 |
| 8 | 28.25 | 36 | 12.53 | 0 | 10.47 | M | 0 |
| dj | 24.56 | 0 | 1.27 | 0 | 0 | 2.82 | 28.65 |

Сумма констант приведения сокращенной матрицы:

∑di + ∑dj = 28.65

Нижняя граница подмножества (4,1) равна:

H(4,1) = 265.08 + 28.65 = 293.73 ≤ 335.77

Чтобы исключить подциклы, запретим следующие переходы: (2,4),

Поскольку нижняя граница этого подмножества (4,1) меньше, чем подмножества (4\*,1\*), то ребро (4,1) включаем в маршрут с новой границей H = 293.73

Шаг №3.

Определяем ребро ветвления и разобьем все множество маршрутов относительно этого ребра на два подмножества (i,j) и (i\*,j\*).

С этой целью для всех клеток матрицы с нулевыми элементами заменяем поочередно нули на М(бесконечность) и определяем для них сумму образовавшихся констант приведения, они приведены в скобках.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| i j | 3 | 4 | 5 | 6 | 7 | 8 | di |
| 2 | 15.78 | M | 5.64 | 0(2.85) | 14.05 | 2.85 | 2.85 |
| 3 | M | 0(24.24) | 0(5.64) | 5.19 | 2.1 | 15.4 | 0 |
| 5 | 0(3.69) | 24.24 | M | 9979.7 | 0(0) | 6.16 | 0 |
| 6 | 7.35 | 9984.06 | 9.04 | M | 0(0) | 0(2.85) | 0 |
| 7 | 17.23 | 30.83 | 7.55 | 0(7.55) | M | 9.24 | 7.55 |
| 8 | 3.69 | 36 | 11.26 | 0(3.69) | 10.47 | M | 3.69 |
| dj | 3.69 | 24.24 | 5.64 | 0 | 0 | 2.85 | 0 |

d(2,6) = 2.85 + 0 = 2.85;

d(3,4) = 0 + 24.24 = 24.24;

d(3,5) = 0 + 5.64 = 5.64;

d(5,3) = 0 + 3.69 = 3.69;

d(5,7) = 0 + 0 = 0;

d(6,7) = 0 + 0 = 0;

d(6,8) = 0 + 2.85 = 2.85;

d(7,6) = 7.55 + 0 = 7.55;

d(8,6) = 3.69 + 0 = 3.69;

Наибольшая сумма констант приведения равна (0 + 24.24) = 24.24 для ребра (3,4), следовательно, множество разбивается на два подмножества (3,4) и (3\*,4\*).

Исключение ребра (3,4) проводим путем замены элемента d34 = 0 на M, после чего осуществляем очередное приведение матрицы расстояний для образовавшегося подмножества (3\*,4\*), в результате получим редуцированную матрицу.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| i j | 3 | 4 | 5 | 6 | 7 | 8 | di |
| 2 | 15.78 | M | 5.64 | 0 | 14.05 | 2.85 | 0 |
| 3 | M | M | 0 | 5.19 | 2.1 | 15.4 | 0 |
| 5 | 0 | 24.24 | M | 9979.7 | 0 | 6.16 | 0 |
| 6 | 7.35 | 9984.06 | 9.04 | M | 0 | 0 | 0 |
| 7 | 17.23 | 30.83 | 7.55 | 0 | M | 9.24 | 0 |
| 8 | 3.69 | 36 | 11.26 | 0 | 10.47 | M | 0 |
| dj | 0 | 24.24 | 0 | 0 | 0 | 0 | 24.24 |

Нижняя граница гамильтоновых циклов этого подмножества:

H(3\*,4\*) = 293.73 + 24.24 = 317.97

Включение ребра (3,4) проводится путем исключения всех элементов 3-ой строки и 4-го столбца, в которой элемент d43 заменяем на М, для исключения образования негамильтонова цикла.

В результате получим другую сокращенную матрицу (5 x 5), которая подлежит операции приведения.

После операции приведения сокращенная матрица будет иметь вид:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| i j | 3 | 5 | 6 | 7 | 8 | di |
| 2 | 15.78 | 5.64 | 0 | 14.05 | 2.85 | 0 |
| 5 | 0 | M | 9979.7 | 0 | 6.16 | 0 |
| 6 | 7.35 | 9.04 | M | 0 | 0 | 0 |
| 7 | 17.23 | 7.55 | 0 | M | 9.24 | 0 |
| 8 | 3.69 | 11.26 | 0 | 10.47 | M | 0 |
| dj | 0 | 5.64 | 0 | 0 | 0 | 5.64 |

Сумма констант приведения сокращенной матрицы:

∑di + ∑dj = 5.64

Нижняя граница подмножества (3,4) равна:

H(3,4) = 293.73 + 5.64 = 299.37 ≤ 317.97

Чтобы исключить подциклы, запретим следующие переходы: (2,4), (2,3),

Поскольку нижняя граница этого подмножества (3,4) меньше, чем подмножества (3\*,4\*), то ребро (3,4) включаем в маршрут с новой границей H = 299.37

Шаг №4.

