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 [Berman_2006] 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 [Karp_1972]. Однако для этого частного случая удаётся 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 [Papadimitriou_1993] 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 $w : V \times V \to \{1, 2\}$
- 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 $P_1, P_2, \ldots, P_k$ — непересекающиеся пути, покрывающие все вершины. Тогда тур T можно построить как:

$$T = P_1 \to P_2 \to \cdots \to P_k \to P_1$$

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

```
Инициализация: A ← ∅                    ▷ Начинаем с пустого 2-совпадения
пока ∃C ⊆ E, |C| ≤ K, улучшающее A выполнить
    Найти улучшающее множество C размера ≤ K
    A ← A ⊕ C                           ▷ Применить улучшение
конец пока
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 вершины
- 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

```
function FindSmallImprovement(A, max_k)
    all_edges ← {(i, j) | 0 ≤ i < j < n}
    (k_A, m_A) ← A.count_paths_and_cycle_vertices()
    для k = 1 до min(max_k, |all_edges|) выполнить
        для C в combinations(all_edges, k) выполнить
            C_set ← set(C)
            A_new ← A.symmetric_difference(C_set)
            если не A_new.is_valid() то
                continue
            конец если
            (k_new, m_new) ← A_new.count_paths_and_cycle_vertices()
            если k_new < k_A или (k_new = k_A и m_new > m_A) то
                return C_set
            конец если
        конец для
    конец для
    return None        ▷ Улучшений не найдено
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

B. Основные метрики

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

VII Анализ результатов

A. 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

B. 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.

C. 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.