УДК 519.863

ПРИБЛИЖЁННЫЕ АЛГОРИТМЫ ДЛЯ ЗАДАЧИ КОММИВОЯЖЁРА: АЛГОРИТМ МУРАВЬИНОЙ КОЛОНИИ И 2-OPT

Юрин А.С.

студент,

Российский университет транспорта (МИИТ),

Москва, Россия

Иванова А.П.

к.ф.-м.н., доцент

Российский технологический университет – МИРЭА,

Российский университет транспорта (МИИТ),

Москва, Россия

Аннотация:

В статье проведено исследование алгоритма муравьиной колонии и улучшающего алгоритма 2-opt для решения симметричной задачи коммивояжёра. Была написана программа на языке Python. Расчёты показали, что алгоритм 2-opt даёт хороший результат, если в качестве начального взять маршрут, полученный с помощью алгоритма муравьиной колонии, но если в качестве начального взять маршрут, сгенерированный случайно, он может работать дольше и дать худшее решение. Приведены графики зависимости времени работы алгоритмов от количества вершин.

Ключевые слова: симметричная задача коммивояжёра (TSP), алгоритм муравьиной колонии, алгоритм 2-opt.

COMPARATIVE OF VERTICES

Yurin A.S.

student,

Russian University of Transport (MIIT),

Moscow, Russia

Ivanova A.P.

Ph.D., Associate Professor

MIREA – Russian Technological University,

Russian University of Transport (MIIT),

Moscow, Russia

Abstract: The article investigates the ant colony algorithm and the 2-opt improving algorithm for solving the symmetric traveling salesman problem. The program was written in Python. Calculations have shown that the 2-opt algorithm gives a good result if you use the route obtained using the ant colony algorithm as the initial one, but if you use a randomly generated route as the initial one, it can take longer and give a worse solution. Graphs of the dependence of the running time of the algorithms on the number of vertices are given.

Keywords: symmetric traveling salesman problem (TSP), ant colony algorithm, 2-opt algorithm.

Задача коммивояжёра (TSP) является одной из наиболее известных и изучаемых задач в теории графов [4-6,8]. Она представляет собой классическую проблему оптимизации, решение которой имеет множество практических применений.

Задача коммивояжёра на сегодняшнее время применяется в различных сферах, таких как: маршрутизация транспортных потоков и выбор оптимальной траектории движения рабочего инструмента, а также в сферах, которые на первый взгляд не связаны с маршрутизацией: секвенирования нуклеотидных последовательностей биополимеров, эвристического определения схожести строк, построения практических алгоритмов исследования специально определённых бесконечных грамматических структур и построения эволюционных деревьев.

Её актуальность обусловлена принадлежностью к классу NP-трудных задач, что означает отсутствие известных полиномиальных алгоритмов для её точного решения [4-6,8]. На практике это заставляет исследователей разрабатывать и применять приближённые методы, которые позволяют получать достаточно качественные решения в приемлемые сроки.

Цель данной работы состоит в исследовании приближённого алгоритма муравьиной колонии, а также возможностей улучшения полученного им решения с помощью 2-opt алгоритма.

Задача коммивояжёра формулируется следующим образом. Пусть задан граф $G=(V,E)$, где V – множество вершин, E – множество рёбер. Граф G – полный взвешенный, $C=(c_{ij})_{n\times n}$ – матрица весов, $n=|V|$. Требуется найти гамильтонов цикл минимального суммарного веса.

Кратко приведём описание алгоритма муравьиной колонии [10].

Алгоритм муравьиной колонии основан на наблюдении за поведением реальных муравьёв, которые находят оптимальные маршруты между источниками пищи и гнездом, используя систему феромонов. Алгоритм состоит из следующих основных этапов:

1. Инициализация параметров.

На начальном этапе задаются основные параметры алгоритма:

– Общее количество итераций.

– Число муравьёв, участвующих в построении маршрутов (задаётся в виде числа, которое в дальнейшем умножается на число вершин для получения количества муравьёв).

– Доля лучших маршрутов, на которые будет добавляться феромон.

