**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ……………………………………………………………………………..3](#_Введение.)

[1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ…………………………………………………….........4](#_ТЕОРЕТИЧЕСКАЯ_ЧАСТЬ.)

[1.1 Общие описание…….. ……….......……………………………………...…........4](#_1.1_Предмет_распознавания)

[1.2 Жадный алгоритм……………………………………………………………….6](#_1.2_Жадный_алгоритм)

[1.3 Деревянный алгоритм………………………………………………………........9](#_1.3_Деревянный_алгоритм)

[2 ПРАКТИЧЕСКАЯ ЧАСТЬ……………………………………………..…………14](#_2._ПРАКТИЧЕСКАЯ_ЧАСТЬ)

[2.1 Постановка задачи………………………………..………...…...…………...](#_2.1_Постановка_задачи)....14

[2.2 Математическая модель задачи коммивояжера……..……………….……...17](#_2.1.1_Математическая_модель)

[2.3 Алгоритм решения……………...………………………………………............18](#_2.3_Алгоритм_решения)

[ЗАКЛЮЧЕНИЕ………………………………………………………………….…….24](#_ЗАКЛЮЧЕНИЕ)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ………………………....……...........](#_Toc231386181)25

# ВВЕДЕНИЕ

В современном мире происходит стремительное развитие науки. Одной из актуальных тем является минимизации задачи коммивояжера. Задача о коммивояжере состоит в том, чтобы объехать заданные города по одному разу в таком порядке, чтобы пройденное расстояние было минимальным.

Целью курсового проекта является изучение основных методов –алгоритмов решения задачи коммивояжера. Решение задачи одним из способов для нахождения минимального пути. Посмотреть эффективность данного алгоритма. Осуществить цель необходимо с помощью программного обеспечения (программирования) или решение задачи в письменном виде до достижения результата. Программа - решение должно быть конечным продуктом давая наглядный результат.

Предметом исследования является информационные процессы. В информационных процессов лежат процессы минимизации, представления, обработки информации.

# ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

## 1.1 Общие описание

Задача коммивояжера является одной из знаменитых задач теории комбинаторики. Она была поставлена в 1934 году, и об неё, как об Великую теорему Ферма обламывали зубы лучшие математики. В своей области (оптимизации дискретных задач) задача коммивояжера служит своеобразным полигоном, на котором испытываются всё новые методы.

Постановка задачи следующая.

Коммивояжер (бродячий торговец) должен выйти из первого города, посетить по разу в неизвестном порядке города 2,1,3..n и вернуться в первый город. Расстояния между городами известны. В каком порядке следует обходить города, чтобы замкнутый путь (тур) коммивояжера был кратчайшим?

Чтобы привести задачу к научному виду, введём некоторые термины. Итак, города перенумерованы числами j∈Т=(1,2,3..n). Тур коммивояжера может быть описан циклической перестановкой t=(j1,j2,..,jn,j1), причём все j1..jn – разные номера; повторяющийся в начале и в конце j1,показывает, что перестановка зациклена. Расстояния между парами вершин Сij образуют матрицу С. Задача состоит в том, чтобы найти такой тур t, чтобы минимизировать функционал



Относительно математизированной формулировки задача коммивояжера уместно сделать два замечания.

Во-первых, в постановке Сij означали расстояния, поэтому они должны быть неотрицательными, т.е. для всех j∈Т:

|  |  |
| --- | --- |
| Сij≥0; Cjj=∞ | (2) |

(последнее равенство означает запрет на петли в туре), симметричными, т.е. для всех i,j:

|  |  |
| --- | --- |
| Сij= Сji. | (3) |

и удовлетворять неравенству треугольника, т.е. для всех:

|  |  |
| --- | --- |
| Сij+ Сjk≥Cik | (4) |

В математической постановке говорится о произвольной матрице. Сделано это потому, что имеется много прикладных задач, которые описываются основной моделью, но всем условиям (2)-(4) не удовлетворяют. Особенно часто нарушается условие (3) (например, если Сij – не расстояние, а плата за проезд: часто туда билет стоит одну цену, а обратно – другую). Поэтому мы будем различать два варианта задача коммивояжера: симметричную задачу, когда условие (3) выполнено, и несимметричную - в противном случае. Условия (2)-(4) по умолчанию мы будем считать выполненными.

Второе замечание касается числа всех возможных туров. В несимметричной задача коммивояжера все туры t=(j1,j2,..,jn,j1) и t’=(j1,jn,..,j2,j1) имеют разную длину и должны учитываться оба. Разных туров очевидно (n-1)!.

Зафиксируем на первом и последнем месте в циклической перестановке номер j1, а оставшиеся n-1 номеров переставим всеми (n-1)! возможными способами. В результате получим все несимметричные туры. Симметричных туров имеется в два раз меньше, т.к. каждый засчитан два раза: как t и как t’.

