УДК 519.7

ЭФФЕКТИВНАЯ РЕАЛИЗАЦИЯ АЛГОРИТМА РЕШЕНИЯ ЗАДАЧИ КОММИВОЯЖЁРА МЕТОДОМ ВЕТВЕЙ И ГРАНИЦ

Ю. Л. Костюк

Национальный исследовательский Томский государственный университет, г. Томск, Россия

E-mail: kostyuk_y_l@sibmail.com

Для известного алгоритма точного решения задачи коммивояжёра методом ветвей и границ (алгоритма Литтла) предлагается следующая модификация. На каждой промежуточной стадии выполнения алгоритма вычисляется более точная оценка снизу для всех альтернативных маршрутов, которые можно построить на основе текущего частичного решения. Благодаря этому, отсечение бесперспективных вариантов выполняется, как правило, намного эффективнее, особенно для случайной несимметричной матрицы расстояний. Рассмотрены реализации модифицированного алгоритма с поиском вглубь и вширь, а также с поиском вглубь, когда ищется приближенный маршрут с произвольно задаваемой погрешностью. Для функции трудоёмкости $U(n) = a \cdot c^n$, измеряющей количество обрабатываемых матриц расстояний (узлов дерева решений) в графе с n вершинами, с помощью вычислительного эксперимента получены оценки констант a и c для различных реализаций алгоритма. При этом время обработки каждого узла увеличивается в 1,5–2 раза при существенном уменьшении общего времени выполнения алгоритма.

Ключевые слова: *задача коммивояжёра, метод ветвей и границ, вычислительный эксперимент.*

Введение

Как известно, задача коммивояжёра (ЗК) является NP-трудной задачей, поэтому точные алгоритмы её решения не могут иметь временную сложность меньше экспоненциальной. Решением ЗК является один из $(n-1)!$ циклов обхода по всем n вершинам полного взвешенного графа, сумма длин рёбер в котором минимальна. Такой граф обычно задаётся матрицей неотрицательных расстояний (длин рёбер графа для каждой пары вершин) размером $n \times n$, в которой диагональные элементы полагаются равными очень большой величине, теоретически бесконечности. В самом общем случае такая матрица асимметрична, а для её элементов не выполняется неравенство треугольника.

Одним из известных ранних алгоритмов точного решения ЗК для общего случая является алгоритм Литтла [1, 2], основанный на методе ветвей и границ, который строит дерево решений для перебора вариантов маршрута (циклов обхода) с отсечением. Отсекаются такие частично построенные маршруты, у которых оценка снизу длины маршрута больше или равна длине ранее построенного полного наилучшего маршрута. При построении оценки снизу на каждом этапе работы алгоритма матрица расстояний подвергается такому преобразованию с трудоемкостью $O(n^2)$, чтобы в каждой её строке и каждом столбце появился хотя бы один нуль. Более точную оценку снизу можно получать, решая задачу о назначениях на матрице расстояний за время $O(n^3)$,

при этом улучшается эффективность отсечений в дереве решений. В данной работе предлагается такое дополнительное преобразование матрицы расстояний, которое улучшает оценку снизу за время $O(n^2)$.

1. Алгоритм Литтла

Для вычисления оценки снизу длины частично построенного маршрута в алгоритме Литтла применяется следующее преобразование матрицы. Сначала из каждой строки вычитается её наименьший элемент, в результате в каждой строке образуется один (или больше) нулей. Затем из каждого столбца вычитается его наименьший элемент. Это преобразование основано на том факте, что если из любого столбца или строки матрицы вычесть константу, то стоимость оптимального маршрута уменьшается на величину этой константы, а сам маршрут остается прежним. Сумма всех вычтенных при этом величин и будет оценкой снизу для всех вариантов маршрута, строящегося по данной матрице.

Следующее действие — выбор некоторого ребра графа, такого, что все возможные варианты маршрута разбиваются на две группы: те, которые включают выбранное ребро, и те, которые его не включают, и для каждой группы создается своя матрица расстояний. Эти матрицы подвергаются аналогичному преобразованию с выбором ребра и т. д. Таким образом, строящееся при этом дерево решений получается двоичным, каждому его узлу соответствует своя матрица расстояний.

