УДК 519.7

ЗАДАЧА КОММИВОЯЖЁРА: УЛУЧШЕННАЯ НИЖНЯЯ ГРАНИЦА В МЕТОДЕ ВЕТВЕЙ И ГРАНИЦ

Ю. Л. Костюк

Национальный исследовательский Томский государственный университет, г. Томск, Россия

E-mail: kostyuk_y_l@sibmail.com

Рассматривается решение задачи коммивояжёра методом ветвей и границ. Предлагается способ дополнительного (по сравнению с алгоритмом Литтла) уточнения нижней границы, особенно эффективный для случая симметричной матрицы, и на его основе строится новый алгоритм. С помощью вычислительного эксперимента получены оценки констант в формулах трудоёмкости для трёх модификаций алгоритма Литтла на трёх видах случайных матриц расстояний: 1) несимметричных матрицах со случайными расстояниями; 2) матрицах с евклидовыми расстояниями между случайными точками внутри квадрата; 3) несимметричных матрицах со случайными расстояниями, удовлетворяющими неравенству треугольника.

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

Введение

В известном алгоритме Литтла [1], реализующем метод ветвей и границ для задачи коммивояжёра (ЗК), на каждом шаге обработки матрицы расстояний из каждой строки и столбца вычитается величина наименьшего в строке или столбце элемента, в результате чего в каждой строке и каждом столбце появляется не менее чем по одному нулю. Сумма величин всех вычтенных элементов даёт нижнюю границу для оптимального решения, на основе которой производится отсечение бесперспективных вариантов решения.

В [2] описано дополнительное преобразование матрицы, являющееся частным случаем преобразования в венгерском алгоритме решения задачи о назначениях [3]. В результате в матрице появляются дополнительные нули и увеличивается нижняя граница оптимального маршрута, что улучшает процесс отсечения. Как показал вычислительный эксперимент [2], алгоритм с таким преобразованием работает гораздо быстрее в случае несимметричной матрицы, однако для симметричной матрицы это не так.

1. Вычисление улучшенной нижней границы оптимального решения задачи коммивояжера

При решении ЗК и задачи о назначениях применяется одно и то же преобразование матрицы. Преобразование основано на том факте, что если из любого столбца или строки матрицы вычесть (или прибавить) константу, то стоимость оптимального маршрута коммивояжёра и стоимость решения задачи о назначениях уменьшается (увеличивается) на величину этой константы, а само решение остается прежним.

В алгоритме Литтла [1] из каждой строки и столбца выполняется вычитание наименьших в строках и столбцах элементов с целью получения в каждой строке и каждом столбце не менее чем по одному нулю. В модифицированном алгоритме [2] после

этого находятся такие группы строк (столбцов), в которых имеется по одному нулю в одном и том же столбце (строке), и для каждой такой группы вычитается минимальный ненулевой элемент, а к столбцу (строке), где находятся нули, добавляется такое же значение. В результате увеличивается количество нулей в матрице и растёт нижняя граница стоимости оптимального маршрута. Трудоёмкость такого преобразования остаётся в пределах $O(n^2)$, как и для преобразования в алгоритме Литтла, но с увеличенным множителем.

В венгерском алгоритме [3] похожий процесс получения дополнительных нулей в матрице продолжается ещё дальше, до тех пор, пока из полученных нулей не удаётся построить подмножество независимых нулей, когда в каждой строке и каждом столбце имеется ровно по одному независимому нулю. В результате нижняя граница оптимального маршрута может дополнительно увеличиться. Однако трудоёмкость этого действия имеет порядок $O(n^3)$.

Все эти преобразования матрицы позволяют в лучшем случае получить нижнюю границу стоимости оптимального маршрута коммивояжёра, равную стоимости решения задачи о назначениях. Рассмотрим дальнейшее уточнение нижней границы стоимости оптимального маршрута коммивояжёра.

Рассмотрим ориентированный невзвешенный граф, в котором вершины те же, что и в исходной ЗК, а рёбра соответствуют нулевым элементам преобразованной матрицы. В этом графе просмотром вглубь выделим сильносвязные компоненты [4], трудоёмкость такого действия — $O(n^2)$. Для сильносвязных компонент справедливо следующее свойство: из любой вершины можно перейти в любую другую вершину внутри той же самой компоненты по рёбрам с нулевым весом. Если нулевые элементы в матрице получены венгерским алгоритмом, то каждая из компонент будет состоять не менее чем из двух вершин. После преобразования матрицы модифицированным алгоритмом большая часть компонент будет содержать по две или больше вершин, но в отдельных компонентах может содержаться одна вершина. В любом случае цикл по всем вершинам исходной матрицы, т. е. маршрут коммивояжёра, будет, кроме нулевых рёбер, содержать также рёбра перехода между компонентами, часть этих рёбер перехода будут нулевыми, а часть — ненулевыми.

