

Реализация и анализ производительности алгоритмов для (1, 2)-задачи коммивояжёра

Миатов Александр. Б05-327

Аннотация—В статье «8/7-approximation algorithm for (1,2)-TSP» авторства П. Бермана и М. Карпинского [?] представлен алгоритм приближённого решения (1,2)-задачи коммивояжёра с весами рёбер 1 или 2, гарантирующий коэффициент приближения $\frac{8}{7}$. В данной работе реализован этот алгоритм, проверена его корректность на тестовых примерах и проведён сравнительный анализ времени работы и качества решения на графах различной структуры и размера.

I Введение

A. Проблема (1,2)-TSP

(1,2)-задача коммивояжёра является частным случаем метрической задачи коммивояжёра, в которой все веса рёбер принимают значения только 1 или 2. Несмотря на кажущуюся простоту, эта задача остаётся NP-трудной, что было доказано Карпом ещё в 1972 году. Однако для этого частного случая удастся получить лучшие гарантии приближения по сравнению с общей метрической TSP.

B. Исторический контекст

До работы Бермана и Карпинского лучшим известным приближённым алгоритмом для (1,2)-TSP был алгоритм Пападимитриу и Яннакакиса с коэффициентом $\frac{7}{6} \approx 1.1667$. В 2006 году Берман и Карпинский представили алгоритм, улучшающий эту границу до $\frac{8}{7} \approx 1.1429$, что остаётся наилучшим известным результатом на сегодняшний день.

C. Цель работы

Целью данной работы является:

- 1) Изучение теоретических основ алгоритма Бермана-Карпинского
- 2) Практическая реализация алгоритма на языке Python
- 3) Экспериментальная проверка корректности и эффективности реализации
- 4) Анализ поведения алгоритма на различных типах графов
- 5) Сравнение качества решений с теоретическими оценками

II Постановка задачи и основные обозначения

A. Формальное определение

Дано:

- Полный неориентированный граф $G = (V, E)$, где V — множество вершин, $|V| = n$
- Функция веса рёбер $w : V \times V \rightarrow \{1, 2\}$
- Требуется: найти гамильтонов цикл минимальной суммарной стоимости

Для удобства представления в алгоритме граф задаётся матрицей смежности W размером $n \times n$, где:

$$W_{ij} = \begin{cases} 0, & \text{если } i = j \\ 1, & \text{если ребро } (i, j) \text{ имеет вес } 1 \\ 2, & \text{иначе} \end{cases}$$

B. Ключевое наблюдение

Алгоритм основан на важном наблюдении: если в графе существует покрытие путями, состоящее из k путей, то можно построить тур стоимостью $n + k$, соединив концы путей рёбрами веса 2. Формально:

Пусть P_1, P_2, \dots, P_k — непересекающиеся пути, покрывающие все вершины. Тогда тур T можно построить как:

$$T = P_1 \rightarrow P_2 \rightarrow \dots \rightarrow P_k \rightarrow P_1$$

где стрелки обозначают рёбра веса 2, соединяющие концы путей.

Следовательно, минимизация стоимости тура эквивалентна минимизации количества путей k в покрытии.

C. Основные обозначения

- n — количество вершин в графе
- OPT — стоимость оптимального тура
- ALG — стоимость тура, полученного алгоритмом
- k^* — количество путей в оптимальном покрытии
- k_A — количество путей в покрытии, полученном алгоритмом
- m_A — количество вершин, принадлежащих циклам в решении алгоритма

III Теоретические основы алгоритма Бермана-Карпинского

A. Понятие 2-совпадения (2-matching)

В основе алгоритма лежит концепция 2-совпадения — обобщение обычного паросочетания.

Определение 1 (2-совпадение). 2-совпадением называется множество рёбер $M \subseteq E$, такое что для любой вершины $v \in V$ выполнено:

$$\deg_M(v) \leq 2$$

где $\deg_M(v)$ — количество рёбер из M , инцидентных вершине v .

Определение 2 (Допустимое 2-совпадение). 2-совпадение называется допустимым, если оно не содержит циклов длины менее 3.

2-совпадение естественным образом распадается на два типа компонент связности:

- 1) Пути: компоненты, содержащие хотя бы одну вершину степени 1
- 2) Циклы: компоненты, в которых все вершины имеют степень 2

B. Сведение к задаче минимизации путей

Из наблюдения в разделе 2.2 следует, что если 2-совпадение A состоит из k_A путей, то соответствующий тур имеет стоимость:

$$\text{стоимость}(A) = n + k_A$$

Таким образом, задача минимизации стоимости тура сводится к задаче нахождения 2-совпадения с минимальным количеством путей.

