УДК 519.168:519.714.7

ЗАДАЧА КОММИВОЯЖЕРА

Харкевич М.А, студент гр.451003, Заривняк В.С, студент гр.451004

Белорусский государственный университет информатики и радиоэлектроники[1]
г. Минск, Республика Беларусь

Баркова Е.А. – канд. физ.-мат. наук

Аннотация. Главной целью данной научной работы является анализ существующих методов решения задачи коммивояжера и их оптимизация. В данной работе был проанализированы конструктивные алгоритмы “Ближайший сосед”, алгоритм с точностью 2, алгоритм Кристофидеса-Сердюкова и генетические алгоритмы. Был разработан свой алгоритм, объединяющий конструктивные и генетичекие методы решения.

Ключевые слова. Задача коммивояжера, маршрут, остовное дерево, эйлеров цикл, гамильтонов цикл, конструктивный алгоритм, генетический алгоритм, популяция, особи, хромосома, скрещивание, приспособленность, оптимум, селекция, точность, матрица расстояний, ближайший сосед, Краскал, паросочетание, мутация, комбинированный метод, города, применение.

Жизнь – непредсказуемый поток времени и событий: кто-то идет в школу по знакомому с детства пути, кто-то теряется в переходе между линиями метро, кто-то в пробке и пытается сократить свой путь, чтобы успеть на работу. Мы стремимся найти короткий и удобный маршрут, а для логистов, курьеров, транспортных компаний – это ключевая задача работы. Им приходится прокладывать выгодные маршруты, чтобы сэкономить время, топливо и ресурсы, обеспечивая эффективность перевозок и сервиса.

Задача коммивояжера – одна из фундаментальных проблем дискретной математики, суть которой – найти кратчайший маршрут, проходящий лишь один раз через все заданные точки с возвратом в исходную. Данный вид принадлежит к классу NP-сложных, что означает: точное решение полным перебором выполняется для небольшого числа городов, поскольку с увеличением количества точек сложность задачи возрастает намного быстрее.

Для решения задачи коммивояжера существует много видов алгоритмов. Рассмотрим несколько вариантов, их суть и как составить маршрут, используя обозначенный способ. Алгоритмы, формирующие решение последовательно, называются конструктивными. Примером такого подхода является алгоритм «Ближайший сосед». Рассмотрим 4 города: A, B, C, D и их матрицу расстояний bij (рисунок 1).

$$\begin{pmatrix} 0 & 2 & 4 & 3 \\ 2 & 0 & 1 & 5 \\ 4 & 1 & 0 & 2 \\ 3 & 5 & 2 & 0 \end{pmatrix}$$

Рисунок 6 – Матрица расстояний

Матрица b_{ij} должна удовлетворять неравенству треугольника: $b_{ij} \leq b_{ik} + b_{kj}$ для исключения «лишних» повторных вершин. Выбирем произвольный стартовый город A. Дальше ищем ближайший к нему не посещенный город и добавляем к предыдущим, чтобы образовать итоговый путь, то есть из A–B–C–D–A (рисунок 2). Тогда оптимальный маршрут Opt будет равен 10 (2 + 1 + 2 + 5 = 10). У данного алгоритма нет гарантированной заданной точности, оно может быть максимально большим, так как не все маршруты могут быть оптимальными. Точность алгоритма – отношение длины решения алгоритма к оптимальной длине Opt.

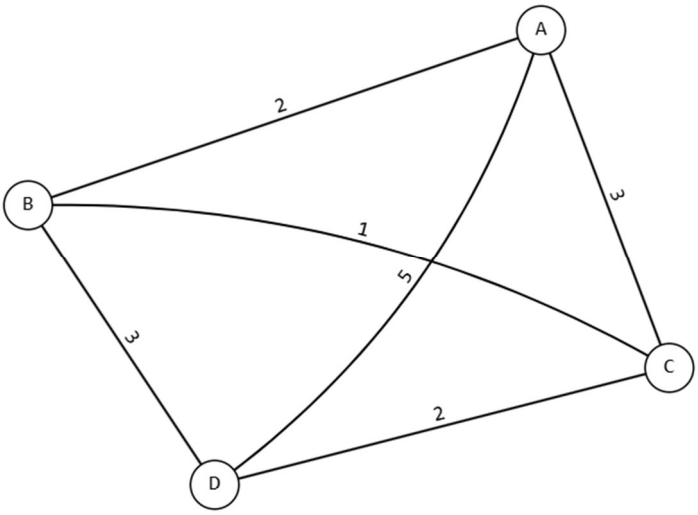

Рисунок *7* – Алгоритм «Ближайший сосед»

Рассмотрим алгоритм с гарантированной точностью r = 2. Пусть даны 4 вершины A, B, C, D и граф (рисунок 3).