Выбор ребра на очередном шаге производится таким образом, чтобы оптимальный вариант маршрута содержал выбранное ребро с наибольшей вероятностью. Для этого просматриваются строки матрицы и среди них выделяется та, в которой второе минимальное ребро имеет наибольший вес. Затем аналогичным образом просматриваются столбцы и выделяется тот столбец, в котором второе минимальное ребро имеет наибольший вес. Окончательно выбирается такое ребро с нулевым весом, для которого вес второго минимального ребра в строке или столбце максимальный. Выбранное ребро включается в маршрут для вариантов маршрута 1-й группы и исключается из всех вариантов маршрута 2-й группы. В результате оценка снизу для всех вариантов маршрута 2-й группы увеличивается на вес второго минимального ребра в строке или столбце.

Рассмотрим более подробно реализацию алгоритма Литтла. На вход алгоритма подается квадратная матрица расстояний $\{C_{ij}\}$ размером $n \times n$, все $C_{ij} > 0$, причём диагональные элементы $C_{ii} = \infty$. Строки и столбцы матрицы помечаются отрезками маршрута, включёнными в строящийся вариант маршрута. В самом начале все эти отрезки суть отдельные вершины. Отрезок маршрута, начинающийся из вершины k и заканчивающийся в вершине i, задаётся парой чисел (k, i). После того как выбрано ребро (i, j), находящееся на пересечении строки, помеченной отрезком (k, i), и столбца, помеченного отрезком (j, l), для вариантов маршрута 1-й группы эти два отрезка объединяются. Для этого из матрицы расстояний удаляется строка (k, i) и столбец (j, l), в результате чего число строк и столбцов матрицы уменьшается на 1. Строка, помеченная отрезком, заканчивающимся в вершине l, и столбец, помеченный отрезком, начинающимся с вершины k, помечаются отрезком (k, l). Кроме того, элемент матрицы на их пересечении заменяется величиной ∞. Это делается для того, чтобы предотвратить в дальнейшем выбор ребра графа (l, k), замыкающего локальный цикл, не включающий все вершины.

Для вариантов маршрута 2-й группы выбранное ребро (i, j) в матрице заменяется величиной ∞, что предотвращает в последующем выбор этого ребра для маршрута. При этом размер матрицы не изменяется.

Когда размер матрицы уменьшится до 2×2, если при этом оценка снизу окажется меньше стоимости ранее найденного наилучшего маршрута, то строится окончательный замкнутый маршрут путем включения в него как ребра (i, j), так и ребра (l, k). Стоимость нового маршрута совпадает с его оценкой снизу, и он запоминается как наилучший маршрут среди всех ранее просмотренных.

Алгоритм Литтла просматривает строящееся таким образом дерево решений вглубь, в котором сначала делается переход к первой группе вариантов маршрута, а после их просмотра — ко второй группе. Согласно обзору [3], возможен также упорядоченный обход вширь дерева решений, когда на каждом этапе выбирается тот узел дерева решений, которому соответствует матрица с минимальной оценкой снизу. Согласно тому же обзору [3], возможна более точная оценка длины маршрута, вычисленная как решение задачи о назначениях по заданной матрице. Однако если каждое преобразование матрицы в алгоритме Литтла имеет трудоёмкость порядка $\mathrm{O}(n^2)$, то получить решение задачи о назначениях можно лишь за время $\mathrm{O}(n^3)$, используя для этого венгерский алгоритм [4]. Общая же трудоёмкость определяется произведением

1) трудоёмкости преобразования матрицы и получения оценки снизу длины маршрута;

2) количеством просматриваемых узлов дерева решений.

Так как дерево решений двоичное, а его высота обычно ненамного превышает количество n вершин графа расстояний, то количество узлов в нём имеет порядок $\mathrm{O}(2^n)$. Реальная же трудоёмкость алгоритма определяется эффективностью отсечений при просмотре дерева решений, её порядок — $\mathrm{O}(n^2 \cdot c^n)$ для алгоритма Литтла и $\mathrm{O}(n^3 \cdot c^n)$ для случая, когда оценка снизу в узлах дерева решений вычисляется как решение задачи о назначениях, при этом константа c имеет меньшую величину. Эту константу можно оценить с помощью вычислительного эксперимента.

2. Модифицированный алгоритм

Модификация алгоритма Литтла состоит в том, что при обработке алгоритмом очередного узла в дереве решений матрица расстояний подвергается дополнительному преобразованию. При вычислении оценки снизу стоимости вариантов маршрутов, строящихся из текущего узла, в матрице расстояний в каждой строке и каждом столбце появляется не менее чем по одному нулю.