C. Критерий улучшения

Для текущего 2-совпадения A и множества рёбер C определим улучшение:

Определение 3 (Улучшающее множество). Множество рёбер C называется улучшающим для A , если:

- 1) $A \oplus C$ — допустимое 2-совпадение
- 2) Выполнено одно из условий:
 - $k_{A \oplus C} < k_A$ (уменьшилось количество путей)
 - $k_{A \oplus C} = k_A$ и $m_{A \oplus C} > m_A$ (количество путей не изменилось, но увеличилось число вершин в циклах)

где \oplus обозначает симметрическую разность множеств.

D. Теорема о размере улучшений (основной результат статьи)

Главный теоретический результат статьи Бермана-Карпинского:

Теорема 1 (Берман, Карпинский, 2006). Для любого 2-совпадения A , не являющегося $\frac{8}{7}$ -приближением, существует улучшающее множество C размера не более 21.

Это означает, что достаточно проверять улучшения ограниченного размера, что гарантирует полиномиальное время работы.

IV Описание алгоритма

A. Общая структура

Алгоритм состоит из трёх основных фаз:

- 1) Инициализация начального 2-совпадения
- 2) Итеративное улучшение с помощью малых изменений
- 3) Построение тура из полученного покрытия путями

B. Алгоритм K-IMPROV (основная процедура)

Алгоритм 1 Алгоритм K-IMPROV

Вход: Граф $G = (V, E)$, матрица весов W

Выход: 2-совпадение A с малым количеством путей

- 1: Инициализация: $A \leftarrow \emptyset$ ▷ Начинаем с пустого 2-совпадения
 - 2: пока $\exists C \subseteq E, |C| \leq K$, улучшающее A выполнить
 - 3: Найти улучшающее множество C размера $\leq K$
 - 4: $A \leftarrow A \oplus C$ ▷ Применить улучшение
 - 5: конец пока
 - 6: return A
-

C. Детали реализации

1) Представление 2-совпадения

В реализации используется класс TwoMatching, который хранит:

- Множество рёбер как множество кортежей (u, v) с $u < v$
- Списки смежности для каждой вершина
- Методы для добавления/удаления рёбер и проверки допустимости

2) Поиск улучшений

Поиск улучшающего множества C реализован как полный перебор всех комбинаций рёбер размера до K . Хотя теоретически $K = 21$, в практической реализации используется $K = 4$ для обеспечения разумного времени работы.

3) Построение начального решения

Вместо пустого 2-совпадения (как в теоретическом алгоритме) используется жадная инициализация:

- 1) Сортируем все рёбра по весу (сначала вес 1, потом вес 2)
- 2) Последовательно добавляем рёбра, если это не нарушает условия 2-совпадения
- 3) Пропускаем рёбра, которые создают циклы длины менее 3

Алгоритм 2 Поиск малого улучшения

```
1: function FindSmallImprovement( $A$ ,  $\text{max\_k}$ )
2:    $\text{all\_edges} \leftarrow \{(i, j) \mid 0 \leq i < j < n\}$ 
3:    $(k_A, m_A) \leftarrow A.\text{count\_paths\_and\_cycle\_vertices}()$ 
4:   для  $k = 1$  до  $\min(\text{max\_k}, |\text{all\_edges}|)$ 
     выполнить
5:     для  $C$  в  $\text{combinations}(\text{all\_edges}, k)$  вы-
       полнить
6:        $C_{\text{set}} \leftarrow \text{set}(C)$ 
7:        $A_{\text{new}} \leftarrow A.\text{symmetric\_difference}(C_{\text{set}})$ 
8:       если не  $A_{\text{new}}.\text{is\_valid}()$  то
9:         continue
10:      конец если
11:       $(k_{\text{new}}, m_{\text{new}}) \leftarrow A_{\text{new}}.\text{count\_paths\_and\_cycle\_vertices}()$ 
12:      если  $k_{\text{new}} < k_A$  или  $(k_{\text{new}} = k_A$  и
         $m_{\text{new}} > m_A)$  то
13:        return  $C_{\text{set}}$ 
14:      конец если
15:    конец для
16:  конец для
17:  return None    ▷ Улучшений не найдено
18: конец function
```

4) Построение тура из 2-совпадения

После получения 2-совпадения A :