Можно представить, что С состоит только из единиц и нулей. Тогда С можно интерпретировать, как граф, где ребро (i,j) проведено, если Сij=0 и не проведено, если Сij=1. Тогда, если существует тур длины 0, то он пройдёт по циклу, который включает все вершины по одному разу. Такой цикл называется гамильтоновым циклом. Незамкнутый гамильтонов цикл называется гамильтоновой цепью (гамильтоновым путём).

В терминах теории графов симметричную задача коммивояжера можно сформулировать так:

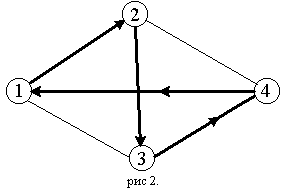
*Дана полная сеть с n вершинами, длина ребра (i,j)=* *Сij. Найти гамильтонов цикл минимальной длины.*

В несимметричной задача коммивояжера вместо «цикл» надо говорить «контур», а вместо «ребра» - «дуги» или «стрелки».

Некоторые прикладные задачи формулируются как задача коммивояжера, но в них нужно минимизировать длину не гамильтонова цикла, а гамильтоновой цепи. Такие задачи называются незамкнутыми. Некоторые модели сводятся к задаче о нескольких коммивояжерах, но мы здесь их рассматривать не будем.

## 1.2 Жадный алгоритм

Жадный алгоритм – алгоритм нахождения наикратчайшего расстояния путём выбора самого короткого, ещё не выбранного ребра, при условии, что оно не образует цикла с уже выбранными рёбрами. «Жадным» этот алгоритм назван потому, что на последних шагах приходится жестоко расплачиваться за жадность.



Посмотрим, как поведет себя при решении задача коммивояжера жадный алгоритм. Здесь он превратится в стратегию «иди в ближайший (в который еще не входил) город». Жадный алгоритм, очевидно, бессилен в этой задаче. Рассмотрим для примера сеть на рис. 2, представляющую уЗадача коммивояжераий ромб. Пусть коммивояжер стартует из города 1. Алгоритм «иди вы ближайший город» выведет его в город 2, затем 3, затем 4; на последнем шаге придется платить за жадность, возвращаясь по длинной диагонали ромба. В результате получится не кратчайший, а длиннейший тур.

В пользу процедуры «иди в ближайший» можно сказать лишь то, что при старте из одного города она не уступит стратегии «иди в дальнейший».

Как видим, жадный алгоритм ошибается. Можно ли доказать, что он ошибается умеренно, что полученный им тур хуже минимального, положим, в 1000 раз? Мы докажем, что этого доказать нельзя, причем не только для жадного логарифма, а для алгоритмов гораздо более мощных. Но сначала нужно договориться, как оценивать погрешность неточных алгоритмов, для определенности, в задаче минимизации. Пусть fB - настоящий минимум, а fA - тот квазиминимум, который получен по алгоритму. Ясно, что fA/ fB≥1, но это – тривиальное утверждение, что может быть погрешность. Чтобы оценить её, нужно зажать отношение оценкой сверху:

|  |  |
| --- | --- |
| fA/fB ≥1+ nε, | ((5) |

где, как обычно в высшей математике, ε≥0, но, против обычая, может быть очень большим. Величина ε и будет служить мерой погрешности. Если алгоритм минимизации будет удовлетворять неравенству (5), мы будем говорить, что он имеет погрешность ε.

Предположим теперь, что имеется алгоритм А решения задача коммивояжера, погрешность которого нужно оценить. Возьмем произвольный граф G (V,E) и по нему составим входную матрицу задача коммивояжера:

|  |  |
| --- | --- |
| С[i,j]={ | 1,если ребро (i,j) принадлежит Е |
| 1+nε в противном случае |

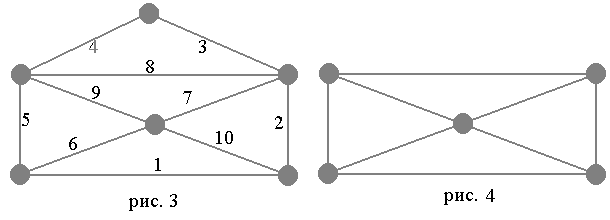
Если в графе G есть гамильтонов цикл, то минимальный тур проходит по этому циклу и fB = n. Если алгоритм А тоже всегда будет находить этот путь, то по результатам алгоритма можно судить, есть ли гамильтонов цикл в произвольном графе. Однако, непереборного алгоритма, который мог бы ответить, есть ли гамильтонов цикл в произвольном графе, до сих пор никому не известно. Таким образом, наш алгоритм А должен иногда ошибаться и включать в тур хотя бы одно ребро длины 1+nε. Но тогда fA≥(n-1)+(1+nε) так что fA/fB=1+nε т.е. превосходит погрешность ε на заданную неравенством (5). О величине ε в нашем рассуждении мы не договаривались, так что ε может быть произвольно велик.