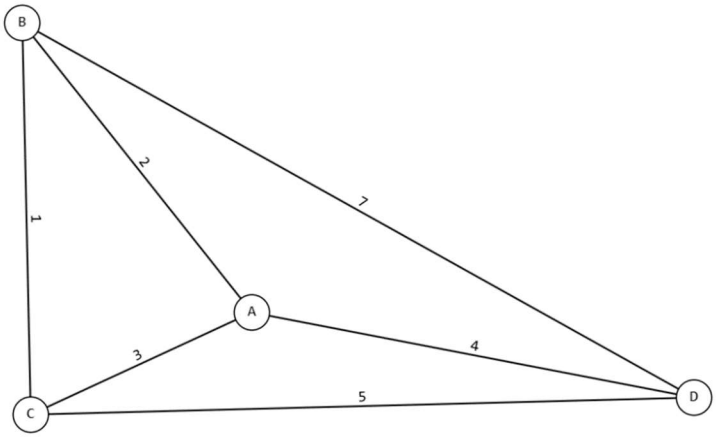

Рисунок 3 – Исходный граф для точности r=2

Пользуясь алгоритмом Краскала, построим для минимального веса остовного дерева полный взвешенный граф:

1. Отсортируем все ребра по весу
2. Добавляем их в дерево, связывая их, избегая циклов.

Путём дублирования каждого ребра получаем вершины с четными степенями: A–B–B–C–C–D–D–C–B–A. Теперь построим эйлеров цикл. Записывая только первые посещения вершин: A–B–C–D–C–B–A. Чтобы преобразовать эйлеров цикл в гамильтонов, выполним следующее: начиная с произвольной вершины, отмечаем посещённые вершины, двигаясь по эйлерову циклу. При встрече помеченной вершины пропускаем её и продолжаем движение до первой непомеченной или до возврата в первую: A–B–C–D–A. Тогда маршрут равен 2+1+5+4=12.

Еще одним примером является алгоритм Кристофидеса-Сердюкова. Ключевое отличие заключается в способе получения эйлерова графа. Возьмем 4 вершины A, B, C, D и расстояния A–B=2, B–C=1, C–D=2, A–D=3. По алгоритму Краскала строим минимальное остовное дерево: A–B–C–D. Выделяем нечетные вершины и составляем паросочетание: A–D, которое будет минимальным и единственным. Объединяем дерево с паросочетанием: A–B–C–D–A. Строим эйлеров и гамильтонов цикл и получаем в итоге маршрут A–B–C–D–A и его длину, равную 8.

А что будет, если скомбинировать несколько алгоритмов? Попробуем использовать не одно минимальное остовное дерево, а два. Пусть граф задан матрицей расстояний (рисунок 4):

0	3	4	2	5
3	0	1	4	3
4	1	0	3	2
2	4	3	0	6
5	3	2	6	0

Рисунок 4 – Матрица расстояний для алгоритма Кристофидеса-Сердюкова

По алгоритму Краскала находим два минимальных остовных дерева, причем для второго зададим ограничение длина ребра ≤ Opt/3. Первое дерево T1 имеет такие ветви, как B–C, A–D, C–E, A–B, второе T2 , исходя из ограничения, тоже содержит ветви B–C, A–D, C–E, A–B. Вес деревьев равен 8. Дальше найдем общие ребра для двух деревьев, которые образуют подграф G*: B–C, A–D, C–E, A–B. Выделяем вершины нечетной степени и строим паросочетания, то есть для вершин C, D, E паросочетания: C–D=3, C–E=2, D–E=6. Выбираем минимальное и объединяем с T1 и T2: A–B–C–E–C–B–A–D–A. Преобразуем в гамильтонов цикл: A–B–C–E–B–D–A. Opt из оптимального цикла A–D–C–E–B равно 13. Значит, длина равна 14, а r=14/13=1.08.

