RU EN ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ ISSN 2079-3316
научная статья МЕТОДЫ ОПТИМИЗАЦИИ И ТЕОРИЯ УПРАВЛЕНИЯ

УДК 004.421.2:519.712.43
10.25209/2079-3316-2026-17-4-137-158

Приближённый и точный алгоритмы задачи нескольких коммивояжёров

Олег Александрович **Клименко**
Южный Федеральный Университет, Ростов-на-Дону, Россия

Аннотация. Рассматривается задача нескольких коммивояжёров в минимаксной постановке без депо, которая состоит в том, чтобы на графе со взвешенными рёбрами найти набор заранее заданного количества непересекающихся циклов, покрывающих все вершины, у которого сумма весов рёбер наибольшего цикла минимальна.

Получен основанный на методе ветвей и границ точный алгоритм решения этой задачи. В полученном алгоритме, как и в известном алгоритме Балаша-Кристофидеса решения задачи одного коммивояжёра, используется венгерский алгоритм решения задачи о назначениях.

Разработана параллельная версия полученного алгоритма, а также разработана модификация, позволяющая находить приближённое решение с заданной точностью.

Приводятся результаты численных экспериментов со случайными графами большой размерности. Полученные результаты демонстрируют превосходство разработанного алгоритма по сравнению с известными аналогами.

Ключевые слова и фразы: задача коммивояжёра, задача о назначениях, венгерский алгоритм, метод ветвей и границ, параллельное программирование, приближённые алгоритмы

Благодарности:
Работа поддержана грантом № FENW-2026-0031

Для цитирования: Клименко О. А. *Приближённый и точный алгоритмы задачи нескольких коммивояжёров* // Программные системы: теория и приложения. 2026. **Т. 17**. № 4(73). С. 137–158. https://psta.psiras.ru/read/psta2026_4_137-158.pdf

© Клименко О. А. 2026 CC BY

1. Введение

В статье рассматриваются приближенный и точный параллельные алгоритмы решения задачи нескольких коммивояжёров в минимаксной постановке без депо. Эта задача является обобщением известной задачи коммивояжёра (одного), решению которой посвящено множество публикаций, в частности [1–9].

Задача коммивояжёра TSP (Traveling Salesman Problem) состоит в нахождении гамильтонова цикла с минимальной суммой весов рёбер в полном взвешенном ориентированном графе, веса рёбер которого неотрицательны. Эта задача относится к классу NP-трудных задач [10]. Из этого часто делается вывод о том, что точные алгоритмы решения этой задачи работают только для малых размерностей. Однако, на современных компьютерах для имеющих прикладное значение размерностей задачи коммивояжёра точное решение можно получать в разумное время.

Быстрая программа поиска точного решения задачи коммивояжёра описана в 1981 г. в работе Балаша и Кристофидеса [1]. Их алгоритм основан на методе ветвей и границ и реализован на компьютере CDC-7600. Программа находила точное решение для графов размерности 325 вершин со случайными целыми весами из сегмента [0,1000) за 82 секунды (в худшем случае из 100 испытаний).

Эффективная реализация этого алгоритма, адаптированная к современным массовым процессорам Интел, представлена в [2, 5, 7] и позволяет находить точное решение задачи коммивояжёра для графа с 1000 вершин и случайными весами из промежутка [0, 1000000) за 1 секунду. В частности, применяются оптимизация использования памяти и распараллеливание. Программа *Concorde* [9] уступает по быстродействию [2, 5, 7].

Полиномиальный приближенный алгоритм решения задачи коммивояжёра с оценкой качества найденного решения представлен в [4]. Найденное этим алгоритмом решение хуже оптимального не более чем в 1.5 раза, а вычислительная сложность этого алгоритма $O(N^3)$. В [5] есть приближенный алгоритм с оценкой «найденное решение хуже оптимального не более, чем в k раз ($k-$параметр)».

Этот алгоритм для случайных графов размерности 50000 вершин на компьютере с шестиядерным процессором находит решение задачи коммивояжёра, которое хуже оптимального не более, чем на 5% ($k = 1.05$). Некоторые эвристические алгоритмы решения задачи коммивояжёра представлены в [8].

Обход дерева вариантов в методе ветвей и границ можно описывать рекурсивными функциями, распараллеливание которых рассматривалось в работах [11, 12].

Такие параллельные алгоритмы существенно отличаются от распараллеливания циклов, которые могут содержать рекуррентную зависимость [13].

При распараллеливании основанных на методе ветвей и границ алгоритмов, параллельно выполняется обход разных ветвей дерева вариантов. Для высокопроизводительного кластера с распределенной памятью такое распараллеливание представлено в [14]. Для многоядерных процессоров (вычислительных систем с общей памятью) распараллеливание представлено в [2, 5, 7].

В этих работах отмечены проблемы распараллеливания основанных на методе ветвей и границ алгоритмов. Проблемы состоят в том, что ветки дерева вариантов требуют разного времени обхода, и текущее рекордное значение целевой функции является общей переменной для всех параллельно выполняемых потоков. Из-за этих проблем ускорение от распараллеливания сильно отличается от количества потоков.

В задаче нескольких коммивояжёров требуется отыскать набор из нескольких циклов, покрывающих все вершины графа ровно один раз. Обычно в этой задаче ищется такой набор циклов, сумма весов рёбер которого минимальна. Такую задачу обозначают mTSP – multiple Traveling Salesman Problem. Обзор 2006 года работ по решению этой задачи представлен в статье [15]. В [16, 17] и [18] рассматривается точное решение при небольшом количестве коммивояжёров. Рассматриваемая формулировка целевой функции минимизирует суммарную длину (стоимость) пройденного пути и может случиться, что решением окажется один цикл: один коммивояжёр обходит все вершины графа, а остальные не работают.

Будем искать такой набор заданного количества циклов (коммивояжёров), в котором максимальная сумма весов рёбер минимальна, и каждая вершина принадлежит ровно одному циклу. В такой постановке задача известна под названием Min-max cycle cover problem [19]. Она по-видимому более интересна при планировании транспортных перевозок.