Определяем ребро ветвления и разобьем все множество маршрутов относительно этого ребра на два подмножества (i,j) и (i\*,j\*).

С этой целью для всех клеток матрицы с нулевыми элементами заменяем поочередно нули на М(бесконечность) и определяем для них сумму образовавшихся констант приведения, они приведены в скобках.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| i j | 3 | 5 | 6 | 7 | 8 | di |
| 2 | M | 0(1.91) | 0(0) | 14.05 | 2.85 | 0 |
| 5 | 0(3.69) | M | 9979.7 | 0(0) | 6.16 | 0 |
| 6 | 7.35 | 3.4 | M | 0(0) | 0(2.85) | 0 |
| 7 | 17.23 | 1.91 | 0(1.91) | M | 9.24 | 1.91 |
| 8 | 3.69 | 5.62 | 0(3.69) | 10.47 | M | 3.69 |
| dj | 3.69 | 1.91 | 0 | 0 | 2.85 | 0 |

d(2,5) = 0 + 1.91 = 1.91; d(2,6) = 0 + 0 = 0; d(5,3) = 0 + 3.69 = 3.69; d(5,7) = 0 + 0 = 0; d(6,7) = 0 + 0 = 0; d(6,8) = 0 + 2.85 = 2.85; d(7,6) = 1.91 + 0 = 1.91; d(8,6) = 3.69 + 0 = 3.69;

Наибольшая сумма констант приведения равна (0 + 3.69) = 3.69 для ребра (5,3), следовательно, множество разбивается на два подмножества (5,3) и (5\*,3\*).

Исключение ребра (5,3) проводим путем замены элемента d53 = 0 на M, после чего осуществляем очередное приведение матрицы расстояний для образовавшегося подмножества (5\*,3\*), в результате получим редуцированную матрицу.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| i j | 3 | 5 | 6 | 7 | 8 | di |
| 2 | M | 0 | 0 | 14.05 | 2.85 | 0 |
| 5 | M | M | 9979.7 | 0 | 6.16 | 0 |
| 6 | 7.35 | 3.4 | M | 0 | 0 | 0 |
| 7 | 17.23 | 1.91 | 0 | M | 9.24 | 0 |
| 8 | 3.69 | 5.62 | 0 | 10.47 | M | 0 |
| dj | 3.69 | 0 | 0 | 0 | 0 | 3.69 |

Нижняя граница гамильтоновых циклов этого подмножества:

H(5\*,3\*) = 299.37 + 3.69 = 303.06

Включение ребра (5,3) проводится путем исключения всех элементов 5-ой строки и 3-го столбца, в которой элемент d35 заменяем на М, для исключения образования негамильтонова цикла.

В результате получим другую сокращенную матрицу (4 x 4), которая подлежит операции приведения.

После операции приведения сокращенная матрица будет иметь вид:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| i j | 5 | 6 | 7 | 8 | di |
| 2 | 0 | 0 | 14.05 | 2.85 | 0 |
| 6 | 3.4 | M | 0 | 0 | 0 |
| 7 | 1.91 | 0 | M | 9.24 | 0 |
| 8 | 5.62 | 0 | 10.47 | M | 0 |
| dj | 0 | 0 | 0 | 0 | 0 |

Сумма констант приведения сокращенной матрицы:

∑di + ∑dj = 0

Нижняя граница подмножества (5,3) равна:

H(5,3) = 299.37 + 0 = 299.37 ≤ 303.06

Чтобы исключить подциклы, запретим следующие переходы: (2,4), (2,3), (2,5),

Поскольку нижняя граница этого подмножества (5,3) меньше, чем подмножества (5\*,3\*), то ребро (5,3) включаем в маршрут с новой границей H = 299.37

Шаг №5.

Определяем ребро ветвления и разобьем все множество маршрутов относительно этого ребра на два подмножества (i,j) и (i\*,j\*).

С этой целью для всех клеток матрицы с нулевыми элементами заменяем поочередно нули на М(бесконечность) и определяем для них сумму образовавшихся констант приведения, они приведены в скобках.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| i j | 5 | 6 | 7 | 8 | di |
| 2 | M | 0(2.85) | 14.05 | 2.85 | 2.85 |
| 6 | 1.49 | M | 0(10.47) | 0(2.85) | 0 |
| 7 | 0(1.49) | 0(0) | M | 9.24 | 0 |
| 8 | 3.71 | 0(3.71) | 10.47 | M | 3.71 |
| dj | 1.49 | 0 | 10.47 | 2.85 | 0 |

d(2,6) = 2.85 + 0 = 2.85;

d(6,7) = 0 + 10.47 = 10.47;

d(6,8) = 0 + 2.85 = 2.85;

d(7,5) = 0 + 1.49 = 1.49;

d(7,6) = 0 + 0 = 0;

d(8,6) = 3.71 + 0 = 3.71;

Наибольшая сумма констант приведения равна (0 + 10.47) = 10.47 для ребра (6,7), следовательно, множество разбивается на два подмножества (6,7) и (6\*,7\*).