– Коэффициент испарения феромона, определяющий скорость уменьшения его концентрации на рёбрах (число, на которое будут умножаться все феромоны после каждой итерации, то есть чем меньше число, тем быстрее будут испаряться феромоны).

– Влияние феромона на выбор следующего шага муравья, регулирующее баланс между длиной маршрута и уровнем феромона (например, значение 0.2 смещает акцент в сторону длины маршрута).

2. Построение маршрутов.

Каждый муравей начинает с произвольно выбранной стартовой вершины графа. При переходе между вершинами муравей выбирает ребро с учётом двух факторов:

– Длина ребра (короткие рёбра предпочтительнее).

– Уровень феромона (рёбра с более высокой концентрацией феромона имеют больший приоритет).

Выбор осуществляется случайным образом, но с учётом вышеуказанного веса, что имитирует процесс принятия решений реальных муравьёв.

3. Обновление феромонов.

После завершения маршрута всеми муравьями производится обновление концентрации феромонов на рёбрах графа.

Рёбра, которые входят в состав наиболее удачных маршрутов, усиливаются за счёт дополнительного феромона.

На остальных рёбрах феромон постепенно испаряется, чтобы избежать чрезмерного накопления и обеспечить возможность поиска новых решений.

4. Итеративный процесс.

Шаги построения маршрутов и обновления феромонов повторяются в течение заданного числа итераций. В процессе работы алгоритма качество решений постепенно улучшается, так как феромоны концентрируются на рёбрах, формирующих оптимальные или близкие к оптимальным маршруты.

Кратко приведём описание 2-opt алгоритма [2].

Алгоритм 2-opt создан для того чтобы избавляться от самопересечения маршрута. Он делает это, инвертируя четвёрки вершин в пути и если это приводит к улучшению маршрута, то переставляет вершины и начинает заново.

Когда пройдёт до конца и не найдёт ни одной четвёрки, улучшающей маршрут, то остановится.

Пример. Пусть в графе 5 вершин. Координаты (x_i, y_i) i-ой вершины графа будем генерировать как псевдослучайные равномерно распределённые числа в интервале [0;100]. Пусть сгенерированы координаты:

(10;20), (30;40), (50;25), (15;5), (45;10).

Вычислим элементы симметричной матрицы весов (расстояний):

$$c_{ij} = c_{ji} = \begin{cases} \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}, & i \neq j, \\ \infty, & i = j. \end{cases} \quad (1)$$

Для удобства округлим результаты вычислений до одного знака после запятой, тогда матрица расстояний будет иметь вид:

$$C = \begin{pmatrix} \infty & 28.3 & 40.3 & 15.8 & 36.4 \\ 28.3 & \infty & 25.0 & 38.1 & 33.5 \\ 40.3 & 25.0 & \infty & 40.3 & 15.8 \\ 15.8 & 38.1 & 40.3 & \infty & 30.4 \\ 36.4 & 33.5 & 15.8 & 30.4 & \infty \end{pmatrix}.$$

Соответствующий граф изображён на рис. 1.

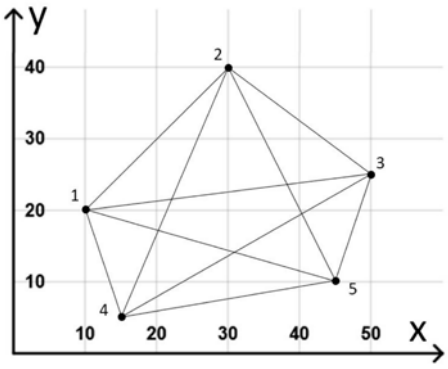
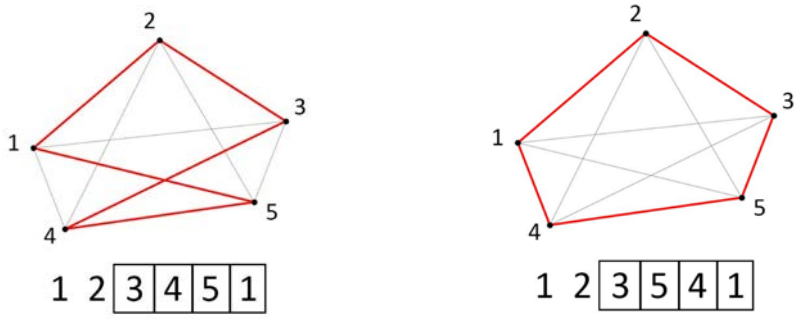