Дополнительное преобразование состоит в том, что в матрице отмечается такая группа строк, в которых имеется ровно по одному нулевому элементу и все эти нули находятся в одном и том же столбце. Если такая группа, состоящая хотя бы из двух строк, найдена, то минимальный ненулевой элемент (пусть его величина равна a) в этих строках вычитается из всех строк группы и добавляется к тому столбцу, в котором находятся нули. В результате ранее полученные нули в этой группе строк сохраняются и добавляется, по крайней мере, один новый нуль. Оптимальный маршрут при этом не изменяется, а его стоимость уменьшается на величину $a(p-1)$, где p — количество строк в группе. На такую же величину увеличивается оценка снизу стоимости вариантов маршрутов, строящихся из текущего узла. Матрица просматривается до тех пор, пока не будут обработаны все такие группы строк. Далее аналогичным образом в матрице просматриваются столбцы, которые, если будут найдены подобные группы, подвергаются аналогичному преобразованию.

Такое преобразование матрицы является частным случаем преобразования, используемого в алгоритме решения задачи о назначении венгерским методом [4]. Заметим, что здесь не требуется полностью решать задачу о назначении для всей матрицы расстояний, поэтому дополнительное преобразование матрицы имеет порядок $O(n^2)$ и увеличивает время обработки одного узла дерева решений не более чем в 1,5–2 раза. Благодаря такому дополнительному преобразованию, для многих узлов дерева решений увеличивается оценка снизу стоимости вариантов маршрутов, вследствие чего отсечение ветвей дерева происходит раньше. Поэтому количество обрабатываемых узлов чаще всего будет существенно меньше, чем в известном алгоритме. В качестве иллюстрации рассмотрим решение ЗК для следующей матрицы расстояний размера 5×5, которая является частью матрицы 7×7 примера, приведённого в [2]:

	1	2	3	4	5
1	∞	3	93	13	33
2	4	∞	77	42	21
3	45	17	∞	36	16
4	39	90	80	∞	56
5	28	46	88	33	∞

Здесь дерево решений будем просматривать вглубь. В корневом узле дерева решений матрица подвергается следующей обработке. После вычитания из строк и столбцов минимальных элементов получим оценку снизу, равную 136, и матрицу

	1	2	3	4	5
1	∞	0	49	5	30
2	0	∞	32	33	17
3	29	1	∞	15	0
4	0	51	0	∞	17
5	0	18	19	0	∞

Дополнительное преобразование не изменяет эту матрицу. Ребро графа, которое здесь следует выбрать, — $(4, 3)$. В 3-м столбце второй минимальный элемент равен 19, поэтому для правого узла дерева решений оценка снизу будет не менее чем $136 + 19 = 155$. Матрица для левого узла дерева решений получается путём удаления 4-й строки и 3-го столбца, а также замене ребра $(3, 4)$ на ∞:

	1	2	4–3	5
1	∞	0	5	30
2	0	∞	33	17
4–3	29	1	∞	0
5	0	18	0	∞

Преобразование не изменяет эту матрицу. Выбирается ребро графа $(2, 1)$. Оценка снизу для правого узла: $136 + 17 = 153$. Матрица для левого узла после удаления 2-й строки и 1-го столбца и замены ребра $(1, 2)$ на ∞:

	2–1	4–3	5
2–1	∞	5	30
4–3	1	∞	0
5	18	0	∞

После вычитания из 1-й строки числа 5 и из 1-го столбца числа 1 оценка снизу 142, а матрица равна

	2–1	4–3	5
2–1	∞	0	25
4–3	0	∞	0
5	17	0	∞

Здесь в 1-й и 3-й строках имеется по одному нулю во 2-м столбце, в этих строках второй минимальный элемент равен 17, поэтому из 1-й и 3-й строк вычитается 17, а ко 2-му столбцу добавляется 17. Новая оценка снизу равна $142 + 17 = 159$, а преобразованная матрица имеет вид

	2–1	4–3	5
2–1	∞	0	8
4–3	0	∞	0
5	0	0	∞

Выбирается ребро графа $(1, 4)$, в результате два отрезка маршрута 2–1 и 4–3 объединяются в отрезок 2–1–4–3. Оценка снизу для правого узла: $159 + 8 = 167$. Матрица для левого узла после удаления 1-й строки и 2-го столбца и замены ребра $(3, 2)$ на ∞:

	2–3	5
2–3	∞	0
5	0	∞

Преобразование не изменяет эту матрицу. Так как её размер 2×2, то в маршрут добавляются рёбра $(3, 5)$ и $(5, 2)$. Наилучшее решение на этом этапе 1–4–3–5–2, его стоимость 159.