Исключение ребра (6,7) проводим путем замены элемента d67 = 0 на M, после чего осуществляем очередное приведение матрицы расстояний для образовавшегося подмножества (6\*,7\*), в результате получим редуцированную матрицу.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| i j | 5 | 6 | 7 | 8 | di |
| 2 | M | 0 | 14.05 | 2.85 | 0 |
| 6 | 1.49 | M | M | 0 | 0 |
| 7 | 0 | 0 | M | 9.24 | 0 |
| 8 | 3.71 | 0 | 10.47 | M | 0 |
| dj | 0 | 0 | 10.47 | 0 | 10.47 |

Нижняя граница гамильтоновых циклов этого подмножества:

H(6\*,7\*) = 299.37 + 10.47 = 309.84

Включение ребра (6,7) проводится путем исключения всех элементов 6-ой строки и 7-го столбца, в которой элемент d76 заменяем на М, для исключения образования негамильтонова цикла.

В результате получим другую сокращенную матрицу (3 x 3), которая подлежит операции приведения.

После операции приведения сокращенная матрица будет иметь вид:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| i j | 5 | 6 | 8 | di |
| 2 | M | 0 | 2.85 | 0 |
| 7 | 0 | M | 9.24 | 0 |
| 8 | 3.71 | 0 | M | 0 |
| dj | 0 | 0 | 2.85 | 2.85 |

Сумма констант приведения сокращенной матрицы:

∑di + ∑dj = 2.85

Нижняя граница подмножества (6,7) равна:

H(6,7) = 299.37 + 2.85 = 302.22 ≤ 309.84

Чтобы исключить подциклы, запретим следующие переходы: (2,4), (2,3), (2,5),

Поскольку нижняя граница этого подмножества (6,7) меньше, чем подмножества (6\*,7\*), то ребро (6,7) включаем в маршрут с новой границей H = 302.22

Шаг №6.

Определяем ребро ветвления и разобьем все множество маршрутов относительно этого ребра на два подмножества (i,j) и (i\*,j\*).

С этой целью для всех клеток матрицы с нулевыми элементами заменяем поочередно нули на М(бесконечность) и определяем для них сумму образовавшихся констант приведения, они приведены в скобках.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| i j | 5 | 6 | 8 | di |
| 2 | M | 0(0) | 0(6.39) | 0 |
| 7 | 0(10.1) | M | 6.39 | 6.39 |
| 8 | 3.71 | 0(3.71) | M | 3.71 |
| dj | 3.71 | 0 | 6.39 | 0 |

d(2,6) = 0 + 0 = 0; d(2,8) = 0 + 6.39 = 6.39; d(7,5) = 6.39 + 3.71 = 10.1; d(8,6) = 3.71 + 0 = 3.71;

Наибольшая сумма констант приведения равна (6.39 + 3.71) = 10.1 для ребра (7,5), следовательно, множество разбивается на два подмножества (7,5) и (7\*,5\*).

Исключение ребра (7,5) проводим путем замены элемента d75 = 0 на M, после чего осуществляем очередное приведение матрицы расстояний для образовавшегося подмножества (7\*,5\*), в результате получим редуцированную матрицу.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| i j | 5 | 6 | 8 | di |
| 2 | M | 0 | 0 | 0 |
| 7 | M | M | 6.39 | 6.39 |
| 8 | 3.71 | 0 | M | 0 |
| dj | 3.71 | 0 | 0 | 10.1 |

Нижняя граница гамильтоновых циклов этого подмножества:

H(7\*,5\*) = 302.22 + 10.1 = 312.32

Включение ребра (7,5) проводится путем исключения всех элементов 7-ой строки и 5-го столбца, в которой элемент d57 заменяем на М, для исключения образования негамильтонова цикла.

В результате получим другую сокращенную матрицу (2 x 2), которая подлежит операции приведения.

После операции приведения сокращенная матрица будет иметь вид:

|  |  |  |  |
| --- | --- | --- | --- |
| i j | 6 | 8 | di |
| 2 | 0 | 0 | 0 |
| 8 | 0 | M | 0 |
| dj | 0 | 0 | 0 |

Сумма констант приведения сокращенной матрицы:

∑di + ∑dj = 0

Нижняя граница подмножества (7,5) равна:

H(7,5) = 302.22 + 0 = 302.22 ≤ 312.32

Поскольку нижняя граница этого подмножества (7,5) меньше, чем подмножества (7\*,5\*), то ребро (7,5) включаем в маршрут с новой границей H = 302.22

В соответствии с этой матрицей включаем в гамильтонов маршрут ребра (2,8) и (8,6).

В результате по дереву ветвлений гамильтонов цикл образуют ребра:

(1,2), (2,8), (8,6), (6,7), (7,5), (5,3), (3,4), (4,1),

Длина маршрута равна F(Mk) = 359.22

Во вторую группу попали следующие точки:

|  |
| --- |
| 1. АОПП Внуково |
| 2. АОПП Домодедово |
| 3. МПКО-Юг |
| 4. ПЖДП при Павелецком вокзале |

Обозначим их группой «Юг».