Кроме конструктивных алгоритмов существуют также методы, основанные на имитации естественного отбора, а также наследования. Генетический алгоритм – метод, вдохновленный эволюцией в природе. Основой данного алгоритма выступают такие понятия как популяция и особи. Алгоритм работает не с одним решением, а сразу с группой (популяцией). Каждое решение (особь) – это последовательность городов для коммивояжёра. Чем лучше решение – тем выше его "приспособленность".

Понятия, используемые в данном алгоритме:

Хромосома – последовательность вершин, которые образуют маршрут.

Скрещивание – операция, при которой две хромосомыобмениваются своими частями.

Мутация – случайное изменение одной или нескольких позиций в хромосоме.

Генетический алгоритм начинается с генерации начальной популяции. Создаем первый маршрут в популяции: вставляем начальную вершину и добавляем следующие в порядке возрастания. Дальше генерируем популяцию, переставляя гены случайным образом, пока не достигнем нужного размера. Для каждой особи вычисляем приспособленность, то есть длину маршрута. Далее выбираем два решения, родители S1 и S2, используя вероятностный оператор. По этим решениям оператор скрещивания строит новое решение S´, которое подвергается мутациям. После этого проверяем корректность решения. Далее в отсортированную по возрастанию длины маршрута популяцию вставляем выживших потомков. Так как размер популяции обычно фиксированный, то оставляем N лучших. Жизненный цикл популяции длится пока не сменят друг друга заданное число поколений (рисунок 5):

Рисунок 5 – Генетический алгоритм

Можно сказать, что производительность генетического алгоритма зависит от качества начальной популяции. И мы подумали, что можно использовать конструктивные алгоритмы для оптимизации генетического, а именно использовать для создания начальной популяции. Также мы заметили, что если использовать один метод конструктивного алгоритма, то существует риск снижения разнообразия, то есть алгоритм может застрять в локальном оптимуме. Поэтому, мы предлагаем использовать несколько разных конструктивных алгоритмов или комбинировать их со случайной генерацией, чтобы сохранить разнообразие в начальной популяции.

Рассмотрим на примере. Пусть дана матрица расстояний для 5 городов (рисунок 6):

0	10	15	20	25
10	0	35	25	30
15	35	0	30	20
20	25	30	0	15
25	30	20	15	0

Рисунок 6 – Матрица расстояний для 5 городов для комбинированного способа

Всего возможно (5 - 1)! маршрутов, то есть 24. Поэтому максимальный размер популяции − 24 уникальных решения. Используем алгоритм "Ближайший сосед" для создания 6 особей (~25%), алгоритм Кристофидеса-Сердюкова для создания 6 особей (~25%) и случайной генерацией создадим 12 особей (~50%).

Таким образом, начальная популяция будет сформирована следующим образом:

• 6 решений (25%) с помощью алгоритма ближайшего соседа, который быстро строит разумные, но не обязательно оптимальные маршруты;

• 6 решений (25%) с использованием алгоритма Кристофидеса-Сердюкова, который строит приближённое решение задачи коммивояжёра с гарантированным качеством (не более 1.5 от оптимума для метрических пространств);

• 12 решений (50%) получены случайной генерацией, что вносит необходимое разнообразие.

Далее генетический алгоритм будет выполняться в стандартном виде:

1. Оценка приспособленности каждой особи (длина маршрута).
2. Селекция: отбор лучших решений на основе приспособленности.
3. Кроссовер для генерации потомков.
4. Мутация для поддержания разнообразия.
5. Замена: формирование новой популяции (например, замена худших решений или частичная замена).

Для подтверждения эффективности данного алгоритма мы написали программу на языке программирования Java, которая сравнивает скорости выполнения классического генетического алгоритма и ГА + конструктивные алгоритмы.