От mTSP рассматриваемая задача отличается целевой функцией (максимум из весов циклов вместо их суммы) и отсутствием депо (единственной общей вершины, которую обязаны содержать все циклы). Постановка задачи с депо, но целевой функции в виде максимума весов

циклов, несомненно, также интересна, и будет рассмотрена в следующих работах. Первая версия алгоритма решения задачи представлена в работе [20].

Для построения быстрого алгоритма решения рассматриваемой задачи, как и в известном алгоритме [1], используется венгерский алгоритм решения вспомогательной задачи о назначениях.

Разработанный последовательный алгоритм распараллелен на основе OpenMP для 8-ядерного процессора. В перспективе интересно будет распараллелить этот алгоритм на процессоры с большим количеством ядер.

Приводятся численные эксперименты со случайными графами, демонстрирующие эффективность представленного алгоритма. В отличие от [1], веса рёбер случайных графов берутся не из промежутка [0,1000), а, как и в [2, 5, 7], как случайные целые числа промежутка [0,1000000) – это существенно при рассмотрении графов большой размерности. Быстродействие полученного алгоритма на порядок превосходит известный алгоритм решения mTSP [16] для 4 коммивояжёров.

На основе точного алгоритма, как и в [5], строится приближенный алгоритм, позволяющий находить решение с заданной точностью. Эффективность приближённого алгоритма обуславливается более сильной оценкой для отсечения ветвей в методе ветвей и границ.

Ниже представлено описание способа хранения данных, которое позволило получить производительность программы, превосходящую известные.

2. Метод ветвей и границ для поиска точного решения минимаксной задачи нескольких коммивояжёров без депо

Задача 1. ***Дан*** *ориентированный граф*

$$G = (V, w),$$

где

V – *множество вершин,*
$w : V \times V \to \mathbb{R}^+$ – *весовая функция рёбер, а*
$\mathbb{R}^+$ – *множество неотрицательных вещественных чисел.*

Требуется *найти такой набор k простых циклов $c_1, c_2, \ldots, c_k$, что каждая вершина графа принадлежит ровно одному циклу и наибольшая стоимость (сумма весов рёбер) циклов набора минимальна.*

В дальнейшем целевую функцию будем обозначать f. Будем искать точное решение задачи методом ветвей и границ. В методе ветвей и границ исходная задача рекурсивно разбивается на более простые подзадачи. Рекурсивные вызовы алгоритма решения задачи образуют дерево вариантов, лежащее в основе метода ветвей и границ. Псевдокод для метода ветвей и границ представлен в листинге 1.

```
def solve(matrix, k):
    P_0 = init(matrix, k) # инициализируем исходную задачу
    REC = INF_SOLUTION # инициализируем исходное рекордное решение.
    # Пока ни одного настоящего решения пока не найдено,
    # считаем, что INF_SOLUTION это решение заведомо худшее, чем любое
    ↪ допустимое решение.
    q = queue() # инициализируем очередь для подзадач
    q.push(P_0) # добавляем исходную задачу в очередь
    while q: # пока очередь не пуста...
        P = q.pop_min() # извлекаем из q подзадачу с наименьшей нижней
        ↪ оценкой LB
        P_1, P_2, ..., P_l = split(P, REC) # разбиваем подзадачу на дочерние
        ↪ подзадачи.
        # Если в процессе отыскали решение лучшее, чем REC, обновляем его.
        q.push(P_1, P_2, ..., P_l) # добавляем новые подзадачи в очередь
        # убираем подзадачи с нижней оценкой не меньшей, чем рекорд
        while q.max >= f(REC):
            q.pop_max()
    return REC
```

Листинг 1. Псевдокод метода ветвей и границ (без деталей)

Как и в алгоритме Балаша-Кристофидеса, для рассматриваемой задачи дерево вариантов является модификацией дерева вариантов алгоритма Литтла [6]. для задачи одного коммивояжёра. Как и в [1, 5, 6] на каждом шаге алгоритма множество всех рёбер графа разбивается на три подмножества, которые называются замороженные, запрещенные и нерассмотренные.

Замороженные рёбра – это те, о которых предполагается, что они принадлежат одному из циклов искомого набора.

Запрещенные – те, о которых предполагается, что они не принадлежат ни одному из циклов искомого набора.

Нерассмотренные рёбра – это все остальные (соответственно термину «нерассмотренные»).

В рассматриваемой задаче, как и в [**1**, **5**], для построения дерева вариантов берется цикл из незапрещённых вершин.

Во время работы алгоритма выполняется обход дерева вариантов. Попутно находятся допустимые решения задачи, и запоминается лучшее из них. Лучшее найденное допустимое решение сохраняется в переменной REC, а значение целевой функции $f(REC)$.

Для каждой подзадачи P решается задача о назначениях (по аналогии с алгоритмом Балаша-Кристофидеса для задачи одного коммивояжёра). На её основе вычисляется нижняя оценка $LB(P)$ для подзадачи.

Нижняя оценка подзадачи не превосходит оптимального значения целевой функции этой подзадачи.

Если нижняя оценка для подзадачи превосходит текущее рекордное значение ($LB(P) \geq f(REC)$), то решение подзадачи не улучшит его, и подзадачу можно отбросить, сократив обход дерева вариантов и, следовательно, время работы алгоритма.

На первом шаге программы решается задача о назначениях для исходного графа, с её помощью вычисляется нижняя оценка, и соответствующая задача P_0 добавляется в очередь (строки 2–7 в листинге 1).

На очередной итерации алгоритма (строки 9–15 в листинге 1)

Шаг 1. Подзадача P с наименьшей нижней оценкой извлекается из очереди для последующей обработки (строка 9 в листинге 1).

Шаг 2. Подзадача P разбивается на дочерние подзадачи $P_1, P_2 \ldots, P_l$ согласно процедуре, указанной выше. Для каждой из подзадач находится оптимальное решение задачи о назначениях; (строка 10, внутри метода *split* в листинге 1).