Исходная матрица рассотяний между точками:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 |
| 1 | - | 157,81 | 97,50 | 77,00 |
| 2 | 152,00 | - | 67,50 | 125,24 |
| 3 | 96,00 | 52,86 | - | 43,15 |
| 4 | 93,00 | 133,13 | 47,46 | - |

Возьмем в качестве произвольного маршрута:

X0 = (1,2);(2,3);(3,4);(4,1)

Тогда F(X0) = 157.81 + 67.50 + 43.15 + 93.00 = 361.46

Для определения нижней границы множества воспользуемся операцией редукции или приведения матрицы по строкам, для чего необходимо в каждой строке матрицы D найти минимальный элемент.

di = min(j) dij

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| i j | 1 | 2 | 3 | 4 | di |
| 1 | M | 157.81 | 97.5 | 77 | 77 |
| 2 | 152 | M | 67.5 | 125.24 | 67.5 |
| 3 | 96 | 52.86 | M | 43.15 | 43.15 |
| 4 | 93 | 133.13 | 47.46 | M | 47.46 |

Затем вычитаем di из элементов рассматриваемой строки. В связи с этим во вновь полученной матрице в каждой строке будет как минимум один ноль.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| i j | 1 | 2 | 3 | 4 |
| 1 | M | 80.81 | 20.5 | 0 |
| 2 | 84.5 | M | 0 | 57.74 |
| 3 | 52.85 | 9.71 | M | 0 |
| 4 | 45.54 | 85.67 | 0 | M |

Такую же операцию редукции проводим по столбцам, для чего в каждом столбце находим минимальный элемент:

dj = min(i) dij

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| i j | 1 | 2 | 3 | 4 |
| 1 | M | 80.81 | 20.5 | 0 |
| 2 | 84.5 | M | 0 | 57.74 |
| 3 | 52.85 | 9.71 | M | 0 |
| 4 | 45.54 | 85.67 | 0 | M |
| dj | 45.54 | 9.71 | 0 | 0 |

После вычитания минимальных элементов получаем полностью редуцированную матрицу, где величины di и dj называются константами приведения.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| i j | 1 | 2 | 3 | 4 |
| 1 | M | 71.1 | 20.5 | 0 |
| 2 | 38.96 | M | 0 | 57.74 |
| 3 | 7.31 | 0 | M | 0 |
| 4 | 0 | 75.96 | 0 | M |

Сумма констант приведения определяет нижнюю границу H:

H = ∑di + ∑dj

H = 77.00+67.50+43.15+47.46+45.54+9.71+0+0 = 290.36

Элементы матрицы dij соответствуют расстоянию от пункта i до пункта j.

Поскольку в матрице n городов, то D является матрицей nxn с неотрицательными элементами dij ≥ 0

Каждый допустимый маршрут представляет собой цикл, по которому ТС посещает точку только один раз и возвращается в исходную.

Длина маршрута определяется выражением:

F(Mk) = ∑dij

Причем каждая строка и столбец входят в маршрут только один раз с элементом dij.

Шаг №1.

Определяем ребро ветвления и разобьем все множество маршрутов относительно этого ребра на два подмножества (i,j) и (i\*,j\*).

С этой целью для всех клеток матрицы с нулевыми элементами заменяем поочередно нули на М(бесконечность) и определяем для них сумму образовавшихся констант приведения, они приведены в скобках.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| i j | 1 | 2 | 3 | 4 | di |
| 1 | M | 71.1 | 20.5 | 0(20.5) | 20.5 |
| 2 | 38.96 | M | 0(38.96) | 57.74 | 38.96 |
| 3 | 7.31 | 0(71.1) | M | 0(0) | 0 |
| 4 | 0(7.31) | 75.96 | 0(0) | M | 0 |
| dj | 7.31 | 71.1 | 0 | 0 | 0 |

d(1,4) = 20.5 + 0 = 20.5;

d(2,3) = 38.96 + 0 = 38.96;

d(3,2) = 0 + 71.1 = 71.1;

d(3,4) = 0 + 0 = 0;

d(4,1) = 0 + 7.31 = 7.31;

d(4,3) = 0 + 0 = 0;

Наибольшая сумма констант приведения равна (0 + 71.1) = 71.1 для ребра (3,2), следовательно, множество разбивается на два подмножества (3,2) и (3\*,2\*).

Исключение ребра (3,2) проводим путем замены элемента d32 = 0 на M, после чего осуществляем очередное приведение матрицы расстояний для образовавшегося подмножества (3\*,2\*), в результате получим редуцированную матрицу.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| i j | 1 | 2 | 3 | 4 | di |
| 1 | M | 71.1 | 20.5 | 0 | 0 |
| 2 | 38.96 | M | 0 | 57.74 | 0 |
| 3 | 7.31 | M | M | 0 | 0 |
| 4 | 0 | 75.96 | 0 | M | 0 |
| dj | 0 | 71.1 | 0 | 0 | 71.1 |

Нижняя граница гамильтоновых циклов этого подмножества:

H(3\*,2\*) = 290.36 + 71.1 = 361.46

Включение ребра (3,2) проводится путем исключения всех элементов 3-ой строки и 2-го столбца, в которой элемент d23 заменяем на М, для исключения образования негамильтонова цикла.