Возврат по дереву решений к матрице 3×3 даёт оценку правой ветви 167, поэтому производится ещё один возврат, к матрице 4×4, которая даёт оценку правой ветви 153, что меньше, чем 159. Матрица в правой ветви дерева решений:

	1	2	4–3	5
1	∞	0	5	30
2	∞	∞	33	17
4–3	29	1	∞	0
5	0	18	0	∞

После вычитания из 2-й строки числа 17 оценка снизу равна 153, а матрица

	1	2	4–3	5
1	∞	0	5	30
2	∞	∞	16	0
4–3	29	1	∞	0
5	0	18	0	∞

Здесь во 2-й и 3-й строках имеется по одному нулю в 4-м столбце, в этих строках второй минимальный элемент равен 1, поэтому из 2-й и 3-й строк вычитается 1, а к 4-му столбцу добавляется 1. Кроме того, в 1-м и 3-м столбцах имеется по одному нулю в 4-й строке, в этих столбцах второй минимальный элемент равен 5, поэтому из 1-го и 3-го столбца вычитается 5, а к 5-й строке добавляется 5. В результате новая оценка снизу равна $153+1+5=159$, что равно ранее полученной стоимости. Поэтому делается возврат по дереву решений к матрице 5×5, у неё для правой ветви оценка 155, а матрица после вычитания числа 19 из 3-го столбца равна

	1	2	3	4	5
1	∞	0	30	5	30
2	0	∞	13	33	17
3	29	1	∞	15	0
4	0	51	∞	∞	17
5	0	18	19	0	∞

Здесь во 2-й и 4-й строках имеется по одному нулю в 1-м столбце, в этих строках второй минимальный элемент равен 13, поэтому из 2-й и 3-й строк вычитается 13, а к 1-му столбцу добавляется 13. Новая оценка снизу равна $155+13=168$, что больше ранее полученной стоимости, поэтому делается возврат к корню дерева решений и алгоритм завершает свою работу.

На рис. 1 изображено получившееся дерево решений, в котором прямоугольники обозначают вершины, подвергнутые обработке.

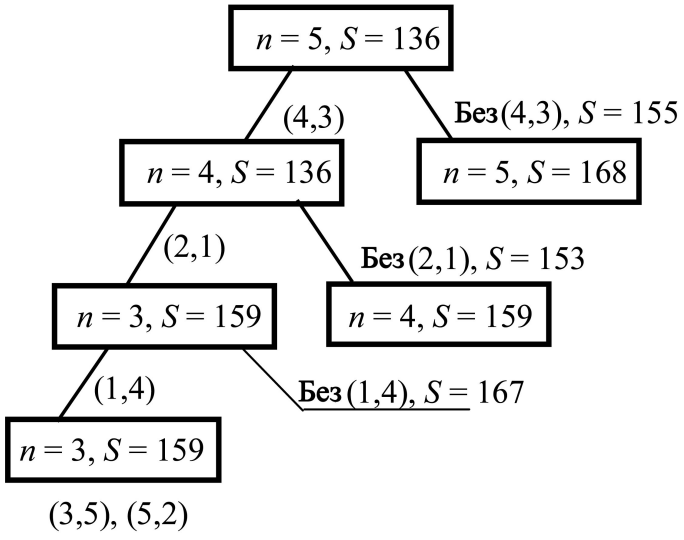

Рис. 1. Дерево решений для примера ЗК с матрицей 5×5

Таким образом, для получения оптимального маршрута 1–4–3–5–2 стоимостью 159 всего было построено и обработано 6 узлов дерева решений. Для сравнения, алгоритм Литтла получил такой же маршрут, построив и обработав 13 узлов дерева решений.

3. Реализация алгоритмов

В алгоритмах используются следующие глобальные переменные:

– стоимость $S_{\min}$ наилучшего на текущий момент построенного маршрута;

– размер n_0 исходной матрицы расстояний;

– массив P_{opt} размером n_0, в котором записан наилучший на текущий момент построенный маршрут.

Удобнее всего реализовать объектную модель, в которой каждый узел дерева решений представляется экземпляром класса со следующей структурой:

– указатель на другой узел;

– размер матрицы в узле n;

– 1-я оценка снизу S_0, равная сумме вычтенных из строк и столбцов матрицы величин;

– 2-я оценка снизу S, более точная, чем S_0 ($S \geqslant S_0$);

– матрица расстояний в узле M, её размер $n \times n$;

– два массива размером n, в которых записаны начальные и конечные вершины отрезков частично построенного маршрута;

– четыре массива размером n, в которых записаны номера нулевых элементов и вторых наименьших элементов в строках и столбцах;

– массив P размером n_0, в котором записан частично построенный маршрут, для каждого входящего в маршрут ребра (i, j): $P[i] = j$.