Шаг 3. На основе полученных решений задачи о назначениях для каждой из подзадач $P_1, P_2 \ldots, P_l$ вычисляются нижние оценки для задачи k коммивояжёров (строка 10, в функции *split* в листинге 1).

Шаг 4. Если набор рёбер, полученных в качестве оптимального решения задачи о назначениях какой-либо из подзадач $P_1, P_2 \ldots, P_l$, образует k непересекающихся циклов, сравним максимальный из весов циклов с REC и, если нужно, обновим REC (строка 10, в функции *split* в листинге 1).

Шаг 5. Подзадачи $P_1, P_2 \ldots, P_l$ добавляются в очередь (строка 12 в листинге 1).

Шаг 6. Из очереди удаляются все подзадачи, нижняя оценка которых не меньше текущего рекордного значения (строки 14–15 в листинге 1).

Процедура повторяется до тех пор, пока в очереди остаются подзадачи.

Последний шаг позволяет очистить очередь заданий от заведомо неперспективных подзадач.

Поскольку из очереди заданий извлекаются как подзадачи с наименьшей нижней оценкой (шаг 1), так и с наибольшей нижней оценкой (шаг 6), для хранения заданий разумно использовать сбалансированное дерево поиска (например, красно-чёрное дерево), позволяющая производить эти операции за $O(\log m)$, где m – количество элементов в очереди заданий.

На шаге 4 количество циклов определяется обходом подграфа, состоящего из рёбер решения задачи о назначениях.

3. Задача о назначениях в алгоритме поиска решения задачи

Как и в алгоритме Балаша-Кристофидеса для задачи коммивояжёра (нескольких коммивояжёров) решается вспомогательная задача о назначениях [12]. Приведем некоторые понятия из этой задачи для полноты изложения и использования в описании алгоритма.

Исходный ориентированный граф представлен матрицей неотрицательных весов.

Задача 2 (о назначениях). *Найти минимальное совершенное паросочетание для двудольного неориентированного графа со взвешенными вершинами.*

Исходный граф рассматриваемой задачи – это ориентированный граф со взвешенными рёбрами, который задан матрицей весов $W = \{w_{ij}\}_{i=1..n}^{j=1..n}$. Этому графу (условно) ставится в соответствие двудольный граф:

вершины одной доли двудольного графа – это строки матрицы весов;

вершины другой доли двудольного графа – это столбцы этой же матрицы весов;

вес ребра (i, j) двудольного графа – это элемент (i, j) рассматриваемой матрицы весов исходного графа.

Решением задачи о назначениях является совершенное паросочетание, сумма весов которого минимальна. В исходном ориентированном графе из каждой вершины выходит ровно одно ребро найденного паросочетания, и в каждую вершину графа входит ровно одно ребро паросочетания. Подграф из рёбер паросочетания – это некоторый набор циклов, покрывающих все вершины графа ровно один раз.

Задача о назначениях решается для каждой подзадачи. Запрещенные рёбра при решении задачи о назначениях имеют вес $+\infty$. Если вершина

имеет замороженное исходящее ребро, она не участвует в левой доле, а если имеет замороженное входящее ребро, то не участвует в правой доле. Множество рёбер подграфа исходного графа, полученное в результате решения задачи о назначениях для подзадачи P, обозначим $AP(P)$ (Assignment Problem – задача о назначениях).

Описанное решение ищется одновременно с решением двойственной задачи, которая формулируется следующим образом.

Задача 3 (двойственная к задаче о назначениях [12]). *Найти два таких комплекта чисел* $\{u_1, u_2, \ldots, u_n\}$ *и* $\{v_1, v_2, \ldots, v_n\}$, *что*

$$w_{ij} - u_i - v_j \geq 0$$

$$S = u_1 + u_2 + \ldots + u_n + v_1 + v_2 + \ldots + v_n \to \max.$$

Иными словами, надо найти два таких комплекта чисел $\{u_1, u_2, \ldots, u_n\}$ и $\{v_1, v_2, \ldots, v_n\}$, что, если из каждой i-ой строки матрицы W вычесть u_i и из каждого j-ого столбца матрицы W вычесть v_j, то элементы новой матрицы будут неотрицательны и сумма чисел обоих комплектов будет максимальна. Если ребро (i, j) запрещено (что, как оговорено ранее, соответствует весу $+\infty$), ограничения на сумму $u_i + v_j$ не накладываются.

Сумма $S = S(P)$ показывает, что в рассматриваемой задаче k коммивояжёров не существует набора циклов, у которых сумма весов меньше S, а длина самого большого цикла не может быть меньше S/k. Таким образом, решение двойственной задачи дает в методе ветвей и границ оценку целевой функции рассматриваемой задачи. Используя эту оценку можно отсекать неперспективные ветви при обходе дерева вариантов, сокращая время выполнения алгоритма.

Обозначим

$$LB(P) = S(P)/k.$$

Как и в алгоритме Балаша-Кристофидеса для решения задачи о назначениях и двойственной к ней задачи используется венгерский алгоритм. Для решения задачи о назначениях в некотором узле дерева вариантов (некоторой подзадачи) частичное решение строится по решению задачи о назначениях узла-отца в дереве вариантов. Это позволяет находить новое решение за одну итерацию венгерского алгоритма.