В результате получим другую сокращенную матрицу (3 x 3), которая подлежит операции приведения.

После операции приведения сокращенная матрица будет иметь вид:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| i j | 1 | 3 | 4 | di |
| 1 | M | 20.5 | 0 | 0 |
| 2 | 38.96 | M | 57.74 | 38.96 |
| 4 | 0 | 0 | M | 0 |
| dj | 0 | 0 | 0 | 38.96 |

Сумма констант приведения сокращенной матрицы:

∑di + ∑dj = 38.96

Нижняя граница подмножества (3,2) равна:

H(3,2) = 290.36 + 38.96 = 329.32 ≤ 361.46

Поскольку нижняя граница этого подмножества (3,2) меньше, чем подмножества (3\*,2\*), то ребро (3,2) включаем в маршрут с новой границей H = 329.32

Шаг №2.

Определяем ребро ветвления и разобьем все множество маршрутов относительно этого ребра на два подмножества (i,j) и (i\*,j\*).

С этой целью для всех клеток матрицы с нулевыми элементами заменяем поочередно нули на М(бесконечность) и определяем для них сумму образовавшихся констант приведения, они приведены в скобках.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| i j | 1 | 3 | 4 | di |
| 1 | M | 20.5 | 0(39.28) | 20.5 |
| 2 | 0(18.78) | M | 18.78 | 18.78 |
| 4 | 0(0) | 0(20.5) | M | 0 |
| dj | 0 | 20.5 | 18.78 | 0 |

d(1,4) = 20.5 + 18.78 = 39.28;

d(2,1) = 18.78 + 0 = 18.78;

d(4,1) = 0 + 0 = 0;

d(4,3) = 0 + 20.5 = 20.5;

Наибольшая сумма констант приведения равна (20.5 + 18.78) = 39.28 для ребра (1,4), следовательно, множество разбивается на два подмножества (1,4) и (1\*,4\*).

Исключение ребра (1,4) проводим путем замены элемента d14 = 0 на M, после чего осуществляем очередное приведение матрицы расстояний для образовавшегося подмножества (1\*,4\*), в результате получим редуцированную матрицу.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| i j | 1 | 3 | 4 | di |
| 1 | M | 20.5 | M | 20.5 |
| 2 | 0 | M | 18.78 | 0 |
| 4 | 0 | 0 | M | 0 |
| dj | 0 | 0 | 18.78 | 39.28 |

Нижняя граница гамильтоновых циклов этого подмножества:

H(1\*,4\*) = 329.32 + 39.28 = 368.6

Включение ребра (1,4) проводится путем исключения всех элементов 1-ой строки и 4-го столбца, в которой элемент d41 заменяем на М, для исключения образования негамильтонова цикла.

В результате получим другую сокращенную матрицу (2 x 2), которая подлежит операции приведения.

После операции приведения сокращенная матрица будет иметь вид:

|  |  |  |  |
| --- | --- | --- | --- |
| i j | 1 | 3 | di |
| 2 | 0 | M | 0 |
| 4 | M | 0 | 0 |
| dj | 0 | 0 | 0 |

Сумма констант приведения сокращенной матрицы:

∑di + ∑dj = 0

Нижняя граница подмножества (1,4) равна:

H(1,4) = 329.32 + 0 = 329.32 ≤ 368.6

Поскольку нижняя граница этого подмножества (1,4) меньше, чем подмножества (1\*,4\*), то ребро (1,4) включаем в маршрут с новой границей H = 329.32

В соответствии с этой матрицей включаем в гамильтонов маршрут ребра (2,1) и (4,3).

В результате по дереву ветвлений гамильтонов цикл образуют ребра:

(3,2), (2,1), (1,4), (4,3),

Длина маршрута равна F(Mk) = 329.32

Аналогично предыдущим двум группам получаем третью группу “Восток”.

|  |
| --- |
| 1. ММПО |
| 2. Москва EMS СЦ 130214 |
| 3. МПКО-Восток |
| 4. МРАСЦ |
| 5. СЦ Мытищи |

Возьмем в качестве произвольного маршрута:

X0 = (1,2);(2,3);(3,4);(4,5);(5,1)

Тогда F(X0) = 138.33 + 128.00 + 167.48 + 210.83 + 100000.00 = 100644.64

Для определения нижней границы множества воспользуемся операцией редукции или приведения матрицы по строкам, для чего необходимо в каждой строке матрицы D найти минимальный элемент.

di = min(j) dij

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| i j | 1 | 2 | 3 | 4 | 5 | di |
| 1 | M | 138.33 | 53.29 | 127.38 | 10000 | 53.29 |
| 2 | 103.13 | M | 128 | 174.69 | 10000 | 103.13 |
| 3 | 55 | 100000 | M | 167.48 | 94 | 55 |
| 4 | 128.18 | 165.79 | 155.16 | M | 210.83 | 128.18 |
| 5 | 100000 | 100000 | 93.33 | 215.64 | M | 93.33 |