Чтобы получить более точную нижнюю границу маршрута коммивояжёра, к ранее вычисленной границе необходимо добавить нижнюю границу суммарной длины рёбер перехода между компонентами, чтобы получился единый маршрут. Для этого каждую из выделенных сильносвязных компонент будем считать супервершиной, т. е. вершиной другого графа — графа компонент. Для каждой пары (i, j) супервершин вычислим ориентированное расстояние как длину минимального ребра между парами вершин (p, q) матрицы ЗК, где вершина p принадлежит i-й компоненте, а вершина q — j-й компоненте. Трудоёмкость такого действия не превышает $O(n^2)$ при любом количестве компонент. Однако рёбра полученной матрицы расстояний графа компонент между супервершинами ещё не являются кратчайшими расстояниями перехода из одной компоненты в другую, так как в оптимальном маршруте вершины исходной матрицы, принадлежащие одной и той же компоненте, могут идти не подряд, а перемежаться с вершинами из других компонент. Поэтому надо из матрицы расстояний графа компонент получить матрицу кратчайших расстояний. Это можно сделать либо алгоритмом Флойда, либо k-кратным применением алгоритма Дейкстры [4], где k — количество компонент. В обоих случаях трудоёмкость такого действия — $O(k^3)$.

После этого полученную матрицу кратчайших расстояний подвергнем такому же преобразованию, как исходную матрицу, тогда сумма вычтенных элементов даст до-

полнительную положительную поправку к нижней границе стоимости оптимального маршрута коммивояжёра.

На втором этапе выполним такие же действия с преобразованной матрицей кратчайших расстояний между супервершинами. Для этого выделим сильносвязные компоненты супервершин, построим по ним матрицу расстояний для суперсупервершин и вычислим на ней кратчайшие расстояния. После преобразования последней матрицы получим ещё одну дополнительную поправку к нижней границе.

Аналогичные действия будем продолжать на третьем, четвертом и т. д. этапах до тех пор, пока очередная дополнительная поправка к нижней границе не станет равной нулю.

Оценим трудоёмкость всего этого процесса. На каждом этапе уточнения нижней границы при обработке матрицы размером n выполняются следующие действия:

1) получение нулей в каждой строке и каждом столбце матрицы: трудоёмкость $\mathrm{O}(n^2)$;
2) выделение сильно связных компонент: трудоёмкость $\mathrm{O}(n^2)$;
3) построение матрицы расстояний супервершин: трудоёмкость $\mathrm{O}(n^2)$;
4) вычисление матрицы кратчайших расстояний для супервершин: трудоёмкость $\mathrm{O}(k^3)$, где k — количество компонент.

Если каждая компонента связности содержит не менее двух вершин, то $k \leqslant n/2$, поэтому в худшем случае общая трудоёмкость вычисления улучшенной нижней границы на всех этапах не превысит суммы

$$T(n) \leqslant a\left(n^2 + (n/2)^2 + (n/4)^2 + \ldots\right) + b\left((n/2)^3 + (n/4)^3 + \ldots\right) \leqslant \frac{4}{3}an^2 + \frac{1}{7}bn^3, \quad (1)$$

где a и b — константы.

В алгоритме, реализующем метод ветвей и границ для ЗК, описанный процесс вычисления нижней границы производится на каждом шаге, как только выбрано очередное ребро для маршрута, т. е. нижняя граница вычисляется с учётом того, что часть рёбер уже включена в маршрут.

2. Дополнительные усовершенствования в реализации алгоритмов

В [2] подробно изложена реализация алгоритма Литтла и модифицированного алгоритма. В данной работе рассмотрим некоторые дополнительные усовершенствования в реализации обоих этих алгоритмов при поиске вглубь на дереве решений. Эти усовершенствования применимы также в случае вычисления на каждом этапе алгоритма улучшенной нижней границы, так, как это описано выше. Предлагаются следующие усовершенствования.