4. Структура данных для подзадач

Структура данных, соответствующая подзадаче, обязана содержать не только информацию о замороженных и запрещённых рёбрах, но и

оптимальное решение прямой и двойственной задачи о назначениях (листинг 2).

```
1 struct Subproblem:
2     int LB # нижняя оценка для этой подзадачи
3     vector<int> frozen # frozen[v] хранит вершину, куда проложено
      ↪ замороженное ребро из вершины v. Если из вершины v не исходит
      ↪ замороженное ребро, то frozen[v] = -1
4     vector<vector<int>> forbidden # списки смежности, в которых хранится
      ↪ информация о замороженных рёбрах
5     vector<int> next # next[v] хранит вершину, куда  проложено ребро из v в
      ↪ оптимальном решении задачи о назначениях для текущей подзадачи
6     vector<int> pred # pred[u] хранит вершину, откуда проложено ребро в u в
      ↪ оптимальном решении задачи о назначениях для текущей подзадачи
7     vector<int> Lw, Rw # комплекты по n чисел, соответствующих оптимальному
      ↪ решению двойственной задаче к задаче о назначениях
```

Листинг 2. Структура данных Subproblem, в которой хранится информация о подзадаче

Каждая подзадача включает:

(1) текущую нижнюю оценку LB (строка 2 в листинге 2);
(2) набор замороженных рёбер (строка 3 в листинге 2);
(3) набор запрещённых рёбер (строка 4 в листинге 2);
(4) набор рёбер, найденных в качестве решения задачи о назначениях (строки 5–6 в листинге 2);
(5) комплекты чисел, соответствующих решению задачи, двойственной к задаче о назначениях (строка 7 в листинге 2).

Нижняя оценка LB необходима для очереди заданий, поскольку из неё достаются подзадачи в порядке неубывания LB. Наборы запрещённых и замороженных рёбер описывают содержание подзадачи. Наборы 4 и 5 не являются частью описания подзадачи, но нужны для дочерних подзадач, поскольку позволяют решить для них задачу о назначениях быстрее (всего за одну дополнительную итерацию венгерского алгоритма).

Множество замороженных рёбер 1 не превосходит размер задачи n. Множество запрещённых рёбер 2 ограничено лишь количеством рёбер $n(n-1)$, но на практике среднее количество запрещённых рёбер почти никогда не превосходит n. Набор 3 состоит в точности из n рёбер. Набор 4

состоит ровно из $2n$ чисел. Таким образом, на практике описание одной подзадачи из очереди занимает $O(n)$ памяти.

Структура Subproblem содержит методы (реализованные на C++)

- froze_edge(v, u). Метод замораживает ребро (v, u). Предполагается, что (v, u) входит в оптимальное решение задачи о назначениях подзадачи. Выполняется за $O(1)$. Иногда для упрощения записи для ребра e будем писать froze_edge(e).
- forbid_edge(v, u). Метод запрещает ребро (v, u). Если (v, u) не входит в оптимальное решение задачи о назначениях подзадачи, то выполняется за $O(1)$, иначе оно выполняется за $O(n^2)$, поскольку решение задачи о назначениях должно быть пересчитано применением одной итерации венгерского алгоритма.

5. Дерево вариантов и оценка целевой функции в методе ветвей и границ для поиска точного решения задачи

Узлы дерева вариантов соответствуют подзадачам исходной задачи. Для каждой такой подзадачи решается вспомогательная задача о назначениях. Один из результатов задачи о назначениях – некоторый набор циклов, покрывающих все вершины графа ровно один раз. Из этих циклов выбирается один для продолжения ветвления, то есть, для построения узлов дерева, дочерних к заданному, и соответствующих этим узлам подзадач.

Предположим, что для некоторой подзадачи уже найдено оптимальное решение задачи о назначениях, и оно уже было сверено с текущим рекордным значением.

Будем считать, что подзадача соответствует листу в дереве вариантов, если выполнено одно из двух условий:

(*1*) каждое ребро является запрещённым или замороженным;

(*2*) не существует покрытия графа циклами, которые не содержат запрещённых рёбер, и каждое замороженное ребро принадлежит какому-либо циклу.

Первое условие проверяется непосредственно. Если оно выполнено, вычисляется количество циклов, порождённых замороженными рёбрами. Если количество циклов равно k, то получено допустимое решение рассматриваемой задачи. Сравним это допустимое решение с рекордным.

Второе условие равносильно тому, что решение задачи о назначениях равно $+\infty$ для этого узла. Такое значение указывает на то, что

венгерский алгоритм был вынужден взять запрещённое ребро. Тогда узел выбрасывается из рассмотрения.

Если оба условия не выполнены, узел не является листом, и ветвление необходимо продолжить.

Найдём в решении задачи о назначениях минимальный по количеству незамороженных рёбер цикл. Пусть такой цикл включает m незамороженных рёбер $e_1, e_2, \ldots, e_m$ в порядке возрастания весов. Сведём задачу к следующим m подзадачам:

0. Ребро e_1 запрещено;
1. Ребро e_1 заморожено, ребро e_2 запрещено;
2. Рёбра e_1, e_2 заморожены, ребро e_3 запрещено;

...

$m-1$. Рёбра $e_1, e_2, \ldots, e_{m-1}$ заморожены, ребро e_m запрещено;

m. Рёбра $e_1, e_2, \ldots, e_m$ заморожены.

В последней подзадаче выбор замороженных рёбер приводит к образованию замороженного цикла. Таким образом, если исходная задача имела n вершин и k коммивояжёров, то в подзадаче останутся $n - m$ вершин и $k - 1$ коммивояжёр. Рассматривать последнюю подзадачу имеет смысл только в тех ситуациях, когда $k > 1$. Формализация этого правила представлена в листинге 3.

Функция $frozen_cycles_count$ (строка 4 в листинге 3) находит количество циклов в решении задачи о назначениях, состоящих целиком из замороженных рёбер. Это проделывается серией обходов с использованием массива $frozen$.

Функция $cycle_count$ (строка 17 в листинге 3) подсчитывает количество циклов в решении задачи о назначениях. Она реализована с помощью серии обходов с использованием массива $next$.

6. Распараллеливание программы

При выполнении алгоритма на многоядерном процессоре, можно добиться ускорения за счет распараллеливания с использованием технологии OpenMP.