В алгоритме узлы дерева решений организованы в виде линейного списка, представляющего собой обход частично построенного дерева решений по висячим узлам слева направо. Первый узел в списке обрабатывается на очередном этапе, при этом возможны следующие варианты работы алгоритма:

1) маршрут достроен до конца, его вес меньше ранее построенного маршрута, новый вариант маршрута запоминается, а узел удаляется из списка;

2) оценка снизу всех маршрутов, которые могут быть построены из текущего узла, не лучше веса ранее построенного маршрута, узел удаляется из списка;

3) оценка снизу всех маршрутов, которые могут быть построены из текущего узла, меньше веса ранее построенного маршрута, тогда выбирается лучшее ребро для маршрута, создается новый первый узел с включением этого ребра, а в текущем узле это ребро удаляется. При этом проверяется оценка снизу для будущих маршрутов из текущего узла, и если она окажется не лучше веса ранее построенного маршрута, то текущий узел удаляется из списка.

Далее приведён общий вид алгоритма Литтла, реализующего поиск вглубь.

1. Задание исходного значения n_0, а также присваивание $S_{\min} = \infty$.
2. Создание начального элемента списка — корня дерева решений, в котором:
   2.1. Указателю на другой узел присваивается пустая ссылка.
   2.2. Размер матрицы в узле $n = n_0$.
   2.3. Обеим оценкам снизу S и S_0 присваиваются нули.
   2.4. Матрице расстояний в узле M задаются исходные значения.
   2.5. Каждому элементу массивов, содержащих начальные и конечные вершины отрезков маршрута, задаются значения, равные их номерам.
3. Цикл, пока список узлов дерева решений не пуст:

3.1. Если $S < S_{\min}$, то для первого узла в списке:
  3.1.1. Вычитание из строк матрицы минимальных значений с коррекцией S_0 и S.
  3.1.2. Вычитание из столбцов матрицы минимальных значений с коррекцией S_0 и S.
  3.1.3. Формирование массивов с номерами нулевых элементов и вторых наименьших элементов в строках.
  3.1.4. Формирование массивов с номерами нулевых элементов и вторых наименьших элементов в столбцах.
3.2. Если $S < S_{\min}$, то для первого узла в списке:
  3.2.1. Выбор наилучшего ребра для включения в маршрут.
  3.2.2. Если $n = 2$, то:
    3.2.2.1. Выбор второго ребра для маршрута.
    3.2.2.2. Запоминание нового маршрута в P_{opt}.
    3.2.2.3. Присваивание $S_{\min} = S_0$.
    3.2.2.4. Удаление первого узла из списка.
  3.2.2. Иначе:
    3.2.2.5. Замена выбранного ребра в матрице величиной ∞ с коррекцией S.
    3.2.2.6. Создание нового узла с размером матрицы в узле $n - 1$ и со ссылкой на предыдущий первый узел.
    3.2.2.7. Копирование матрицы в новый узел (без той строки и того столбца, где было выбранное ребро).
    3.2.2.8. Копирование с коррекцией массивов с записанными начальными и конечными вершинами отрезков маршрута.
    3.2.2.9. Копирование массива P с добавлением выбранного ребра.
    3.2.2.10. Копирование S_0 и S.
3.2. Иначе:
  3.2.3. Удаление первого узла из списка.

Модифицированный алгоритм содержит следующее дополнительное преобразование матрицы, которое необходимо вставить после п. 3.1.4:

3.1.5. Если $S < S_{\min}$, то для первого узла в списке:
  3.1.5.1. Цикл, пока не просмотрены все строки
    3.1.5.1.1. Если найдена строка с одним нулем, то:
      3.1.5.1.1.1. Цикл с поиском других строк с нулем в том же столбце.
      3.1.5.1.1.2. Если найдено две или больше строки, то коррекция строк и столбца, коррекция S_0.
  3.1.5.2. Цикл, пока не просмотрены все столбцы
    3.1.5.2.1. Если найден столбец с одним нулем, то:
      3.1.5.2.1.1. Цикл с поиском других столбцов с нулем в той же строке.
      3.1.5.2.1.2. если найдено два или больше столбца, то коррекция столбцов и строк, коррекция S_0.
  3.1.5.3. коррекция S.