1. В обоих алгоритмах на очередном шаге, после преобразования матрицы, выбирается нулевой элемент M_{ij} в матрице для включения соответствующего ребра в маршрут. Критерий выбора: максимальное значение второго элемента в строке или столбце. После этого из обработанной создается новая матрица, в которой удалена i-я строка и j-й столбец, а в старой матрице элемент M_{ij} заменяется на ∞. Дополнительным улучшением здесь является то, что попутно для обеих матриц оценивается изменение нижней границы всех вариантов маршрута, которые можно построить по этим матрицам.

2. Когда при построении варианта маршрута на очередном шаге матрица уже имеет размер 4×4, то после выбора еще одного ребра новая матрица будет иметь размер

3×3. Для такой матрицы возможно лишь два варианта замыкания полного маршрута. В этом случае можно легко оценить, какой из этих вариантов лучше и имеет ли он меньшую длину, чем ранее найденный маршрут, и для этого не требуется делать полного преобразования новой матрицы.

3. Длина списка всех узлов дерева решений, т. е. матриц, на любом промежуточном этапе просмотра вглубь равна $n - 2$, где n — размер исходной матрицы, так как обработка заканчивается на матрице размером 3×3. При этом размер матрицы в первом узле — n, во втором — $(n - 1)$, в третьем — $(n - 2)$ и т. д. Поэтому целесообразно выделить память для такого списка из $(n - 2)$-х узлов в самом начале, а в процессе выполнения алгоритма копировать новые матрицы в уже существующие узлы.

4. Если первоначальная матрица была симметричной, то для оптимального маршрута коммивояжёра всегда существует равный ему по длине маршрут в обратном направлении. Поэтому, когда для такой матрицы выбирается самое первое ребро M_{ij} для включения в маршрут, то в старой матрице необходимо заменить на ∞ не только элемент M_{ij}, но и элемент M_{ji}, чтобы сразу предотвратить попытку формирования маршрута в противоположном направлении. При этом надо учесть, что при выборе второго и последующих рёбер матрица перестаёт быть симметричной.

Как показал вычислительный эксперимент, в совокупности все эти усовершенствования позволяют уменьшить время выполнения алгоритмов на 20–30 %, а для симметричной матрицы — до 60 %.

3. Реализация алгоритмов

Все три алгоритма (алгоритм Литтла, модифицированный алгоритм, а также модифицированный алгоритм с вычислением улучшенной нижней границы) реализованы со всеми рассмотренными выше усовершенствованиями. 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.
- массив P_t размером n_0, в котором записан частично построенный маршрут: $P_t[i] = j$ для каждого входящего в маршрут ребра (i, j).

Каждый узел дерева решений представляется экземпляром класса со следующей структурой:

- указатели на соседние узлы, на левый и правый узел;
- размер матрицы в узле n;
- первая нижняя граница S_0, равная сумме вычтенных из строк и столбцов матрицы величин;
- вторая нижняя граница S_m, более точная, чем S_0 ($S_m \geqslant S_0$);
- матрица расстояний в узле M, её размер $n \times n$;
- признак симметричности матрицы;
- 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.

В алгоритме узлы дерева решений сцепляются в линейный список, представляющий собой обход частично построенного дерева решений по висячим узлам слева

направо. Узлы в списке обрабатываются, начиная с корневого узла. После обработки текущего узла производится переход к левому или правому узлу в списке.

Далее приведён общий вид модифицированного алгоритма с вычислением улучшенной нижней границы. Алгоритм реализует поиск вглубь на дереве решений.

1. Выделение памяти для начального элемента списка — корневого узла дерева решений — размером $n = n_0$. Присваивание начальных значений для матрицы в корневом узле, вспомогательным массивам и глобальным переменным.
2. Проверка матрицы на симметричность и задание значения признаку симметричности.
3. Выделение памяти для списка узлов дерева решений с присоединением узлов слева от корневого узла и заданием другим узлам размеров соответственно: $n-1$, $n-2$, $\ldots$, 3.
4. Цикл, пока список узлов дерева решений не просмотрен до конца:

4.1. Если $S < S_{\min}$, то для текущего узла в списке:

4.1.1. Вычитание из строк и столбцов матрицы минимальных значений с коррекцией S_0 и S_m.

4.1.2. Если $S < S_{\min}$, то для текущего узла в списке:

4.1.2.1. Дополнительное преобразование матрицы с коррекцией S_0 и S_m.

4.1.2.2. Если $S < S_{\min}$, то для текущего узла в списке:

4.1.2.2.1. Если $n > 4$, то вычисление улучшенной нижней границы с коррекцией S_m.

4.1.2.2.2. Если $S < S_{\min}$, то для текущего узла в списке:

4.1.2.2.2.1. Выбор наилучшего ребра для включения в маршрут.