Таким образом доказана следующая теорема.

Либо алгоритм А определяет, существует ли в произвольном графе гамильтонов цикл, либо погрешность А при решении задача коммивояжера может быть произвольно велика.

Это соображение было впервые опубликовано Сани и Гонзалесом в 1980 г. Теорема Сани-Гонзалеса основана на том, что нет никаких ограничений на длину ребер. Теорема не проходит, если расстояния подчиняются неравенству треугольника (4).

Если оно соблюдается, можно предложить несколько алгоритмов с погрешностью 12. Прежде, чем описать такой алгоритм, следует вспомнить старинную головоломку. Можно ли начертить одной линией открытый конверт? Рис.2 показывает, что можно (цифры на отреЗадача коммивояжераах показывают порядок их проведения). Закрытый конверт (рис.3.) одной линией нарисовать нельзя и вот почему. Будем называть линии ребрами, а их перекрестья – вершинами.



Когда через точку проводится линия, то используется два ребра – одно для входа в вершину, одно – для выхода. Если степень вершины нечетна – то в ней линия должна начаться или кончиться. На рис. 3 вершин нечетной степени две: в одной линия начинается, в другой – кончается. Однако на рис. 4 имеется четыре вершины степени три, но у одной линии не может быть четыре конца. Если же нужно прочертить фигуру одной замкнутой линией, то все ее вершины должны иметь четную степень.

Верно и обратное утверждение: если все вершины имеют четную степень, то фигуру можно нарисовать одной незамкнутой линией. Действительно, процесс проведения линии может кончиться, только если линия придет в вершину, откуда уже выхода нет: все ребра, присоединенные к этой вершине (обычно говорят: инцидентные этой вершине), уже прочерчены. Если при этом нарисована вся фигура, то нужное утверждение доказано; если нет, удалим уже нарисованную часть G’. После этого от графа останется одна или несколько связных компонент; пусть G’ – одна из таких компонент. В силу связности исходного графа G, G’ и G’’ имеют хоть одну общую вершину, скажем, v. Если в G’’ удалены какие-то ребра, то по четному числу от каждой вершины. Поэтому G’’ – связный и все его вершины имеют четную степень. Построим цикл в G’’ (может быть, не нарисовав всего G’’) и через v добавим прорисованную часть G’’ к G’. Увеличивая таким образом прорисованную часть G’, мы добьемся того, что G’ охватит весь G.

Эту задачу когда-то решил Эйлер, и замкнутую линию, которая покрывает все ребра графа, теперь называю эйлеровым циклом. По существу была доказана следующая теорема.

Эйлеров цикл в графе существует тогда и только тогда, когда (1) граф связный и (2) все его вершины имеют четные степени*.*

## 1.3 Деревянный алгоритм

Теперь можно обсудить алгоритм решения задача коммивояжера через построение кратчайшего остовного дерева. Для краткости будет называть этот алгоритм деревянным.

Вначале обсудим свойство спрямления. Рассмотрим какую-нибудь цепь, например, на рис.5. Если справедливо неравенство треугольника, то d[1,3]≤d[1,2]+d[2,3] и d[3,5]≤d[3,4]+d[4,5] Сложив эти два неравенства, получим d[1,3]+d[3,5]≤d[1,2]+d[2,3]+d[3,4]+d[4,5]. По неравенству треугольника получим. d[1,5]≤d[1,3]+d[3,5]. Окончательно

d[1,5]≤ d[1,2]+d[2,3]+d[3,4]+d[4,5]

Итак, если справедливо неравенство треугольника, то для каждой цепи верно, что расстояние от начала до конца цепи меньше (или равно) суммарной длины всех ребер цепи. Это обобщение расхожего убеждения, что прямая короче кривой.

Вернемся к задача коммивояжера и опишем решающий ее деревянный алгоритм.

Построим на входной сети задача коммивояжера кратчайшее остовное дерево и удвоим все его ребра. Получим граф G – связный и с вершинами, имеющими только четные степени.

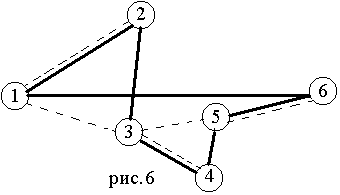
Построим эйлеров цикл G, начиная с вершины 1, цикл задается перечнем вершин.

Просмотрим перечень вершин, начиная с 1, и будем зачеркивать каждую вершину, которая повторяет уже встреченную в последовательности. Останется тур, который и является результатом алгоритма.

Пример 1. Дана полная сеть, показанная на рис.5. Найти тур жадным и деревянным алгоритмами.