При распараллеливании, как обычно, для алгоритмов, основанных на методе ветвей и границ, разные процессорные ядра получают разные подзадачи из очереди. Первоначальное решение задачи о назначениях происходит на одном ядре и выполняется сравнительно быстро относительно времени работы программы. Как только в очереди появляются

```
def split(P, REC):
    AP = AP(P) # для P задача о назначениях к этому моменту уже решена,
    ↪ обозначим за AP множество рёбер, составляющих это решение.
    C = min_cycle(AP) # по решению задачи о назначениях, которая хранится в
    ↪ P.next, находим минимальный по количеству оставшихся незамороженных
    ↪ рёбер цикл (если минимальный цикл уже состоит исключительно из
    ↪ замороженных рёбер, его не рассматриваем)
    least_salesmen = k - frozen_cycles_count(P) # находим количество циклов,
    ↪ состоящих целиком из замороженных рёбер (оно соответствует количеству
    ↪ готовых маршрутов коммивояжёров)
    # Пусть e_1, e_2, ..., e_m -- оставшиеся незамороженные рёбра в цикле C
    children = []
    for i in 0..m:
        P_child = P
        # если остался 1 коммивояжёр, и предлагается заморозить цикл целиком
        ↪ (и он при этом не проходит через все вершины графа, пропускаем
        ↪ этот случай)
        if least_salesmen = 1 and i = m and len(C) < n:
            continue
        for e in (e_1, ..., e_{i}):
        P_child.froze_edge(e) # в случае i = 0 подразумевается, что рёбра не
        ↪ замораживаются вовсе
        if i < m:
            P_child.forbid_edge(e_{i+1})
        AP_child = AP(P_child) # смотрим решение задачи о назначениях для
        ↪ P_child
        if cycle_count(AP_child) = k and f(AP_child) < f(REC):
            REC = AP_child
        children.push(P_child)
```

Листинг 3. Алгоритм ветвления (генерация дочерних подзадач).

новые подзадачи, каждое из незанятых ядер берёт одну из них. Решив подзадачу, ядро, если необходимо, обновляет рекордное значение и добавляет в очередь дочерние подзадачи. Выполнение подзадач происходит асинхронно, то есть ядру нет необходимости ожидать окончания решения подзадач другими ядрами прежде, чем он сможет взять новую подзадачу из очереди.

Все ядра используют матрицу весов W. Поскольку W не изменяется в ходе выполнения программы, конфликта данных не возникает. Для

корректной работы программы необходимы критические секции

(*1*) при записи нового рекордного значения;

(*2*) при взятии или добавлении подзадач в очередь.

7. Результаты численных экспериментов

Проведены численные эксперименты со случайными графами. Рассматривались полные ориентированные графы. Вес рёбра выбирался случайным целым неотрицательным числом, меньшим 10^6, равновероятно.

Численные эксперименты проводились на компьютере со следующими характеристиками. Windows 11, процессор Ryzen 7 5800X, 8 ядер, базовая частота 3.8 ГГц, объём L1 кэша 512 КБ, объем L2 кэша 4 мб, объём L3 кэша 32 Мб. 32 Гб оперативной памяти, DDR4, частота 3200 МГц.

Программа разработана на языке C++. Компилятор gcc 15.1, программа скомпилирована с директивой -O3.

Результаты численных экспериментов представлены в таблице 1.

Таблица 1. Среднее время (в секундах) решения рассматриваемой задачи в зависимости от количества коммивояжёров и числа вершин случайного графа. По горизонтали указано количество вершин, по вертикали указано количество коммивояжёров.

	1	2	3	4	5
30	0.001	0.007	0.058	2.138	8.30
40	0.001	0.018	0.130	7.601	234.13
50	0.002	0.025	0.468	11.727	–
60	0.002	0.053	0.675	30.019	–
80	0.003	0.149	0.85	64.59	–
100	0.003	0.314	3.715	186.6	–
150	0.008	1.876	28.404	–	–

В таблице указано среднее время решения на 100 тестах за исключением случаев, где среднее время превышало 20 секунд – для них среднее время бралось по 5 тестам. Прочерки в таблицах статьи означают, что для таких размерностей время работы программы длилось дольше 10 минут, и программа была приостановлена.

По таблице 1 и по рисунку 1 можно наблюдать экспоненциальный рост времени вычислений при росте количества вершин графа.

Обратим внимание, что описанная версия программы быстрее версии, предложенной в статье [**20**]. Это связано со следующей оптимизацией.

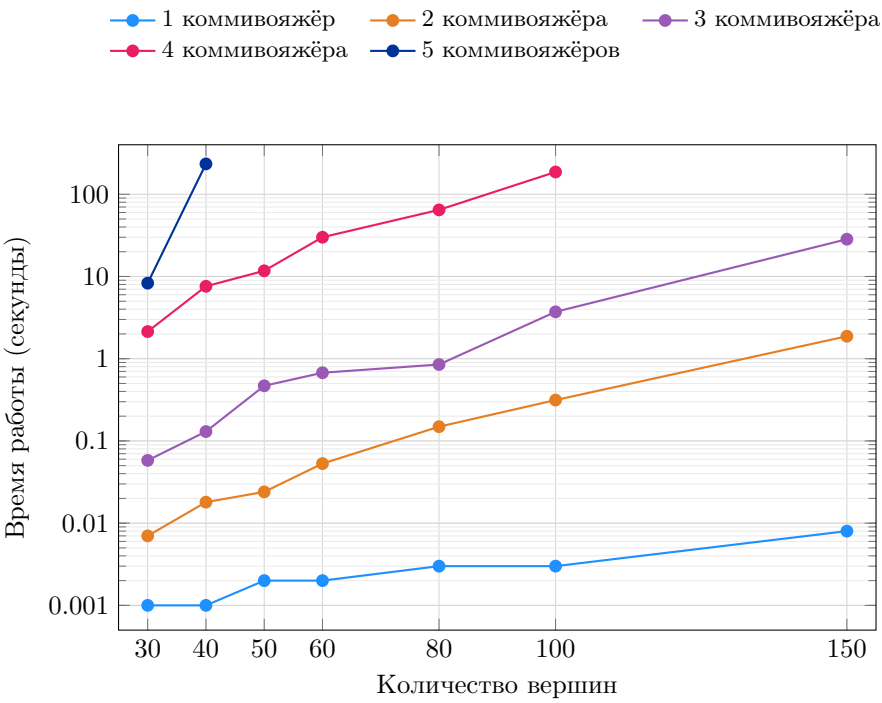


Рисунок 1. Зависимость времени работы алгоритма от количества вершин в логарифмической шкале

Выбор ребра в ходе венгерского алгоритма необходимо осуществлять среди рёбер, не являющихся запрещенными. Ранее (в [20]) набор запрещённых рёбер хранился в структуре данных «неупорядоченное множество», реализованном в стандартной библиотеке C++ на хеш-таблице. На настоящий момент запрещённые рёбра хранятся в виде списков смежности.