Рис. 1 – Пример графа для n=5. Авторская разработка

Для сгенерированного графа приведём пример работы 2-opt алгоритма. Пусть получен маршрут, показанный на рис. 2, а). После перестановки вершин четвёрки 3,4,5,1 получен маршрут, представленный на рис. 2, б).



а) б)

Рис. 2 – Иллюстрация работы 2-opt алгоритма. Авторская разработка

Далее будем исследовать возможности совместного применения алгоритмов муравьиной колонии и 2-opt. Рассмотрим случайно сгенерированный граф с 1000 вершинами: координаты вершин – равномерно распределённые числа на интервале [0;100], матрица расстояний вычислена по формуле (1).

На рис. 3 приведён маршрут, построенный с помощью алгоритма муравьиной колонии (параметры алгоритма: общее количество итераций – 5, количество муравьев – 1, количество лучших маршрутов – 0.1, коэффициент испарения феромона – 1, вес феромона – 0.5): длина маршрута 4250, время работы алгоритма – 518 с.

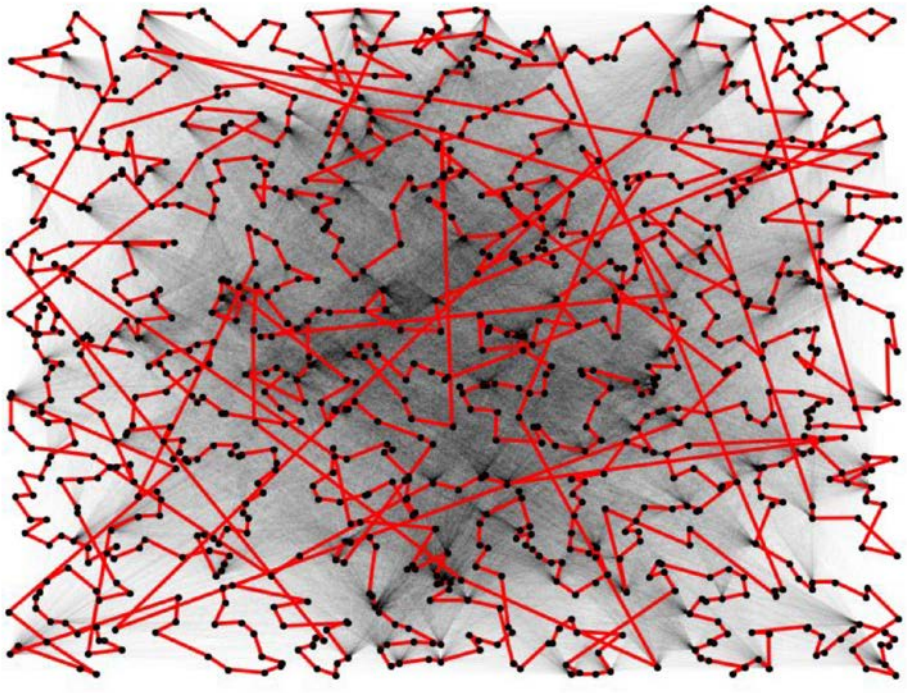


Рис. 3 – Результат работы алгоритма муравьиной колонии для n=1000, длина маршрута 4250, время работы – 518 с. Авторская разработка

На рис. 4 приведён маршрут, улучшенный с помощью 2-opt алгоритма, в качестве начального взят маршрут с рис. 3, построенный с помощью алгоритма муравьиной колонии: длина маршрута 2507, время работы – 143 с. Суммарное время работы обоих алгоритмов равно 661 с.