Затем вычитаем di из элементов рассматриваемой строки. В связи с этим во вновь полученной матрице в каждой строке будет как минимум один ноль.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| i j | 1 | 2 | 3 | 4 | 5 |
| 1 | M | 85.04 | 0 | 74.09 | 9946.71 |
| 2 | 0 | M | 24.87 | 71.56 | 9896.87 |
| 3 | 0 | 99945 | M | 112.48 | 39 |
| 4 | 0 | 37.61 | 26.98 | M | 82.65 |
| 5 | 99906.67 | 99906.67 | 0 | 122.31 | M |

Такую же операцию редукции проводим по столбцам, для чего в каждом столбце находим минимальный элемент:

dj = min(i) dij

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| i j | 1 | 2 | 3 | 4 | 5 |
| 1 | M | 85.04 | 0 | 74.09 | 9946.71 |
| 2 | 0 | M | 24.87 | 71.56 | 9896.87 |
| 3 | 0 | 99945 | M | 112.48 | 39 |
| 4 | 0 | 37.61 | 26.98 | M | 82.65 |
| 5 | 99906.67 | 99906.67 | 0 | 122.31 | M |
| dj | 0 | 37.61 | 0 | 71.56 | 39 |

После вычитания минимальных элементов получаем полностью редуцированную матрицу, где величины di и dj называются константами приведения.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| i j | 1 | 2 | 3 | 4 | 5 |
| 1 | M | 47.43 | 0 | 2.53 | 9907.71 |
| 2 | 0 | M | 24.87 | 0 | 9857.87 |
| 3 | 0 | 99907.39 | M | 40.92 | 0 |
| 4 | 0 | 0 | 26.98 | M | 43.65 |
| 5 | 99906.67 | 99869.06 | 0 | 50.75 | M |

Сумма констант приведения определяет нижнюю границу H:

H = ∑di + ∑dj

H = 53.29+103.13+55.00+128.18+93.33+0+37.61+0+71.56+39 = 581.1

Элементы матрицы dij соответствуют расстоянию от пункта i до пункта j.

Поскольку в матрице n городов, то D является матрицей nxn с неотрицательными элементами dij ≥ 0

Каждый допустимый маршрут представляет собой цикл, по которому транспорт посещает точку только один раз и возвращается в исходную.

Длина маршрута определяется выражением:

F(Mk) = ∑dij

Причем каждая строка и столбец входят в маршрут только один раз с элементом dij.

Шаг №1.

Определяем ребро ветвления и разобьем все множество маршрутов относительно этого ребра на два подмножества (i,j) и (i\*,j\*).

С этой целью для всех клеток матрицы с нулевыми элементами заменяем поочередно нули на М(бесконечность) и определяем для них сумму образовавшихся констант приведения, они приведены в скобках.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| i j | 1 | 2 | 3 | 4 | 5 | di |
| 1 | M | 47.43 | 0(2.53) | 2.53 | 9907.71 | 2.53 |
| 2 | 0(0) | M | 24.87 | 0(2.53) | 9857.87 | 0 |
| 3 | 0(0) | 99907.39 | M | 40.92 | 0(43.65) | 0 |
| 4 | 0(0) | 0(47.43) | 26.98 | M | 43.65 | 0 |
| 5 | 99906.67 | 99869.06 | 0(50.75) | 50.75 | M | 50.75 |
| dj | 0 | 47.43 | 0 | 2.53 | 43.65 | 0 |

d(1,3) = 2.53 + 0 = 2.53;

d(2,1) = 0 + 0 = 0;

d(2,4) = 0 + 2.53 = 2.53;

d(3,1) = 0 + 0 = 0;

d(3,5) = 0 + 43.65 = 43.65;

d(4,1) = 0 + 0 = 0;

d(4,2) = 0 + 47.43 = 47.43;

d(5,3) = 50.75 + 0 = 50.75;

Наибольшая сумма констант приведения равна (50.75 + 0) = 50.75 для ребра (5,3), следовательно, множество разбивается на два подмножества (5,3) и (5\*,3\*).

Исключение ребра (5,3) проводим путем замены элемента d53 = 0 на M, после чего осуществляем очередное приведение матрицы расстояний для образовавшегося подмножества (5\*,3\*), в результате получим редуцированную матрицу.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| i j | 1 | 2 | 3 | 4 | 5 | di |
| 1 | M | 47.43 | 0 | 2.53 | 9907.71 | 0 |
| 2 | 0 | M | 24.87 | 0 | 9857.87 | 0 |
| 3 | 0 | 99907.39 | M | 40.92 | 0 | 0 |
| 4 | 0 | 0 | 26.98 | M | 43.65 | 0 |
| 5 | 99906.67 | 99869.06 | M | 50.75 | M | 50.75 |
| dj | 0 | 0 | 0 | 0 | 0 | 50.75 |

Нижняя граница гамильтоновых циклов этого подмножества:

H(5\*,3\*) = 581.1 + 50.75 = 631.85

Включение ребра (5,3) проводится путем исключения всех элементов 5-ой строки и 3-го столбца, в которой элемент d35 заменяем на М, для исключения образования негамильтонова цикла.