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| - | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | - | 6 | 4 | 8 | 7 | 14 |
| 2 | 6 | - | 7 | 11 | 7 | 10 |
| 3 | 4 | 7 | - | 4 | 3 | 10 |
| 4 | 8 | 11 | 4 | - | 5 | 11 |
| 5 | 7 | 7 | 3 | 5 | - | 7 |
| 6 | 14 | 10 | 10 | 11 | 7 | - |
| табл. 1 | | | | | | |



Решение. Жадный алгоритм (иди в ближайший город из города 1) дает тур 1–(4)–3-(3)–5(5)–4–(11)–6–(10)–2–(6)–1, где без скобок показаны номера вершин, а в скобках – длины ребер. Длина тура равна 39, тур показана на рис. 5.

2. Деревянный алгоритм вначале строит остовное дерево, показанное на рис. 6 штриховой линией, затем эйлеров цикл 1-2-1-3-4-3-5-6-5-3-1, затем тур 1-2-3-4-5-6-1 длиной 43, который показан сплошной линией на рис. 6.

Теорема. Погрешность деревянного алгоритма равна 1.

Доказательство. Возьмем минимальный тур длины fB и удалим из него максимальное ребро. Длина получившейся гамильтоновой цепи LHC меньше fB. Но эту же цепь можно рассматривать как остовное дерево, т. к. эта цепь достигает все вершины и не имеет циклов. Длина кратчайшего остовного дерева LMT меньше или равна LHC. Имеем цепочку неравенств

|  |  |
| --- | --- |
| fB>LHC≥LMT | ((6) |

Но удвоенное дерево – оно же эйлеров граф – мы свели к туру посредством спрямлений, следовательно, длина полученного по алгоритму тура удовлетворяет неравенству

|  |  |
| --- | --- |
| 2LMT>fA | ((7) |

Умножая (6) на два и соединяя с (7), получаем цепочку неравенств

|  |  |
| --- | --- |
| 2fB>2LHC≥2LMT≥fA | ((8) |

Т.е. 2fB>fA, т.е. fA/fB>1+ε; ε=1.

Теорема доказана.

Таким образом, мы доказали, что деревянный алгоритм ошибается менее, чем в два раза. Такие алгоритмы уже называют приблизительными, а не просто эвристическими.

Известно еще несколько простых алгоритмов, гарантирующих в худшем случае ε=1. Для того, чтобы найти среди них алгоритм поточнее, зайдем с другого конца и для начала опишем «brute-force enumeration» - «перебор животной силой», как его называют в англоязычной литературе. Понятно, что полный перебор практически применим только в задачах малого размера. Напомним, что ЗАДАЧА КОММИВОЯЖЕРА с n городами требует при полном переборе рассмотрения (n-1)!/2 туров в симметричной задаче и (n-1)! Туров в несимметричной, а факториал, как показано в следующей таблице, растет удручающе быстро:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 5! | 10! | 15! | 20! | 25! | 30! | 35! | 40! | 45! | 50! |
| ~102 | ~106 | ~1012 | ~1018 | ~10125 | ~1032 | ~1040 | ~1047 | ~1056 | ~1064 |

Чтобы проводить полный перебор в ЗАДАЧА КОММИВОЯЖЕРА, нужно научиться (разумеется, без повторений) генерировать все перестановки заданного числа m элементов. Это можно сделать несколькими способами, но самый распространенный (т.е. приложимый для переборных алгоритмов решения других задач) – это перебор в лексикографическом порядке.

Пусть имеется некоторый алфавит и наборы символов алфавита (букв), называемые словами. Буквы в алфавите упорядочены: например, в русском алфавите порядок букв а∝б∝я (символ ∝ читается «предшествует)». Если задан порядок букв, можно упорядочить и слова. Скажем, дано слово u=(u1,u2,..,um) – состоящее из букв u1,u2,..,um - и слово v =(v1,v2,..,vb). Тогда если u1∝v1, то и u∝v, если же u1=v1, то сравнивают вторые буквы и т.д. Этот порядок слов и называется лексикографическим. Поэтому в русских словарях (лексиконах) слово «абажур» стоит раньше слова «абака». Слово «бур» стоит раньше слова «бура», потому что пробел считается предшествующим любой букве алфавита.

Рассмотрим, скажем, перестановки из пяти элементов, обозначенных цифрами 1..5. Лексикографически первой перестановкой является 1-2-3-4-5, второй – 1-2-3-5-4, …, последней – 5-4-3-2-1. Нужно осознать общий алгоритм преобразования любой перестановки в непосредственно следующую.