Наиболее близкой к рассматриваемой задаче среди NP-трудных задач является mTSP. В [16] представлен алгоритм решения задачи mTSP (несколько коммивояжёров с минимизацией суммы весов маршрутов и с депо) и численные эксперименты для программной реализации алгоритма. Численные эксперименты показали в таблице 2 производительность, которая выше производительности программы [16], более чем на порядок.

Проведён вычислительный эксперимент для точного алгоритма задачи одного коммивояжёра на одном потоке. Измерены следующие параметры:

(*1*) Время работы;

(2) Суммарное количество итераций венгерского алгоритма.

(3) Максимальное количество подзадач в очереди заданий.

Всюду взято среднее значение, полученное на 100 тестах. Результаты представлены в таблице 2. Обратим внимание, что количество итераций и максимальное количество подзадач в очереди зависят от стратегий ветвления и нижней оценки и не зависят от характеристик компьютера.

Таблица 2. Анализ (в среднем) алгоритма точного решения задачи одного коммивояжёра

n	Время работы (с)	Количество итераций	Максимальное количество подзадач в очереди
100	0.003	186	14
200	0.005	429	22
300	0.037	899	23
500	0.086	1153	27
750	0.171	1324	36
1000	0.847	2756	81
1250	1.410	3178	63
1500	1.352	2731	46
2000	4.131	4191	61
3000	14.269	6229	92

Для случайного графа с 30000 вершин точное решение задачи одного коммивояжёра работает за 2426 секунд в среднем за три испытания.

8. Приближённый алгоритм

Метод ветвей и границ позволяет решать задачу приближённо с точностью $\alpha \leq 1$. Это означает, что при значении целевой функции оптимального решения, равного OPT, будет найдено решение со значением целевой функции, не превосходящим $\dfrac{OPT}{\alpha}$.

Для получения приближённого решения достаточно задать новое, более сильное условие, при котором подзадача рассматривается:

$$LB < \alpha \cdot REC.$$

Действительно, отсечение будет дополнительно происходить в тех ситуациях, когда $\alpha \cdot REC \leq LB < REC$, хотя такие ветви и могут содержать лучшее решение (OPT). Если программа отсекла ветвь с оптимальным

решением (OPT), то для этой ветви $\alpha \cdot REC \leq LB \leq OPT$. Отсюда $REC \leq \dfrac{OPT}{\alpha}$.

Для приложений может быть достаточно приближённого решения, а более жёсткое условие отсечения позволяет отбрасывать больше ветвей, что может значительно ускорить алгоритм.

В таблицах 3 и 4 указаны результаты численных экспериментов при решении задачи k коммивояжёров с точностью 0.99 и 0.95.

Таблица 3. Точность $\alpha = 0.99$

	1	2	3	4	5
30	0.001	0.005	0.071	1.315	–
40	0.001	0.009	0.170	6.158	–
50	0.002	0.009	0.442	–	–
60	0.002	0.015	0.722	–	–
80	0.002	0.028	1.695	–	–
100	0.002	0.038	2.245	–	–
150	0.003	0.065	–	–	–
200	0.005	0.118	–	–	–

Таблица 4. Точность 0.95

	1	2	3	4	5
30	0.001	0.002	0.007	0.496	2.829
40	0.001	0.002	0.007	0.912	39.233
50	0.002	0.002	0.007	0.973	–
60	0.002	0.003	0.010	1.112	–
80	0.002	0.004	0.011	2.044	–
100	0.002	0.005	0.012	2.711	–
150	0.003	0.010	0.015	5.307	–
200	0.005	0.021	0.015	–	–
300	0.009	0.097	0.035	–	–
500	0.014	0.179	0.085	–	–
750	0.042	0.118	0.198	–	–
1000	0.103	0.205	0.479	–	–
1500	0.266	0.586	1.252	–	–
2000	0.6	1.309	2.952	–	–
3000	1.645	1.309	4.067	–	–
5000	5.069	15.291	–	–	–
50000	2200	–	–	–	–

Заключение

В работе описан способ хранения данных в алгоритме решения рассматриваемой задачи. Этот способ состоит в пометке запрещенных и разрешенных рёбер и хранении подзадач в очереди заданий. Представленные способы хранения данных позволили получить производительность программы, превосходящую известные, на полных ориентированных графах, где вес ребра берётся из равномерного распределения. В частности, при обходе дерева вариантов матрица весов графа не переписывается, а запоминаются только рассматриваемые рёбра. Это позволяет при очень больших размерностях обходиться без свопинга, как это представлено в алгоритме [5].

Алгоритм может найти применение при планировании транспортных перевозок.

Автор выражает признательность научному руководителю д.т.н. Штейнбергу Б.Я. за руководство работой.

Список использованных источников

[1] Balas E., Christofides N. *A restricted Lagrangean approach to the traveling salesman problem* // Mathematical Programming.– 1981.– Vol. **21**.– No. 1.– Pp. 19–46 .

[2] Бурховецкий В. В. *Оптимизация и распараллеливание упрощенного алгоритма Балаша–Кристофидеса для задачи коммивояжёра* // Программные системы: теория и приложения.– 2020.– Т. **11**.– № 4.– С. 3–16 .

[3] Böhm M., Friggstad Z., Mömke T., Spoerhase J. *Approximating traveling salesman problems using a bridge lemma* // *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA25 (New Orleans, Louisiana, U.S., 12–15 January 2025).– SIAM.– 2025.– Pp. 1166–1177.– ISBN: 978-1-61197-832-2 .