В результате получим другую сокращенную матрицу (4 x 4), которая подлежит операции приведения.

После операции приведения сокращенная матрица будет иметь вид:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| i j | 1 | 2 | 4 | 5 | di |
| 1 | M | 47.43 | 2.53 | 9907.71 | 2.53 |
| 2 | 0 | M | 0 | 9857.87 | 0 |
| 3 | 0 | 99907.39 | 40.92 | M | 0 |
| 4 | 0 | 0 | M | 43.65 | 0 |
| dj | 0 | 0 | 0 | 43.65 | 46.18 |

Сумма констант приведения сокращенной матрицы:

∑di + ∑dj = 46.18

Нижняя граница подмножества (5,3) равна:

H(5,3) = 581.1 + 46.18 = 627.28 ≤ 631.85

Поскольку нижняя граница этого подмножества (5,3) меньше, чем подмножества (5\*,3\*), то ребро (5,3) включаем в маршрут с новой границей H = 627.28

Шаг №2.

Определяем ребро ветвления и разобьем все множество маршрутов относительно этого ребра на два подмножества (i,j) и (i\*,j\*).

С этой целью для всех клеток матрицы с нулевыми элементами заменяем поочередно нули на М(бесконечность) и определяем для них сумму образовавшихся констант приведения, они приведены в скобках.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| i j | 1 | 2 | 4 | 5 | di |
| 1 | M | 44.9 | 0(44.9) | 9861.53 | 44.9 |
| 2 | 0(0) | M | 0(0) | 9814.22 | 0 |
| 3 | 0(40.92) | 99907.39 | 40.92 | M | 40.92 |
| 4 | 0(0) | 0(44.9) | M | 0(9814.22) | 0 |
| dj | 0 | 44.9 | 0 | 9814.22 | 0 |

d(1,4) = 44.9 + 0 = 44.9; d(2,1) = 0 + 0 = 0; d(2,4) = 0 + 0 = 0; d(3,1) = 40.92 + 0 = 40.92; d(4,1) = 0 + 0 = 0; d(4,2) = 0 + 44.9 = 44.9; d(4,5) = 0 + 9814.22 = 9814.22;

Наибольшая сумма констант приведения равна (0 + 9814.22) = 9814.22 для ребра (4,5), следовательно, множество разбивается на два подмножества (4,5) и (4\*,5\*).

Исключение ребра (4,5) проводим путем замены элемента d45 = 0 на M, после чего осуществляем очередное приведение матрицы расстояний для образовавшегося подмножества (4\*,5\*), в результате получим редуцированную матрицу.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| i j | 1 | 2 | 4 | 5 | di |
| 1 | M | 44.9 | 0 | 9861.53 | 0 |
| 2 | 0 | M | 0 | 9814.22 | 0 |
| 3 | 0 | 99907.39 | 40.92 | M | 0 |
| 4 | 0 | 0 | M | M | 0 |
| dj | 0 | 0 | 0 | 9814.22 | 9814.22 |

Нижняя граница гамильтоновых циклов этого подмножества:

H(4\*,5\*) = 627.28 + 9814.22 = 10441.5

Включение ребра (4,5) проводится путем исключения всех элементов 4-ой строки и 5-го столбца, в которой элемент d54 заменяем на М, для исключения образования негамильтонова цикла.

В результате получим другую сокращенную матрицу (3 x 3), которая подлежит операции приведения.

После операции приведения сокращенная матрица будет иметь вид:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| i j | 1 | 2 | 4 | di |
| 1 | M | 44.9 | 0 | 0 |
| 2 | 0 | M | 0 | 0 |
| 3 | 0 | 99907.39 | 40.92 | 0 |
| dj | 0 | 44.9 | 0 | 44.9 |

Сумма констант приведения сокращенной матрицы:

∑di + ∑dj = 44.9

Нижняя граница подмножества (4,5) равна:

H(4,5) = 627.28 + 44.9 = 672.18 ≤ 10441.5

Чтобы исключить подциклы, запретим следующие переходы: (3,4),

Поскольку нижняя граница этого подмножества (4,5) меньше, чем подмножества (4\*,5\*), то ребро (4,5) включаем в маршрут с новой границей H = 672.18

Шаг №3.

Определяем ребро ветвления и разобьем все множество маршрутов относительно этого ребра на два подмножества (i,j) и (i\*,j\*).

С этой целью для всех клеток матрицы с нулевыми элементами заменяем поочередно нули на М(бесконечность) и определяем для них сумму образовавшихся констант приведения, они приведены в скобках.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| i j | 1 | 2 | 4 | di |
| 1 | M | 0(99862.49) | 0(0) | 0 |
| 2 | 0(0) | M | 0(0) | 0 |
| 3 | 0(99862.49) | 99862.49 | M | 99862.49 |
| dj | 0 | 99862.49 | 0 | 0 |

d(1,2) = 0 + 99862.49 = 99862.49;

d(1,4) = 0 + 0 = 0;

d(2,1) = 0 + 0 = 0;

d(2,4) = 0 + 0 = 0;

d(3,1) = 99862.49 + 0 = 99862.49;

Наибольшая сумма констант приведения равна (0 + 99862.49) = 99862.49 для ребра (1,2), следовательно, множество разбивается на два подмножества (1,2) и (1\*,2\*).