Правило такое: скажем, дана перестановка 1-3-5-4-2. Нужно двигаться по перестановке справа налево, пока впервые не увидим число, меньшее, чем предыдущее (в примере это 3 после 5). Это число, Pi-1 надо увеличить, поставив вместо него какое-то число из расположенных правее, от Pi до Pn. Число большее, чем Pi-1, несомненно, найдется, так как Pi-1< Pi . Если есть несколько больших чисел, то, очевидно, надо ставить меньшее из них. Пусть это будет Pj,j>i-1. Затем число Pi-1 и все числа от Pi до Pn, не считая Pj нужно упорядочить по возрастанию. В результате получится непосредственно следующая перестановка, в примере – 1-4-2-3-5. Потом получится 1-4-2-5-3 (тот же алгоритм, но упрощенный случай) и т.д.

Нужно понимать, что в задача коммивояжера с n городами не нужны все перестановки из n элементов. Потому что перестановки, скажем, 1-3-5-4-2 и 3-5-4-2-1 (последний элемент соединен с первым) задают один и тот же тур, считанный сперва с города 1, а потом с города 3. Поэтому нужно зафиксировать начальный город 1 и присоединять к нему все перестановки из остальных элементов. Этот перебор даст (n-1)! разных туров, т.е. полный перебор в несимметричной задача коммивояжера (мы по-прежнему будем различать туры 1-3-5-4-2 и 1-2-4-5-3).

# 2 ПРАКТИЧЕСКАЯ ЧАСТЬ

## 2.1 Постановка задачи

Рассмотрим задачу о коммивояжере.

Имеются n городов, расстояния (стоимость проезда, расход горючего на дорогу и т.д.) между которыми известны. Коммивояжер должен пройти все n городов по одному разу, вернувшись в тот город, с которого начал. Требуется найти такой маршрут движения, при котором суммарное пройденное расстояние (стоимость проезда и т.д.) будет минимальным.

Очевидно, что задача коммивояжера – это задача отыскания кратчайшего гамильтонова цикла в полном графе.

Можно предложить следующую простую схему решения задачи коммивояжера: сгенерировать все n! возможных перестановок вершин полного графа, подсчитать для каждой перестановки длину маршрута и выбрать кратчайший. Однако, n! с ростом n растет быстрее, чем любой полином от n, и даже быстрее, чем . Таким образом, решение задачи коммивояжера методом полного перебора оказывается практически неосуществимым, даже при достаточно небольших n.

Решить задачу коммивояжера также можно с помощью алгоритма «Жадного» и «деревянного» алгоритма. Эти методы ускоряют разработку алгоритма по сравнению с методом полного перебора, однако не всегда дают оптимальное решение.

Существует метод решения задачи коммивояжера, который дает оптимальное решение. Этот метод называется методом ветвей и границ. Решение задачи коммивояжера методом ветвей и границ по-другому называют алгоритмом Литтла.

Если считать города вершинами графа, а коммуникации (i,j) – его дугами, то требование нахождения минимального пути, проходящего один и только один раз через каждый город, и возвращения обратно, можно рассматривать как нахождение на графе гамильтонова контура минимальной длины.

Для практической реализации идеи метода ветвей и границ применительно к задаче коммивояжера нужно найти метод определения нижних границ подмножества и разбиения множества гамильтоновых контуров на подмножества (ветвление).

Определение нижних границ базируется на том утверждении, что если ко всем элементам i-й строки или j-го столбца матрицы C прибавить или отнять число , то задача останется эквивалентной прежней, то есть оптимальность маршрута коммивояжера не изменится, а длина любого гамильтонова контура изменится на данную величину .

Опишем алгоритм Литтла для нахождения минимального гамильтонова контура для графа с n вершинами. Граф представляют в виде матрицы его дуг. Если между вершинами Xi и Xj нет дуги, то ставится символ «∞».

Алгоритм Литтла для решения задачи коммивояжера можно сформулировать в виде следующих правил:

1. Находим в каждой строке матрицы  минимальный элемент  и вычитаем его из всех элементов соответствующей строки. Получим матрицу, приведенную по строкам, с элементами

.

2. Если в матрице , приведенной по строкам, окажутся столбцы, не содержащие нуля, то приводим ее по столбцам. Для этого в каждом столбце матрицы  выбираем минимальный элемент ,  и вычитаем его из всех элементов соответствующего столбца. Получим матрицу

,

каждая строка и столбец, которой содержит хотя бы один нуль. Такая матрица называется приведенной по строкам и столбцам.

3. Суммируем элементы и , получим константу приведения

,

которая будет нижней границей множества всех допустимых гамильтоновых контуров, то есть

.

4. Находим степени нулей для приведенной по строкам и столбцам матрицы. Для этого мысленно нули в матице заменяем на знак «∞» и находим сумму минимальных элементов строки и столбца, соответствующих этому нулю. Записываем ее в правом верхнем углу клетки:

.