[4] Christofides N. *Worst-case analysis of a new heuristic for the travelling salesman problem*, Working paper 62-75-76.– Pittsburgh, PA: Graduate School of Industrial Administration, Carnegie-Mellon University.– 1976 ; Christofides N. *Worst-case analysis of a new heuristic for the travelling salesman problem* // Operations Research Forum.– 2022.– Vol. **3**.– No. 1.– id. 20.– 4 pp. .

[5] Бурховецкий В. В., Штейнберг Б. Я. *Точное и приближенное решения задачи коммивояжёра большого размера* // Вычислительные методы и программирование.– 2024.– Т. **25**.– № 4.– С. 476–482 .

[6] Little J. D. C., Murty K. G., Sweeney D. W., Karel C. *An algorithm for the traveling salesman problem* // Operations Research.– 1963.– Vol. **11**.– No. 6.– Pp. 972–989 URL.

[7] Steinberg B., Baglij A., Petrenko V., Burhovetckij V., Steinberg O., Metelica E. *An analyzer for program parallelization and optimization* // *ICAIT'2018: Proceedings of the 3rd International Conference on Applications in Information Technology* (Aizu-Wakamatsu, Japan, 1–3 November 2018), New York: ACM.– 2018.– Pp. 90–95.– ISBN: 978-1-4503-6516-1 DOI.

[8] Логунова Ю. О., Курейчик В. М. *Задача коммивояжёра. Обзор и методы её решения.*– Саарбрюккен: Palmarium Academic Publishing.– 2020.– ISBN: 6202383208.– 60 с.

[9] D. L. Applegate, R. E. Bixby, V. Chvátal, W. J. Cook *The Traveling Salesman Problem: A Computational Study*, Princeton Series in Applied Mathematics.– Princeton University Press.– 2007.– ISBN: 9781400841103.– 606 pp.

[10] Гэри М., Джонсон Д. *Вычислительные машины и трудно решаемые задачи.*– М.: Мир.– 1982.– 416 с.

[11] Feautrier P. *A parallelization framework for recursive tree programs* // *Euro-Par'98 Parallel Processing: 4th International Euro-Par Conference* (Southampton, UK, 1–4 September 1998), Lecture Notes in Computer Science.– vol. **1470**, Berlin–Heidelberg: Springer.– 2006.– Pp. 470–479.– ISBN: 978-3-540-64952-6 DOI.

[12] Kuhn H. W. *The Hungarian method for the assignment problem* // Naval Research Logistics Quarterly.– 1955.– Vol. **2**.– No. 1–2.– Pp. 83–97 DOI URL ; *50 Years of Integer Programming 1958–2008*, Berlin–Heidelberg: Springer.– 2010.– Pp. 29–47.– ISBN: 978-3-540-68274-5 DOI.

[13] Штейнберг Б. Я. *Распараллеливание рекуррентных циклов с условными операторами* // Автоматика и телемеханика.– 1995.– № 6.– С. 176–184.

[14] Посыпкин М. А., Сигал И. Х. *Комбинированный параллельный алгоритм решения задачи о ранце* // Известия Российской академии наук. Теория и системы управления.– 2008.– № 4.– С. 50–58 ✻.

[15] Bektas T. *The multiple traveling salesman problem: An overview of formulations and solution procedures* // Omega.– 2006.– Vol. **34**.– No. 3.– Pp. 209–219 DOI.

[16] Ahmeda Z. H., Al-Dayel I. *An exact algorithm for the single-depot multiple travelling salesman problem* // Ijcsns International Journal of Computer Science and Network Security.– 2020.– Vol. **20**.– No. 9.– Pp. 65–75 DOI.

[17] Ali A. I., Kennington J. L. *The asymmetric M-traveling salesmen problem: A duality based branch-and-bound algorithm* // Discrete Applied Mathematics.– 1986.– Vol. **13**.– No. 2–3.– Pp. 259–276 DOI.

[18] Laporte G., Nobert Y. *A cutting planes algorithm for the m-salesmen problem* // Journal of the Operational Research Society.– 1980.– Vol. **31**.– Pp. 1017–1023 .

[20] Клименко О. А., Штейнберг Б. Я. *Поиск точного решения задачи нескольких коммивояжёров* // Электронные библиотеки.– 2026.– Т. **29**.– № 2.– С. 414–427 .

[19] Yu W., Liu Zh. *Improved approximation algorithms for some min-max and minimum cycle cover problems* // Theoretical Computer Science.– 2016.– Vol. **654**.– Pp. 45–58 .

Поступила в редакцию 15.07.2026;
одобрена после рецензирования 06.09.2026;
принята к публикации 24.10.2026;
опубликована онлайн 02.11.2026.

Рекомендовал к публикации *д.ф.-м.н. С. М. Абрамов*

Информация об авторе:



Олег Александрович Клименко

аспирант мехмата ЮФУ; Научные интересы: параллельные вычисления, теория алгоритмов

0009-0000-5236-2415

e-mail: oleg.klimenko.math@mail.ru

Декларация об отсутствии личной заинтересованности: *благополучие автора не зависит от результатов исследования.*

PROGRAM SYSTEMS: THEORY AND APPLICATIONS ISSN 2079-3316
Research Article OPTIMIZATION METHODS AND CONTROL THEORY

UDC 004.421.2:519.712.43
10.25209/2079-3316-2026-17-4-137-158

Approximate and exact algorithms for the multiple traveling salesman problem

Oleg Aleksandrovich **Klimenko**
Southern Federal University, Rostov-on-Don, Russia

Abstract. This paper considers the multiple traveling salesman problem (TSP) without depot, which consists of finding a set of a predetermined number of disjoint cycles covering all vertices on a graph with weighted edges, such that the sum of the edge weights of the largest cycle is minimized.