Исключение ребра (1,2) проводим путем замены элемента d12 = 0 на M, после чего осуществляем очередное приведение матрицы расстояний для образовавшегося подмножества (1\*,2\*), в результате получим редуцированную матрицу.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| i j | 1 | 2 | 4 | di |
| 1 | M | M | 0 | 0 |
| 2 | 0 | M | 0 | 0 |
| 3 | 0 | 99862.49 | M | 0 |
| dj | 0 | 99862.49 | 0 | 99862.49 |

Нижняя граница гамильтоновых циклов этого подмножества:

H(1\*,2\*) = 672.18 + 99862.49 = 100534.67

Включение ребра (1,2) проводится путем исключения всех элементов 1-ой строки и 2-го столбца, в которой элемент d21 заменяем на М, для исключения образования негамильтонова цикла.

В результате получим другую сокращенную матрицу (2 x 2), которая подлежит операции приведения.

После операции приведения сокращенная матрица будет иметь вид:

|  |  |  |  |
| --- | --- | --- | --- |
| i j | 1 | 4 | di |
| 2 | M | 0 | 0 |
| 3 | 0 | M | 0 |
| dj | 0 | 0 | 0 |

Сумма констант приведения сокращенной матрицы:

∑di + ∑dj = 0

Нижняя граница подмножества (1,2) равна:

H(1,2) = 672.18 + 0 = 672.18 ≤ 100534.67

Поскольку нижняя граница этого подмножества (1,2) меньше, чем подмножества (1\*,2\*), то ребро (1,2) включаем в маршрут с новой границей H = 672.18

В соответствии с этой матрицей включаем в гамильтонов маршрут ребра (2,4) и (3,1).

В результате по дереву ветвлений гамильтонов цикл образуют ребра:

(5,3), (3,1), (1,2), (2,4), (4,5),

Длина маршрута равна F(Mk) = 672.18

Результат.

В результате мы получили три гамильтоновых цикла, которые являются оптимальным решением конкретной задачи.

(1,2), (2,8), (8,6), (6,7), (7,5), (5,3), (3,4), (4,1)

(3,2), (2,1), (1,4), (4,3)

(5,3), (3,1), (1,2), (2,4), (4,5)

Соответсвенно, вершнувшись к стандартным обозначениям получим следующие маршруты.

1)ШереметьевоШереметьево(ММПО) Рем. Цех ПЖДП при Казанском вокзале Цех МПО при Казанском вокзале ПЖПД при Ярославском вокзале ММПО Север ММПО Шереметьево

2) МПКО Юг Домодедово Внуково ПЖПД при Павелецком вокзале

3) СЦ Мытищи МПКО Восток ММПО Москва EMS МРАСЦ

Список литературы

1. Меламед И. И. Задача коммивояжера. Приближенные алгоритмы /

И. И. Меламед, С. Сергеев, И. Сигал // Автоматика и телемеханика. 1989

2. Пожидаев М. С. Сбалансированная задача маршрутизации транспортных

средств и проблемы практического применения метаэвристическихстрате-

гий / Ю. Л. Костюк, М. С. Пожидаев // Информационные технологии и

математическое моделирование ( ИТТМ \_ 2009): Материалы VIII Всерос.

научн.-практ. конф. с межд. участием (13–14 ноября 2009 г.). \_ Томск:

Изд-во Том. ун-та, 2009. \_ Ч. 2. \_ С. 233–235.

3. Пождаев М. С. Алгоритмы решения задачи маршрутизации транспорта

Томск, 2010.

4. Schmitt L.J. An evaluation of a genetic algorithmic approach to the vehicle

routing problem // Working paper, Department of Information Technology

Management. \_ Christian Brothers University, Memphis, 1995

5. Schmitt L.J. An empirical computational study of genetic algorithms to solve

order based problems: An emphasis on TSP and VRPTC : Ph.D. Dissertation

/ L.J. Schmitt; Fogelman College of Business and Economics. \_ University of

Memphis, 1994

6. Bean J.C. Genetic algorithms and random keys for sequencing and

optimization // ORSA Journal on Computing. \_ 1994. \_ № 6. \_ P. 154–160

7. Меламед И. К задаче нескольких коммивояжеров // Межвуз. сб. Вып. 647.

М,: МИИТ, 1981.

8. Алексеев А. О. Экспериментальная оценка эффективности алгоритмов

решения задачи -комивояжеров / А. О. Алексеев, О. Г. Алексеев, О. А. Ку-

лагин // Экономика и математические методы, 1993 № 3. с. 496–502.

9. Гэри М. Вычислительные машины и труднорешаемые задачи / М. Гэри,

Д. Джонсон // М.: Мир, 1982.