5. Выбираем дугу , для которой степень нулевого элемента достигает максимального значения

.

6. Разбиваем множество всех гамильтоновых контуров  на два подмножества  и . Подмножество гамильтоновых контуров содержит дугу ,  - ее не содержит. Для получения матрицы контуров , включающих дугу , вычеркиваем в матрице  строку  и столбец . Чтобы не допустить образования негамильтонова контура, заменим симметричный элемент на знак «∞».

7. Приводим матрицу гамильтоновых контуров . Пусть  - константа ее приведения. Тогда нижняя граница множества определится так:

.

8. Находим множество гамильтоновых контуров , не включающих дугу . Исключение дуги  достигается заменой элемента  в матрице  на ∞.

9. Делаем приведение матрицы гамильтоновых контуров . Пусть  - константа ее приведения. Нижняя граница множества определится так:

.

10. Сравниваем нижние границы подмножества гамильтоновых контуров  и . Если , то дальнейшему ветвлению в первую очередь подлежит множество . Если же , то разбиению подлежит множество .

Процесс разбиения множеств на подмножества сопровождается построением дерева ветвлений.

11. Если в результате ветвлений получаем матрицу , то определяем полученный ветвлением гамильтонов контур и его длину.

12. Сравниваем длину гамильтонова контура с нижними границами оборванных ветвей. Если длина контура не превышает их нижних границ, то задача решена. В противном случае развиваем ветви подмножеств с нижней границей, меньшей полученного контура, до тех пор, пока не получим маршрут с меньшей длиной или не убедимся, что такого не существует.

## 2.2 Математическая модель задачи коммивояжера

Задача коммивояжера может быть сформулирована как целочисленная введением булевых переменных , если маршрут включает переезд из города i непосредственно в город j и в противном случае. Тогда можно задать математическую модель задачи, то есть записать целевую функцию и систему ограничений:

 (1)

****

Условия (2) – (4) в совокупности обеспечивают, что каждая переменная  равна или нулю, или единице. При этом ограничения (2), (3) выражают условия, что коммивояжер побывает в каждом городе один раз в него прибыв (ограничение (2)), и один раз из него выехав (ограничение (3)).

Однако этих ограничений не достаточно для постановки задачи коммивояжера, так как они не исключают решения, где вместо простого цикла, проходящего через n вершин, отыскиваются 2 и более отдельных цикла (подцикла), проходящего через меньшее число вершин. Поэтому задача, описанная уравнениями (2) – (4) должна быть дополнена ограничениями, обеспечивающими связность искомого цикла.

Для того, чтобы исключить при постановке задачи все возможные подциклы в систему ограничений задачи включают следующее ограничение:

, где ,  и .

## 2.3 Алгоритм решения

Дана матрица расстояний, представленная в таблице 1. Необходимо с помощью алгоритма Литтла решить задачу коммивояжера.

**Табл.1**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| j  i | **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | **∞** | 7 | 16 | 21 | 2 | 17 |
| **2** | 13 | **∞** | 21 | 15 | 43 | 23 |
| **3** | 25 | 3 | **∞** | 31 | 17 | 9 |
| **4** | 13 | 10 | 27 | **∞** | 33 | 12 |
| **5** | 9 | 2 | 19 | 14 | **∞** | 51 |
| **6** | 42 | 17 | 5 | 9 | 23 | **∞** |

1) Справа к таблице присоединяем столбец *Ui*, в котором записываем минимальные элементы соответствующих строк. Вычитаем элементы *Ui* из соответствующих элементов строки матрицы.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| j  i | **1** | **2** | **3** | **4** | **5** | **6** | ***Ui*** |
| **1** | **∞** | 7 | 16 | 21 | 2 | 17 | 2 |
| **2** | 13 | **∞** | 21 | 15 | 43 | 23 | 13 |
| **3** | 25 | 3 | **∞** | 31 | 17 | 9 | 3 |
| **4** | 13 | 10 | 27 | **∞** | 33 | 12 | 10 |
| **5** | 9 | 2 | 19 | 14 | **∞** | 51 | 2 |
| **6** | 42 | 17 | 5 | 9 | 23 | **∞** | 5 |

2) Внизу полученной матрицы присоединяем строку *Vj*, в которой записываем минимальные элементы столбцов. Вычитаем элементы *Vj* из соответствующих столбцов матрицы.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| j  i | **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | **∞** | 5 | 14 | 19 | 0 | 15 |
| **2** | 0 | **∞** | 8 | 2 | 30 | 10 |
| **3** | 22 | 0 | **∞** | 28 | 14 | 6 |
| **4** | 3 | 0 | 17 | **∞** | 23 | 2 |
| **5** | 7 | 0 | 17 | 12 | **∞** | 49 |
| **6** | 37 | 12 | 0 | 4 | 18 | **∞** |
| **Vj** | 0 | 0 | 0 | 2 | 0 | 2 |

3) В результате вычислений получаем матрицу, приведенную по строкам и столбцам, которая изображена в виде таблицы 2.