An exact algorithm for solving this problem, based on the branch-and-bound method, is obtained. The resulting algorithm, like the well-known Balázs-Christofides algorithm for solving the single traveling salesman problem, utilizes the Hungarian assignment problem algorithm.

A parallel version of the resulting algorithm is developed, along with a modification that enables finding an approximate solution with a specified accuracy.

Results of numerical experiments with high-dimensional random graphs are presented. The obtained results demonstrate the superiority of the developed algorithm over known analogs. (*In Russian*).

Key words and phrases: Traveling salesman problem, assignment problem, Hungarian algorithm, branch and bound method, parallel programming, approximation algorithms

2020 *Mathematics Subject Classification*: 68W10; 68W40, 65Y20

Acknowledgments:
The work was supported by the grant № FENW-2026-0031

For citation: Oleg A. Klimenko. *Approximate and exact algorithms for the multiple traveling salesman problem*. Program Systems: Theory and Applications, 2026, **17**:4(73), pp. 137–158. (*In Russ.*).
https://psta.psiras.ru/read/psta2026_4_137-158.pdf

© Klimenko O. A. 2026

References

[1] E. Balas, N. Christofides. "A restricted Lagrangean approach to the traveling salesman problem", Mathematical Programming, **21**:1 (1981), pp. 19–46.

[2] V. V. Burxoveckij. "Optimization and parallelization of simplified Balas' and Christofides' algorithm for the traveling salesman problem", Program Systems: Theory and Applications, **11**:4 (2020), pp. 3–16 (in Russian).

[3] M. Böhm, Z. Friggstad, T. Mömke, J. Spoerhase. "Approximating traveling salesman problems using a bridge lemma", *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA25 (New Orleans, Louisiana, U.S., 12–15 January 2025), SIAM, 2025, pp. 1166–1177, ISBN: 978-1-61197-832-2.

[4] N. Christofides. *Worst-case analysis of a new heuristic for the travelling salesman problem*, Working paper 62-75-76, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA, 1976 ; N. Christofides. "Worst-case analysis of a new heuristic for the travelling salesman problem", Operations Research Forum, **3**:1 (2022), id. 20, 4 pp.

[5] V. V. Burxoveckij, B. Ya. Shtejnberg. "Exact and approximate solutions of the traveling salesman problem of large size", Vychislitel'nye metody i programmirovanie, **25**:4 (2024), pp. 476–482 (in Russian).

[6] J. D. C. Little, K. G. Murty, D. W. Sweeney, C. Karel. "An algorithm for the traveling salesman problem", Operations Research, **11**:6 (1963), pp. 972–989.

[7] B. Steinberg, A. Baglij, V. Petrenko, V. Burhovetckij, O. Steinberg, E. Metelica. "An analyzer for program parallelization and optimization", *ICAIT'2018: Proceedings of the 3rd International Conference on Applications in Information Technology* (Aizu-Wakamatsu, Japan, 1–3 November 2018), ACM, New York, 2018, pp. 90–95, ISBN: 978-1-4503-6516-1.

[8] Yu. O. Logunova, V. M. Kurejchik. *Zadacha kommivoyazhyora. Obzor i metody eyo resheniya*, Palmarium Academic Publishing, Saarbryukken, 2020, ISBN: 6202383208, 60 pp. (in Russian).

[9] Applegate D. L., Bixby R. E., Chvátal V., Cook W. J.. *The Traveling Salesman Problem: A Computational Study*, Princeton Series in Applied Mathematics, Princeton University Press, 2007, ISBN: 9781400841103, 606 pp.

[10] M. Geri, D. Dzhonson. *Vychislitel'nye mashiny i trudno reshaemye zadachi*, Mir, M., 1982, 416 pp.

[11] P. Feautrier. "A parallelization framework for recursive tree programs", *Euro-Par'98 Parallel Processing: 4th International Euro-Par Conference* (Southampton, UK, 1–4 September 1998), Lecture Notes in Computer Science, vol. **1470**, Springer, Berlin–Heidelberg, 2006, pp. 470–479, ISBN: 978-3-540-64952-6.

[12] H. W. Kuhn. "The Hungarian method for the assignment problem", Naval Research Logistics Quarterly, **2**:1–2 (1955), pp. 83–97; *50 Years of Integer Programming 1958–2008*, Springer, Berlin–Heidelberg, 2010, pp. 29–47, ISBN: 978-3-540-68274-5.

[13] B. Ya. Shtejnberg. "Rasparallelivanie rekurrentnyx ciklov s uslovnymi operatorami", Avtomatika i telemexanika, 1995, no. 6, pp. 176–184.

[14] M. A. Posypkin, I. X. Sigal. "Kombinirovannyj parallel'nyj algoritm resheniya zadachi o rance", Izvestiya Rossijskoj akademii nauk. Teoriya i sistemy upravleniya, 2008, no. 4, pp. 50–58 *.

[15] T. Bektas. "The multiple traveling salesman problem: An overview of formulations and solution procedures", Omega, **34**:3 (2006), pp. 209–219.

[16] Z. H. Ahmeda, I. Al-Dayel. "An exact algorithm for the single-depot multiple travelling salesman problem", Ijcsns International Journal of Computer Science and Network Security, **20**:9 (2020), pp. 65–75.

[17] A. I. Ali, J. L. Kennington. "The asymmetric M-traveling salesmen problem: A duality based branch-and-bound algorithm", Discrete Applied Mathematics, **13**:2–3 (1986), pp. 259–276.

[18] G. Laporte, Y. Nobert. "A cutting planes algorithm for the m-salesmen problem", Journal of the Operational Research Society, **31** (1980), pp. 1017–1023.

[20] O. A. Klimenko, B. Ya. Shtejnberg. "An algorithm for finding an exact solution to the problem of multiple travelling salesmen", Elektronnye biblioteki, **29**:2 (2026), pp. 414–427 (in Russian).

[19] W. Yu, Zh. Liu. "Improved approximation algorithms for some min-max and minimum cycle cover problems", Theoretical Computer Science, **654** (2016), pp. 45–58.