Для матрицы расстояний для 5 городов (рисунок 6) результат следующий (рисунок 7):

```
1. Running Standard GA...

2. Running Hybrid GA...

=== STANDARD GA RESULTS ===
Best route: [4, 3, 1, 0, 2]
Route length: 85
Execution time: 8 ms
========================

=== HYBRID GA RESULTS ===
Best route: [0, 1, 3, 4, 2]
Route length: 85
Execution time: 3 ms
========================
```

Рисунок 7 – Результат комбинированного алгоритма для 5 городов

Это подтверждает эффективность нашей вариации алгоритма для 5 городов. Для 100 городов результат следующий (рисунок 8):

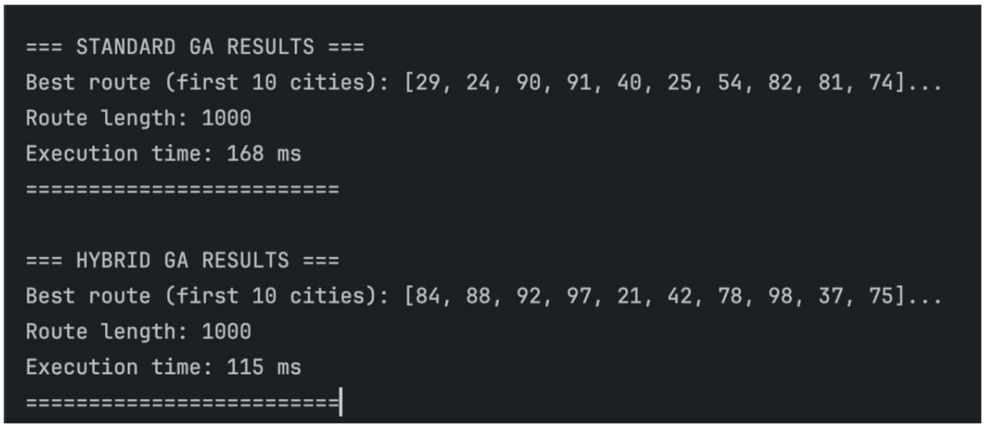

Рисунок 8 – Результат комбинированного алгоритма для 100 городов

Что также подтверждает эффективность нашей вариации алгоритма для большего количества городов.

В ходе выполненного исследования была тщательна изучена задача коммивояжера, рассмотрены основные подходы к её решению такие, как конструктивные алгоритмы, методы на основе остовных деревьев и эйлеровых циклов, генетические алгоритмы. Наибольший практический интерес представляет предложенный нами комбинированный метод, в котором начальная популяция формируется за счёт решений, полученных алгоритмами "Ближайшего соседа" и Кристофидеса-Сердюкова, дополненных случайной генерацией. Такой метод обеспечивает баланс между скоростью вычислений и точностью, предотвращая застревание в локальных оптимумах, а экспериментальная проверка подтвердила высокую эффективность как для небольших (5 городов), так и для масштабных (100 городов) задач, показав его существенные преимущества перед традиционными реализациями генетического алгоритма. Полученные результаты убедительно свидетельствуют, что выбор конкретного алгоритма должен осуществляться с учетом требований к точности и быстродействию решения, причем разработанный комбинированный метод предлагает оптимальный баланс этих характеристик для широкого круга практических применений.

Список использованных источников:

1. Chartrand, G. Graphs & Digraphs / G. Chartrand, L. Lesniak, P. Zhang. – 5th Ed. – Boca Raton : Crc Press, 2011. – 586 P.

2. Christofides, N. Worst-Case Analysis Of A New Heuristic For The Traveling Salesman Problem : Technical Report / N. Christofides. – Pittsburgh : Carnegie Mellon University, 1976. – 32 P. – (Technical Report; № 388).

3. Cook, W. J. In Pursuit Of The Traveling Salesman: Mathematics At The Limit Of Computation / W. J. Cook. – Princeton : Princeton University Press, 2012. – 228 P.

UDC 519.168:519.714.7

THE TRAVELING SALESMAN'S PROBLEM

Kharkevich M.A, Zarivniak V.S.

Belarusian State University of Informatics and Radioelectronics[1], Minsk, Republic of Belarus

Barkova E.A. – PhD in Physics and Mathematics

Annotation. The main objective of this scientific work is to analyze existing methods for solving the traveling salesman problem and optimize them. In this work, constructive algorithms “Nearest neighbor”, an algorithm with accuracy of 2, the Christofides-Serdyukov algorithm and genetic algorithms were analyzed. An algorithm was developed that combines constructive and genetic methods of solution.

Keywords. Traveling salesman problem, route, spanning tree, Euler cycle, Hamiltonian cycle, constructive algorithm, genetic algorithm, population, individuals, chromosome, crossover, fitness, optimum, selection, accuracy, distance matrix, nearest neighbor, Kruskal, matching, mutation, combined method, cities, application.