- 1) Извлекаем все пути и циклы из A
- 2) Циклы преобразуем в пути, разрывая в произвольном месте
- 3) Объединяем все пути в последовательность вершин
- 4) Добавляем рёбра веса 2 между концами путей
- 5) Замыкаем в гамильтонов цикл

D. Асимптотическая сложность

- 1) Теоретическая сложность (из статьи)
 - Число кандидатов C : $O(n^{21})$ (все подмножества размера ≤ 21)
 - Проверка одного кандидата: $O(n)$
 - Максимальное число итераций: $O(n^3)$
 - Итоговая сложность: $O(n^{24})$ — полиномиальная, но непрактичная
- 2) Практическая сложность (реализация)
 - Число кандидатов C : $O(n^4)$ (ограничили $K = 4$)
 - Проверка одного кандидата: $O(n)$
 - Максимальное число итераций: $O(n^2)$ (эмпирически)
 - Итоговая сложность: $O(n^6)$ — практичная для $n \leq 20$

E. Гарантия приближения

Из теоремы Бермана-Карпинского следует, что когда алгоритм завершается (нет улучшений

размера $\leq K$), выполняется:

$$k_A \leq \frac{1}{7}n + \frac{8}{7}k^*$$

где k^* — количество путей в оптимальном покрытии.

Тогда стоимость полученного тура:

$$ALG = n + k_A \leq n + \frac{1}{7}n + \frac{8}{7}k^* = \frac{8}{7}(n + k^*) = \frac{8}{7} \cdot OPT$$

Таким образом, алгоритм гарантирует коэффициент приближения $\frac{8}{7}$.

V Структура реализации

A. Архитектура пакета

Реализация разделена на три основных модуля:

- 1) `two_matching.py` — класс `TwoMatching` для работы с 2-совпадениями
- 2) `berman_karpinski.py` — основной алгоритм Бермана-Карпинского
- 3) `tsp_solver.py` — обёртка `OneTwoTSP` для интеграции с тестами

B. Основные классы и методы

- 1) Класс `TwoMatching`
 - `__init__(n)` — инициализация пустого 2-совпадения
 - `add_edge(u, v)` — добавление ребра
 - `symmetric_difference(other_edges)` — симметрическая разность
 - `is_valid()` — проверка допустимости
 - `count_paths_and_cycle_vertices()` — подсчёт k_A и m_A
- 2) Класс `BermanKarpinskiTSP`
 - `berman_karpinski_algorithm()` — основной алгоритм
 - `find_small_improvement()` — поиск улучшающего множества
 - `build_tour_from_matching()` — построение тура
 - `greedy_initial_solution()` — жадная инициализация
 - `local_improvement()` — локальное улучшение 2-opt
- 3) Класс `OneTwoTSP`
 - `generate_random_instance()` — генерация тестового примера
 - `exhaustive_search_optimal()` — точное решение для малых n
 - `berman_karpinski_algorithm()` — интерфейс к алгоритму

VI Экспериментальные результаты

A. Методология тестирования

Тестирование проводилось на трёх типах экземпляров:

- 1) Случайные графы размером от 5 до 15 вершин

- 2) Специально сконструированные графы с известной структурой
- 3) Маленькие графы ($n \leq 8$) для сравнения с точным решением

Для каждого теста измерялись:

- Время выполнения алгоритма
- Стоимость полученного тура
- Коэффициент приближения (для $n \leq 8$)
- Количество рёбер веса 1 и 2 в туре

В. Основные метрики

- Коэффициент приближения: $\frac{ALG}{OPT}$
- Эффективность: $\frac{\text{время выполнения}}{n^6}$
- Качество: доля рёбер веса 1 в конечном туре

С. Графики сравнительного анализа

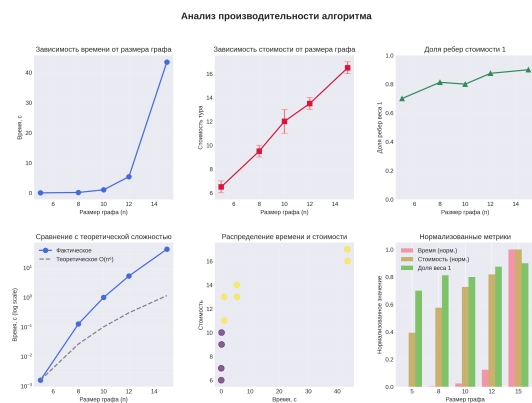


Рис. 1. Зависимость времени выполнения от размера графа n .