Алгоритм с просмотром дерева решений вглубь можно использовать для получения не только точного решения ЗК, но и приближённого с произвольно заданной степенью погрешности, при этом он будет выполняться во много раз быстрее. Для этого в него надо внести следующие изменения. Везде, где встречается проверка «если $S < S_{\min}$», её надо заменить на «если $S(1 + \varepsilon) < S_{\min}$», где ε — величина погрешности. При этом

будет гарантировано, что стоимость найденного маршрута не более чем в $(1+\varepsilon)$ раз больше стоимости оптимального маршрута. Для рассмотренного алгоритма такие замены следует произвести в п.п. 3.1, 3.2, 3.1.5.

Алгоритм с просмотром дерева решений вширь дополнительно содержит следующие действия по упорядочению узлов дерева решений:

3.3. Если 2-й узел в списке существует, то для него:
  3.3.1. Поиск i-го узла в списке, у которого $S_i \geqslant S$.
  3.3.2. Вставка 2-го узла в список перед i-м узлом.
3.4. Если 1-й узел в списке существует, то для него:
  3.3.1. Поиск i-го узла в списке, у которого $S_i \geqslant S$.
  3.3.2. Вставка 1-го узла в список перед i-м узлом.

4. Вычислительный эксперимент

Рассмотренные алгоритмы проверены на следующих модельных данных. Элементы матрицы расстояний M_{ij} генерируются как равномерно распределённые случайные числа из диапазона от 0 до 1000, причём элементы M_{ij} и M_{ji} независимы друг от друга. Размеры матрицы n изменяются в диапазоне от 30 до 100, для каждого n строится от 3000 (для $n = 30$) до 400 (для $n = 100$) вариантов матриц, и для каждого экземпляра матрицы маршруты вычисляются (в общем случае) следующими пятью алгоритмами:

1) алгоритмом Литтла с просмотром вглубь;

2) модифицированным алгоритмом с просмотром вглубь;

3) модифицированным алгоритмом с просмотром вглубь, вычисляющим приближённое решение с $\varepsilon = 0{,}05$;

4) модифицированным алгоритмом с просмотром вширь;

5) модифицированным алгоритмом с просмотром вширь, но с начальным приближением, полученным алгоритмом 3.

Все алгоритмы написаны на языке Паскаль в системе Delphi, вычисления производились на компьютере с процессором Atlon, имеющим кэш-память по 1 Мбайт на каждое процессорное ядро и работающим на частоте 2,9 ГГц. В табл. 1 представлены среднее количество обрабатываемых узлов дерева решений и среднее время вычисления маршрута, полученные по одним и тем же сгенерированным экземплярам матриц для всех пяти алгоритмов. Отсутствуют только те результаты вычислений, для получения которых потребовалось бы слишком много времени.

Кроме того, подсчитана величина разброса (как среднеквадратичное отклонение) для количества узлов дерева решений и затраченного времени. Величина разброса оказалась очень большой для всех случаев, для максимальных размерностей она в 1,5–2 раза превышает соответствующие средние значения. Это является следствием того, что ЗК является NP-трудной, поэтому для наихудших (с точки зрения алгоритмов) экземпляров матрицы количество обработанных узлов может быть больше среднего количества в десятки раз. Из табл. 1 видно, что алгоритму с просмотром вширь требуется обработать несколько меньше узлов дерева решений, чем аналогичному алгоритму с просмотром вглубь, однако на уменьшении времени работы это сказывается лишь при n, меньших 50. При больших n время работы резко увеличивается, при $n = 80$ оно более чем в 10 раз превышает время работы алгоритма с просмотром вглубь. При этом не помогает задание хорошего начального приближения перед выполнением алгоритма с просмотром вширь. Это происходит из-за того, что в алгоритме с просмотром вширь максимальный размер списка узлов дерева решений растёт экспоненциально с ростом n, и когда размер используемой алгоритмом памяти начинает