**Табл.2**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| j  i | **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | **∞** | 5 | 14 | 17 | 019 | 13 |
| **2** | 03 | **∞** | 8 | 02 | 30 | 8 |
| **3** | 22 | 04 | **∞** | 26 | 14 | 4 |
| **4** | 3 | 00 | 17 | **∞** | 23 | 04 |
| **5** | 7 | 07 | 17 | 10 | **∞** | 47 |
| **6** | 37 | 12 | 08 | 2 | 18 | **∞** |

4) Находим константу приведения:

.

Таким образом, нижней границей множества всех гамильтоновых контуров будет число .

5) Находим степени нулей полностью приведенной матрицы. Для этого как бы заменяем в ней все нули на знак «**∞**» и устанавливаем сумму минимальных элементов соответствующей строки и столбца. Степени нулей записаны в правых верхних углах клеток, для которых .

6) Определяем максимальную степень нуля. Она равна 19 и соответствует клетке (1;5). Таким образом, претендентом на включение в гамильтонов контур является дуга (1;5).

7) Разбиваем множество всех гамильтоновых контуров на два:  и . Матрицу  с дугой (1;5) получаем табл.2 путем вычеркивания строки 1 и столбца 5. Чтобы не допускать образования негамильтонова контура, заменяем элемент (5;1) на знак «**∞**».

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| j  i | **1** | **2** | **3** | **4** | **6** |
| **2** | 0 | **∞** | 8 | 0 | 8 |
| **3** | 22 | 0 | **∞** | 26 | 4 |
| **4** | 3 | 0 | 17 | **∞** | 0 |
| **5** | **∞** | 0 | 17 | 10 | 47 |
| **6** | 37 | 12 | 0 | 2 | **∞** |

8) Матрицу гамильтоновых контуров  получим из таблицы 2 путем замены элемента (1;5) на знак «**∞»**.

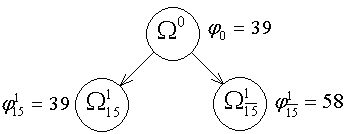
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| j  i | **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | **∞** | 5 | 14 | 17 | **∞** | 13 | 5 |
| **2** | 0 | **∞** | 8 | 0 | 30 | 8 |
| **3** | 22 | 0 | **∞** | 26 | 14 | 4 |
| **4** | 3 | 0 | 17 | **∞** | 23 | 0 |
| **5** | 7 | 0 | 17 | 10 | **∞** | 47 |
| **6** | 37 | 12 | 0 | 2 | 18 | **∞** |
| 14 |

9) Делаем дополнительное приведение матрицы контуров : = 0. Нижняя граница множества  равна .

10) Находим константу приведения для множества контуров :. Следовательно, нижняя граница множества  равна .

11) Сравниваем нижние границы подмножеств  и . Так как , то дальнейшему ветвлению подвергаем множество .

На рис.1 представлено ветвление по дуге (1;5).



**рис.1**

Переходим к ветвлению подмножества . Его приведенная матрица представлена в таблице ниже.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| j  i | **1** | **2** | **3** | **4** | **6** |
| **2** | 03 | **∞** | 8 | 02 | 8 |
| **3** | 22 | 04 | **∞** | 26 | 4 |
| **4** | 3 | 00 | 17 | **∞** | 04 |
| **5** | **∞** | 010 | 17 | 10 | 47 |
| **6** | 37 | 12 | 010 | 2 | **∞** |

12) Узнаем степени нулей матрицы. Претендентами на включение в гамильтонов контур будут несколько дуг (5;2) и (6;3). Для дальнейших расчетов выберем дугу (6;3). Разбиваем множество  на два подмножества: и  (табл. 3 и 4). Чтобы не допустить образования негамильтонова контура, в таблице 3 заменяем элемент (3;6) на знак «**∞**»

**Табл.3**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| j  i | **1** | **2** | **4** | **6** |
| **2** | 0 | **∞** | 0 | 8 |
| **3** | 22 | 0 | 26 | **∞** |
| **4** | 3 | 0 | **∞** | 0 |
| **5** | **∞** | 0 | 10 | 47 |

**Табл.4**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| j  i | **1** | **2** | **3** | **4** | **6** |
| **2** | 0 | **∞** | 8 | 0 | 8 |
| **3** | 22 | 0 | **∞** | 26 | 4 |
| **4** | 3 | 0 | 17 | **∞** | 0 |
| **5** | **∞** | 0 | 17 | 10 | 47 |
| **6** | 37 | 12 | **∞** | 2 | **∞** | 2 |
| 8 |