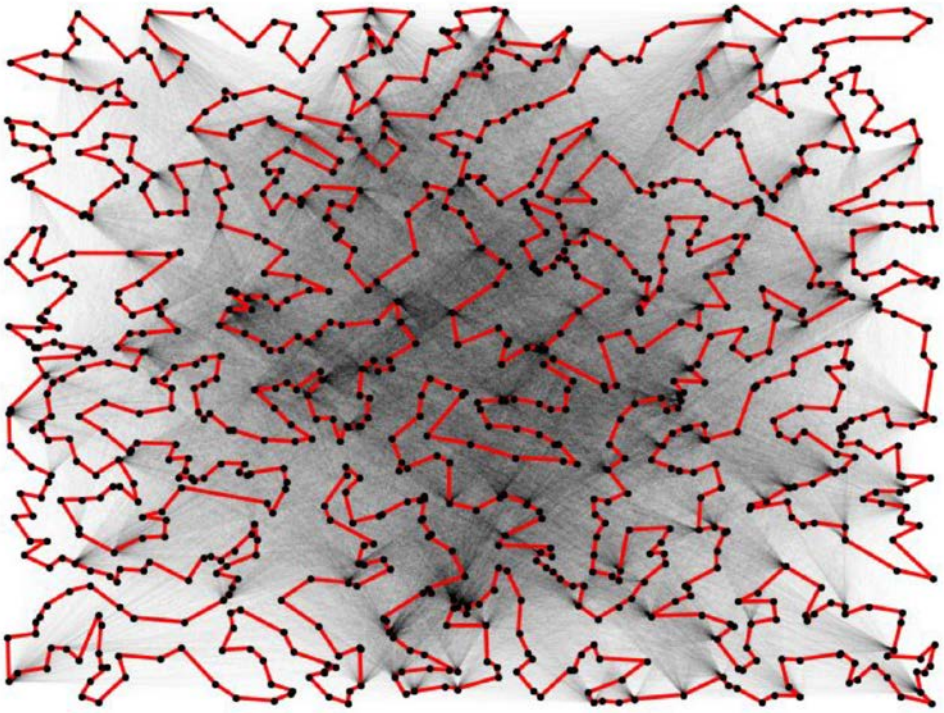


Рис. 4 – Результат после улучшения с помощью алгоритма 2-opt для *n*=1000, длина маршрута 2507, время работы – 143 с. Авторская разработка

Далее сгенерируем случайный маршрут и улучшим его с помощью 2-opt алгоритма. Для генерации случайного маршрута сначала создаётся список целых чисел от 0 до *n*-1, например, для *n*=5: [0,1,2,3,4]. Затем этот список с помощью функции np.random.shuffle() из библиотеки numpy перемешивается, чтобы получить случайный путь, например, получим [1,2,4,3,0]. Затем в конец добавляется первая вершина, чтобы получить замкнутый путь: [1,2,4,3,0,1]. После этого последовательность вершин преобразуется в последовательность рёбер строкой edges = [(nodes[i], nodes[i + 1]) for i in range(n)]. В примере список [1,2,4,3,0,1] преобразуется в последовательность рёбер [(1,2),(2,4),(4,3),(3,0),(0,1)].

На рис. 5 приведён полученный маршрут (*n*=1000), длина маршрута 2583, время работы – 808 с.