4.1.2.2.2.2. Формирование матрицы размером $n-1$ в левом узле списка.

4.1.2.2.2.3. Коррекция матрицы в старом узле с учетом симметричности.

4.1.2.2.2.4. Если $n = 4$, то:

4.1.2.2.2.4.1. Выбор лучшего варианта замыкания маршрута в новой матрице, и если его длина меньше $S_{\min}$, то запоминание нового маршрута.

4.1.2.2.2.4. Иначе текущий узел слева.

4.1.2.2.2. Иначе текущий узел справа.

4.1.2.2. Иначе текущий узел справа.

4.1.2. Иначе текущий узел справа.

4.1. Иначе текущий узел справа.

5. Уничтожение списка узлов дерева решений.

4. Вычислительный эксперимент

В вычислительном эксперименте на трёх видах модельных данных проверялась сравнительная трудоёмкость следующих трех алгоритмов:

1) алгоритма Литтла с усовершенствованиями;

2) модифицированного алгоритма с усовершенствованиями;

3) модифицированного алгоритма с усовершенствованиями и с вычислением улучшенной нижней границы.

Все алгоритмы написаны на языке Паскаль в системе Delphi, вычисления производились на компьютере с процессором Atlon, работающем на частоте 2,9 ГГц.

В табл. 1–3 представлены среднее количество обрабатываемых узлов дерева решений и среднее время вычисления маршрута, полученные по одним и тем же сгенерированным экземплярам матриц для всех трёх алгоритмов для следующих трёх видов распределений элементов матриц:

1) случайная матрица, элементы в которой — равномерно распределённые независимые случайные целые числа из диапазона от 0 до 1000, матрица несимметричная;
2) матрица евклидовых расстояний для n равномерно распределённых независимых случайных точек на плоскости с координатами из диапазона от 0 до 1000; вычисляются расстояния между ними, матрица симметричная;
3) случайная матрица с неравенством треугольника, на которой вычислены кратчайшие расстояния алгоритмом Флойда, матрица несимметричная.

Для каждого размера матриц n генерировалось от 4000 (для минимальных n) до 400 (для максимальных n) вариантов матриц. В таблицах отсутствуют те результаты вычислений, для получения которых потребовалось бы слишком много времени.

Из табл. 1 видно, что для случайных матриц модифицированный алгоритм имеет существенное преимущество над алгоритмом Литтла, при $n = 70$ его время работы в 100 раз меньше. Но вычисление в нём улучшенной нижней границы не даёт дополнительного уменьшения времени. Это происходит из-за того, что у таких матриц часто получается всего одна сильносвязная компонента и нижняя граница не улучшается.

Таблица 1

Результаты эксперимента для случайных матриц

Размер матрицы n	Алгоритм 1		Алгоритм 2		Алгоритм 3	
	Среднее кол-во узлов	Среднее время, с	Среднее кол-во узлов	Среднее время, с	Среднее кол-во узлов	Среднее время, с
30	926	0,0078	152	0,0023	143	0,0026
40	5610	0,079	386	0,0100	361	0,0112
50	29224	0,619	977	0,0351	904	0,0376
60	156106	4,638	2252	0,113	2093	0,119
70	966825	38,61	5573	0,374	5189	0,398
80	–	–	15056	1,326	14176	1,360
90	–	–	27276	3,06	25300	3,11
100	–	–	85623	12,06	79236	12,19
110	–	–	181540	31,47	165508	31,09

Из табл. 2 видно, что для матриц евклидовых расстояний модифицированный алгоритм уступает алгоритму Литтла, его время работы больше в 2–4 раза. На таких матрицах вычисление улучшенной нижней границы даёт наибольший эффект: при $n = 30$ время работы модифицированного алгоритма с вычислением улучшенной нижней границы в 15 раз меньше, чем у алгоритма Литтла. Здесь количество сильносвязных компонент нередко оказывается близким к $n/2$, в результате уточнённая нижняя граница становится заметно больше, чем в алгоритме Литтла, что приводит к улучшению процесса отсечения. Следует также отметить, что матрицы евклидовых расстояний оказались самыми трудными для всех рассматриваемых алгоритмов.

Из табл. 3 видно, что для случайных матриц с неравенством треугольника модифицированный алгоритм имеет существенное преимущество над алгоритмом Литтла, при $n = 45$ его время работы в 20 раз меньше. Но вычисление в нём улучшенной нижней границы хотя и приводит к некоторому уменьшению количества обрабатываемых узлов, но в целом на 10–20 % увеличивает общее время работы.