Таблица 1

Результаты эксперимента

Размер матрицы n	Алгоритм 1		Алгоритм 2		Алгоритм 3		Алгоритм 4		Алгоритм 5	
	Сред. кол-во узлов	Сред. время, с	Сред. кол-во узлов	Сред. время, с	Сред. кол-во узлов	Сред. время, с	Сред. кол-во узлов	Сред. время, с	Сред. кол-во узлов	Сред. время, с
30	999	0,0156	163	0,0039	60	0,0019	118	0,0035	118	0,0034
40	5807	0,134	431	0,0148	115	0,0045	315	0,0131	315	0,0131
50	31506	1,051	980	0,0472	191	0,0102	724	0,0475	724	0,0450
60	189022	8,241	2421	0,160	390	0,029	1847	0,241	1847	0,186
70	–	–	5277	0,459	728	0,071	4076	1,454	4076	1,149
80	–	–	13797	1,481	1485	0,175	11100	18,982	11100	11,159
90	–	–	33504	4,562	2589	0,391	–	–	–	–
100	–	–	89043	14,679	8478	1,602	–	–	–	–

существенно превышать размер кэш-памяти в процессоре, происходит интенсивный обмен данными между кэш-памятью и основной памятью компьютера. В то же время максимальный размер списка узлов дерева решений для алгоритма с просмотром вглубь не превышает n, и весь список обычно помещается в кэш-память. Трудоёмкость по количеству обрабатываемых узлов дерева решений всех алгоритмов приближённо имеет порядок $\mathrm{O}(c^n)$, а трудоёмкость по времени — $\mathrm{O}(n^2 \cdot c^n)$, где константа c имеет разную величину для разных алгоритмов. Оценим величину константы c по методу наименьших квадратов. Количество обрабатываемых узлов U как функция от n:

$$U(n) = a \cdot c^n, \tag{1}$$

где a и c — константы.

Чтобы избавиться от константы a, вместо функции $U(n)$ будем рассматривать отношение двух значений этой функции от разных значений n:

$$U(n_2)/U(n_1) = c^{n_2 - n_1}. \tag{2}$$

После логарифмирования равенства (2) можно записать сумму квадратов разностей измеренных и вычисленных $U(n)$, причём отношение будем записывать для пар соседних измерений:

$$\sum_i \left(\ln \frac{U_i}{U_{i-1}} - (n_i - n_{i-1}) \ln c \right)^2 \to \min_{\ln c}. \tag{3}$$

После дифференцирования (3) по параметру $\ln c$ и приравнивания производной нулю получим выражение для оценки c:

$$c = \exp \left(\sum_i \ln \frac{U_i}{U_{i-1}} (n_i - n_{i-1}) / \sum_i (n_i - n_{i-1})^2 \right). \tag{4}$$

Для оценивания константы a прологарифмируем (1) и запишем сумму квадратов разностей на основе измеренных и вычисленных $U(n)$:

$$\sum_i \left(\ln U_i - \ln a - n_i \cdot \ln c\right)^2 \to \min_{\ln a}. \tag{5}$$

После дифференцирования (5) по параметру $\ln a$ и приравнивания производной нулю получим выражение для оценки a:

$$a = \exp\left(\frac{1}{K}\sum_{i=1}^{K}(\ln U_i - n_i \cdot \ln c)\right). \tag{6}$$

Вычисленные по формулам (4) и (6) оценки параметров a и c для количества обрабатываемых узлов дерева решений (1) у рассмотренных алгоритмов сведены в табл. 2.

Таблица 2

Оценки параметров a и c

Параметр	Алгоритм 1	Алгоритм 2	Алгоритм 3	Алгоритм 4	Алгоритм 5
a	5,238	10,665	5,811	7,725	7,725
c	1,191	1,094	1,073	1,095	1,095

Одной из особенностей многих точных алгоритмов решения NP-трудных задач является то, что после нахождения оптимального решения алгоритм ещё долго перебирает другие варианты решения, пытаясь улучшить найденное. Поэтому представляет интерес выяснить, какова доля времени, затрачиваемого на поиск оптимального маршрута, по сравнению с общим временем вычислений. Оценки среднего значения этого параметра для алгоритмов 1, 2 и 3 приведены в табл. 3. Алгоритмы 4 и 5 реализуют поиск вширь таким образом, что самый первый из найденных вариантов является оптимальным маршрутом, поэтому для них эта доля времени близка к единице.

Таблица 3

Доля времени, затрачиваемого на поиск оптимального маршрута

Размер матрицы n	Алгоритм 1	Алгоритм 2	Алгоритм 3, $\varepsilon = 0{,}05$
30	0,51	0,58	0,72
40	0,39	0,46	0,56
50	0,37	0,35	0,36
60	0,38	0,37	0,32
70	–	0,31	0,33
80	–	0,26	0,29
90	–	0,26	0,22
100	–	0,29	0,23

Для алгоритма 3, вычисляющего приближенное решение ЗК с гарантированной точностью ε, представляет интерес, какова при этом средняя погрешность полученного решения и во сколько раз время вычислений меньше, чем у алгоритма 2. Оценки этих величин представлены в табл. 4 для $\varepsilon = 0{,}05$, 0,1 и 0,2.