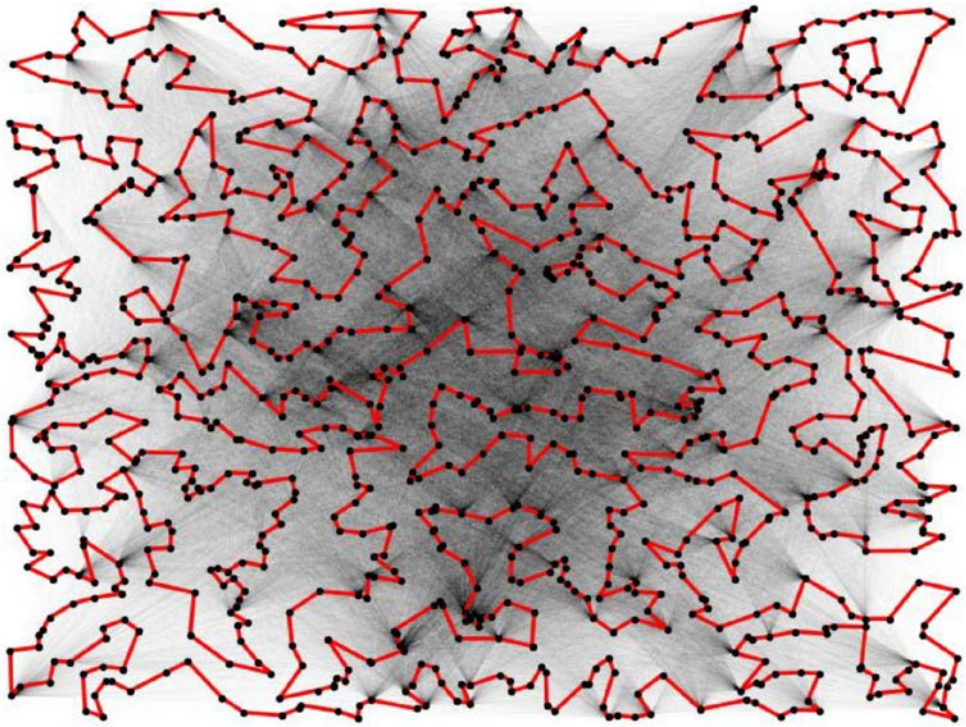


Рис. 5 – Результат работы алгоритма 2-opt для *n*=1000, длина маршрута 2583, время работы – 808 с. Авторская разработка

В табл.1 приведены результаты вычислений для графов с разным количеством вершин, на рис. 6 – зависимости времени работы алгоритмов от количества вершин графа, для каждого значения *n* генерировался один случайный граф, как это описано выше.

Таблица 1 – Время работы алгоритмов и полученная длина маршрута для графов с разным количеством вершин

n – количество вершин графа	Алгоритм муравьиной колонии		Алгоритм муравьиной колонии + улучшение с помощью алгоритма 2-opt		Случайный маршрут + улучшение с помощью алгоритма 2-opt	
	Время	Длина	Время	Длина	Время	Длина
500	82,3	3049,0	82,3+14,3=96,6	1748,0	66,8	1761,0
1000	530,7	4278,0	530,7+65,9= 596,6	2552,0	360,4	2552,0
1500	1627,2	4902,0	1627,2+287,6= 1914,8	3095,0	1232,3	3104,0
2000	3655,6	5610,0	3655,6+576,9= 4232,5	3630,0	4079,3	3614,0
2500	6989,5	6355,0	6989,5+1047,8= 8037,3	4068,0	11076,3	4124,0