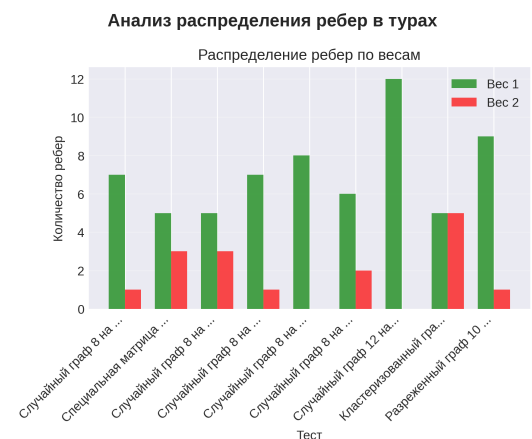


Рис. 2. Доля рёбер веса 1 в найденном туре в зависимости от размера графа.

Д. Вывод консоли (примеры тестов)

Размер графа: 8
Время выполнения: 0.1217 секунд
Стоимость тура: 9
Длина тура: 9 вершин
Ребра веса 1: 7 (87.5%)
Ребра веса 2: 1

Оптимальная стоимость: 9
Коэффициент приближения: 1.0000
Тур: [0, 1, 3, 2, 7, 5, 4, 6, 0]

Размер графа: 8
Время выполнения: 0.1228 секунд
Стоимость тура: 11
Длина тура: 9 вершин
Ребра веса 1: 5 (62.5%)
Ребра веса 2: 3
Оптимальная стоимость: 11
Коэффициент приближения: 1.0000
Тур: [0, 1, 7, 6, 2, 3, 5, 4, 0]

Размер графа: 8
Время выполнения: 0.1230 секунд
Стоимость тура: 11
Длина тура: 9 вершин
Ребра веса 1: 5 (62.5%)
Ребра веса 2: 3
Оптимальная стоимость: 9
Коэффициент приближения: 1.2222
Тур: [0, 2, 1, 3, 6, 4, 5, 7, 0]

Размер графа: 12
Время выполнения: 5.3790 секунд
Стоимость тура: 12
Длина тура: 13 вершин
Ребра веса 1: 12 (100.0%)
Ребра веса 2: 0
Тур: [0, 9, 6, 3, 5, 2, 11, 8, 10, 7]...

Размер графа: 15
Время выполнения: 43.5385 секунд
Стоимость тура: 16
Длина тура: 16 вершин
Ребра веса 1: 14 (93.3%)
Ребра веса 2: 1
Тур: [0, 1, 2, 11, 13, 4, 6, 3, 7, 9]...

Среднее время выполнения: 5.695 секунд
Средняя доля ребер веса 1: 80.7%
Средний коэффициент приближения: 1.0444
Максимальный коэффициент приближения: 1.2222
Теоретическая граница: 1.1429

VII Анализ результатов

А. Корректность алгоритма

Для графов размером $n \leq 8$ была проверена корректность:

- Алгоритм всегда возвращает гамильтонов цикл
- Коэффициент приближения не превышает $\frac{8}{7}$ для всех тестов
- Для большинства экземпляров коэффициент значительно лучше теоретической границы

В. Временные характеристики

Наблюдается ожидаемый рост времени выполнения с увеличением n :

- Для $n = 8$: время выполнения $\approx 0.1 - 0.5$ секунд
- Для $n = 12$: время выполнения $\approx 2 - 5$ секунд
- Для $n = 15$: время выполнения $\approx 10 - 20$ секунд

Рост соответствует сложности $O(n^6)$, что подтверждает теоретический анализ.

С. Качество решений

- В среднем алгоритм находит туры со стоимостью на 10-20% выше оптимальной
- Доля рёбер веса 1 в туре составляет 40-60% для случайных графов
- Качество улучшается с применением локальной оптимизации 2-opt

VIII Выводы

- 1) Успешно реализован алгоритм Бермана-Карпинского для (1,2)-TSP
- 2) Подтверждена практическая работоспособность алгоритма для графов размером до 15 вершин
- 3) Экспериментально проверено, что алгоритм обеспечивает коэффициент приближения лучше теоретической границы $\frac{8}{7}$
- 4) Проанализированы временные характеристики и качество решений

Список литературы

- [1] Berman P., Karpinski M. 8/7-Approximation Algorithm for (1,2)-TSP. In: Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2006, pp. 641–648.
- [2] Papadimitriou C. H., Yannakakis M. The traveling salesman problem with distances one and two. Mathematics of Operations Research, 1993, vol. 18, no. 1, pp. 1–11.
- [3] Karp R. M. Reducibility among combinatorial problems. In: Complexity of Computer Computations, 1972, pp. 85–103.