Таблица 4

Характеристики алгоритма 3

Размер матрицы n	$\varepsilon = 0{,}05$		$\varepsilon = 0{,}1$		$\varepsilon = 0{,}2$	
	Средняя погрешность	Доля времени вычислений	Средняя погрешность	Доля времени вычислений	Средняя погрешность	Доля времени вычислений
30	0,012	0,50	0,022	0,39	0,041	0,31
40	0,013	0,31	0,027	0,21	0,045	0,15
50	0,016	0,22	0,030	0,12	0,051	0,08
60	0,016	0,18	0,032	0,09	0,055	0,05
70	0,018	0,15	0,033	0,053	0,053	0,026
80	0,018	0,12	0,033	0,049	0,052	0,015
90	0,018	0,085	0,035	0,050	0,052	0,014
100	0,018	0,108	0,036	0,038	0,052	0,014

Следует отметить, что результаты проведённого вычислительного эксперимента справедливы лишь для случая несимметричной матрицы расстояний с равномерным распределением длин рёбер. Для других распределений рассмотренные алгоритмы имеют, как правило, намного большую трудоёмкость. Так, для известной ЗК с 48 городами [5], с симметричной матрицей, алгоритм 1 получил результат за 3,5 ч работы, а алгоритм 2 — за 1 ч 45 мин. Интересно, что при этом получен маршрут длиной 11461, что меньше, чем в [5], где длина маршрута указана равной 11470. Алгоритмы 1 и 2 вычислили следующий оптимальный маршрут:
1 2 48 15 43 21 33 30 23 9 10 40 36 34 6 8 47 7 38 14 18 12 22 13 28 32 25 3 5 29 26 41 24 35 17 31 20 11 16 42 4 46 45 39 44 27 37 19.

Программа, реализующая алгоритм 2, включая модуль с описанием класса для решения ЗК, доступна в сети Интернет по ссылке [6].

Заключение

Как показал вычислительный эксперимент, предложенная модификация алгоритма Литтла для случайной матрицы расстояний позволяет существенно сократить количество обрабатываемых узлов дерева решений при поиске оптимального маршрута в задаче коммивояжёра. При этом время обработки каждого узла имеет тот же порядок трудоёмкости, $O(n^2)$, но с увеличенным в 1,5–2 раза множителем. В целом, трудоёмкость обоих алгоритмов имеет порядок $O(n^2 \cdot c^n)$, но у модифицированного алгоритма константа c существенно меньше, что позволяет при примерно одинаковом времени решать задачу для $n = 100$ против $n = 60$. Сравнение реализаций модифицированного алгоритма с поиском вглубь и вширь показало незначительное преимущество поиска вширь только для небольших размерностей ($n < 50$), когда объём требуемой памяти для размещения матриц в узлах дерева решений не слишком велик. Для $n \geqslant 50$ время

работы алгоритма с поиском вширь резко увеличивается из-за ограниченного объёма кэш-памяти.

ЛИТЕРАТУРА

1. *Little J. D. C., Murty K. G., Sweeney D. W., and Karel C.* An algorithm for the Traveling Salesman Problem // Operations Research. 1963. No. 11. P. 972–989.
2. *Рейнгольд Э., Нивергельт Ю., Део Н.* Комбинаторные алгоритмы. Теория и практика: пер. с англ. М.: Мир, 1980. 478 с.
3. *Меламед И.Й., Сергеев С. И., Сигал И. Х.* Задача коммивояжёра. Точные алгоритмы // Автоматика и телемеханика. 1989. № 10. С. 3–29.
4. *Данциг Дж.* Линейное программирование, его применения и обобщения: пер. с англ. М.: Прогресс, 1966.
5. *Хелд М., Карп Р. М.* Применения динамического программирования к задачам упорядочивания // Кибернетический сборник. Вып. 9, старая серия. М.: Мир, 1964. С. 202–218.
6. *Костюк Ю. Л.* Модифицированный алгоритм решения задачи коммивояжёра методом ветвей и границ. Программа и модуль с описанием класса: язык Паскаль в системе Delphi // Электронный ресурс: `www.inf.tsu.ru/Decanat/Staff.nsf/people/KostjukJuL`, 2013.