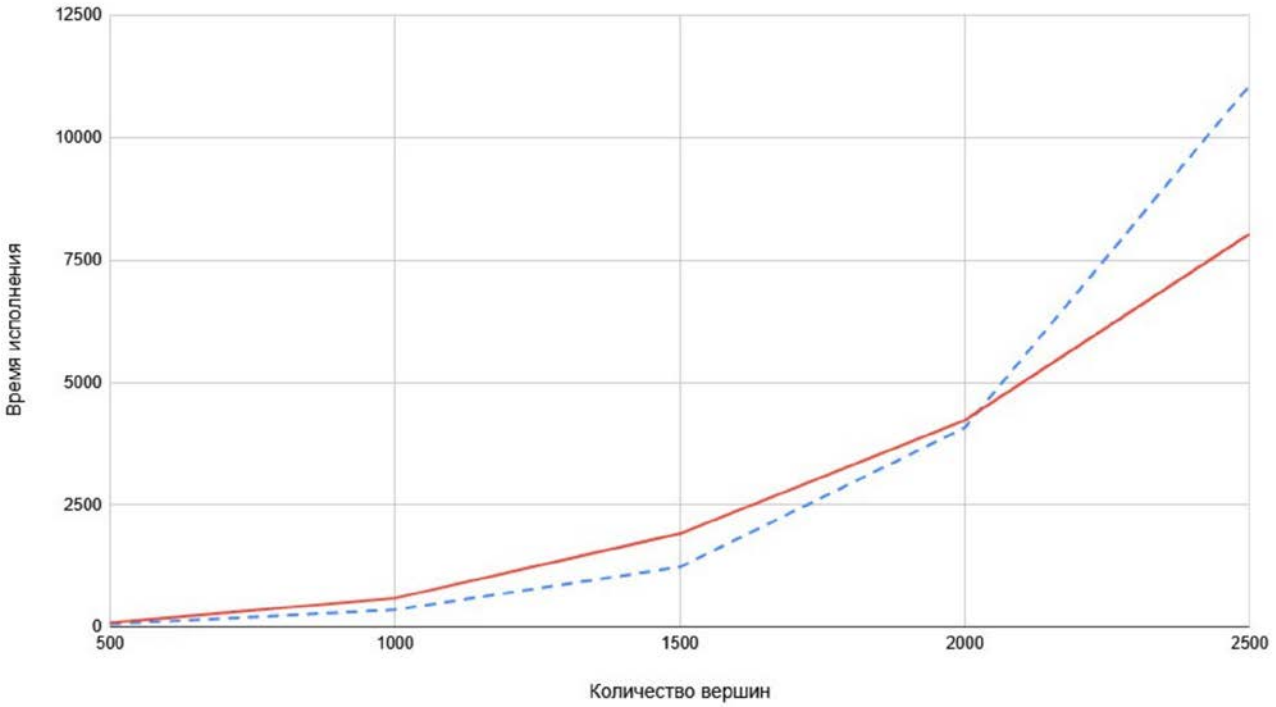


Рис. 6 – Зависимость времени работы алгоритмов от количества вершин графа. Сплошная линия – алгоритм муравьиной колонии + улучшение с помощью алгоритма 2-opt. Штриховая линия – случайный маршрут + улучшение с помощью алгоритма 2-opt. Авторская разработка

На основании полученных результатов можно сделать вывод, что алгоритм 2-opt даёт хороший результат улучшения, если в качестве начального маршрута взять маршрут, полученный с помощью алгоритма муравьиной колонии, но если в качестве начального маршрута взять маршрут, сгенерированный случайно, он может работать дольше и дать худшее решение.

Библиографический список:

1. Алексеев В.Е, Захарова Д.В. Теория графов: Учебное пособие. – Нижний Новгород: Нижегородский госуниверситет, 2017. – 119 с.

2. Колесников А.В., Кириков И.А., Листопад С.В., Румовская С.Б., Доманицкий А.А. Решение сложных задач коммивояжера методами функциональных гибрид-

ных интеллектуальных систем / Под ред. А.В. Колесникова. – М.: ИПИ РАН, 2011. – 295 с.

3. Кормен, Томас Х., Лейзерсон, Чарльз И., Ривест, Рональд Л., Штайн, Клиффорд. Алгоритмы: построение и анализ, 2-е издание.: Пер. с англ. – М.: Издательский дом «Вильямс», 2022. – 1296 с.

4. Кристофидес Н. Теория графов. Алгоритмический подход. – М.: Мир, 1978. – 432 с.

5. Лекции по теории графов / Емеличев В.А., Мельников О.И., Сарванов В.И., Тышкевич Р.И. – М.: Наука, 1990. – 384 с.

6. Мудров В.И. Задача о коммивояжёре. – М.: Знание, 1969.

7. Озерова, Г.П. Основы программирования на языке Python в примерах и задачах: учебное пособие для вузов / Политехнический институт ДВФУ. – Владивосток: Изд-во Дальневост. федерал. ун-та, 2022. – 128 с.

8. Сигал, И.Х. Введение в прикладное дискретное программирование: модели и вычислительные алгоритмы: учебное пособие / И.Х. Сигал, А.П. Иванова. – 2-е изд., испр. и доп. – М.: ФИЗМАТЛИТ, 2007. – 304 с.

9. Теория графов. Часть 1: учебное пособие по дисциплине «Теория графов»: Иванова А.П. – Москва: Янус-К, 2024. – 96 с.

10. Штовба С.Д. Муравьиные алгоритмы // Exponenta Pro. Математика в приложениях, 2003. – №4. – С. 70–75.

Оригинальность 81%