Определим константы приведения для этих матриц , . Следовательно, , . Так как , то дальнейшему ветвлению подлежит подмножество . Находим степени нулей матрицы.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| j  i | **1** | **2** | **4** | **6** |
| **2** | 03 | **∞** | 02 | 8 |
| **3** | 22 | 022 | 26 | **∞** |
| **4** | 3 | 00 | **∞** | 08 |
| **5** | **∞** | 010 | 10 | 47 |

Претендентом к включению в гамильтонов контур станет дуга (3;2). Разбиваем множество  на два  и .

|  |  |  |  |
| --- | --- | --- | --- |
| j  i | **1** | **4** | **6** |
| **2** | 0 | 0 | 8 |
| **4** | 3 | **∞** | 0 |
| **5** | **∞** | 10 | 47 | 10 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| j  i | **1** | **2** | **4** | **6** |
| **2** | 0 | **∞** | 0 | 8 |
| **3** | 22 | **∞** | 26 | **∞** | 22 |
| **4** | 3 | 0 | **∞** | 0 |
| **5** | **∞** | 0 | 10 | 47 |

О

Очевидно, , . Следовательно, очередному ветвлению нужно подвергнуть подмножество .

Переходим к ветвлению подмножества . Его приведенная матрица представлена в таблице ниже.

|  |  |  |  |
| --- | --- | --- | --- |
| j  i | **1** | **4** | **6** |
| **2** | 03 | 00 | 8 |
| **4** | 3 | **∞** | 011 |
| **5** | **∞** | 037 | 37 |

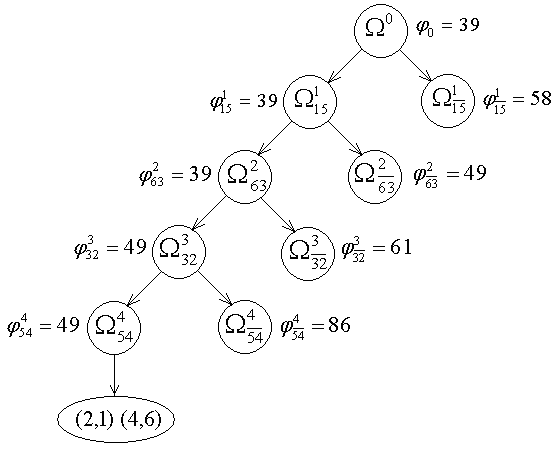
Определяем степени нулей. Претендентом на включение в гамильтонов контур является дуга (5;4). Разбиваем множество  на два подмножества:  и .

|  |  |  |
| --- | --- | --- |
| j  i | **1** | **6** |
| **2** | 0 | 8 |
| **4** | 3 | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
| j  i | **1** | **4** | **6** |
| **2** | 0 | 0 | 8 |
| **4** | 3 | **∞** | 0 |
| **5** | **∞** | **∞** | 37 | 37 |

Находим нижние границы , . Следовательно, очередному ветвлению нужно подвергнуть подмножество . Но его матрица имеет размерность . Поэтому в гамильтонов контур следует включить дуги, соответствующие в матрице  нулевым элементам. Это дуги (2;1) и (4;6).

На рис.2 представлено дерево ветвлений. Определим полученный гамильтонов контур. В него вошли дуги . Длина контура равна . Так как границы оборванных ветвей больше длины контура 1 – 5 – 4 – 6 – 3 – 2 – 1, то этот контура имеет наименьшую длину.



**рис.2**

# ЗАКЛЮЧЕНИЕ

Проведён анализ наиболее рациональных методов решения задачи коммивояжера и определены области их эффективного действия: для малого числа вершин можно использовать точный метод лексического перебора; для большого числа вершин рациональнее применять метод ветвей и границ.

Изучены практические применения задачи коммивояжера. Была Решена задача коммивояжера и найден в ней минимальный путь. Где был показан алгоритм решения достаточно четкий алгоритм решения, что можно реализовать на любом программном языке.

Такая задача актуальна во многих областях, таких как автомобильные, судовые и железнодорожные перевоЗадача коммивояжераи, расчет авиационных линий, конвейерное производство.

# СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Ф. А. Новиков <Дискретная математика для программистов>. - Санкт-Петербург, Питер, 2001, 304 с.,
2. Орлов А.И. <Основы теории принятия решения>. – Учебное пособие. Москва. 2004 г. 249 с.
3. Г.И. Просветов <Математические методы в логистике>. – Москва. Изд-во РДЛ. 2006г.
4. Ловас Л., Пламмер М. <Прикладные задачи теории графов. Теория паросочетаний в математике, физике, химии>. Москва. Изд-во Мир. 1998 г. 658 с.
5. Мудров В. И., <Задача о коммивояжёре>, Москва, 1969 г.