Следует также заметить, что разброс количества обрабатываемых узлов дерева решений и времени вычисления для матриц одинакового размера и одного и того же вида распределения оказался очень большим, разница нередко достигала многих сотен раз.

Таблица 2

Результаты эксперимента для матриц евклидовых расстояний

Размер матрицы n	Алгоритм 1		Алгоритм 2		Алгоритм 3	
	Среднее кол-во узлов	Среднее время, с	Среднее кол-во узлов	Среднее время, с	Среднее кол-во узлов	Среднее время, с
15	793	0,0017	1008	0,0039	174	0,0023
20	14744	0,0404	25846	0,1220	949	0,0183
25	165027	0,578	407836	2,563	5010	0,133
30	4734542	13,0	3152506	22,8	23401	0,842
35	–	–	–	–	117408	5,51
40	–	–	–	–	888505	71,9

Таблица 3

Результаты эксперимента для случайных матриц с неравенством треугольника

Размер матрицы n	Алгоритм 1		Алгоритм 2		Алгоритм 3	
	Среднее кол-во узлов	Среднее время, с	Среднее кол-во узлов	Среднее время, с	Среднее кол-во узлов	Среднее время, с
25	5043	0,0168	873	0,0042	484	0,0040
30	35809	0,132	2624	0,0134	1924	0,0144
35	351811	1,23	7348	0,0458	5976	0,0510
40	631431	3,03	33528	0,180	28153	0,236
45	4221193	21,3	135320	0,86	131190	1,08
50	–	–	2056272	12,8	1803337	15,0

Это является следствием того, что ЗК относится к NP-трудным задачам. Из-за этого, к сожалению, для произвольной матрицы невозможно гарантировать, что время решения на ней ЗК не превысит заранее рассчитанной величины. Поэтому данные в таблицах следует рассматривать как ориентировочные.

Трудоёмкость по количеству обрабатываемых узлов дерева решений у всех алгоритмов приближённо имеет порядок $O(c^n)$, где параметр c имеет разную величину для различных алгоритмов и распределений. При этом трудоёмкость по времени имеет порядок $O(n^2c^n)$ для первого и второго алгоритмов. Согласно формуле (1), у третьего алгоритма (с вычислением улучшенной нижней границы) для различных распределений трудоёмкость по времени в общем случае имеет порядок $O((an^2 + bn^3)c^n)$, где a, b — постоянные параметры. Однако, как показал вычислительный эксперимент, для рассмотренных распределений при относительно небольших n параметр b во много раз меньше a, и им можно пренебречь. Поэтому для всех алгоритмов и распределений можно считать, что трудоёмкость по времени имеет порядок $O(n^2c^n)$.

Оценим величины параметров в формулах трудоёмкости по методу наименьших квадратов. Количество обрабатываемых узлов U как функция от n:

$$U(n) = a \cdot c^n, \tag{2}$$

где a и c — параметры.

В работе [2] выведены следующие формулы для оценивания параметров c и a:

$$c = \exp\left(\sum_i \ln\frac{U_i}{U_{i-1}}(n_i - n_{i-1}) / \sum_i (n_i - n_{i-1})^2\right); \tag{3}$$

$$a = \exp\left(\frac{1}{K}\sum_{i=1}^{K}(\ln U_i - n_i \ln c)\right). \tag{4}$$

Суммирование в этих формулах ведётся по количеству измерений K, помещённых в табл. 1–3. Вычисленные по формулам (3) и (4) оценки параметров a и c для формулы (2) у трёх рассмотренных алгоритмов для трёх видов распределений приведены в табл. 4.

Таблица 4

Оценки параметров a и c в формуле (2) для количества узлов

Вид распределения	Алгоритм 1		Алгоритм 2		Алгоритм 3	
	Параметр a	Параметр c	Параметр a	Параметр c	Параметр a	Параметр c
1	4,9957	1,1898	11,162	1,0926	10,667	1,0922
2	0,11917	1,7854	0,43497	1,7101	0,93746	1,4071
3	1,3519	1,4000	0,20319	1,3642	0,08029	1,3895

Время вычисления (в секундах) как функция от n:

$$U(n) = b \cdot c^n. \tag{5}$$

Формулы для оценивания параметров c и b, вывод которых аналогичен выводу формул (3) и (4), следующие:

$$c = \exp\left(\left(\sum_i \ln\frac{U_i}{U_{i-1}}(n_i - n_{i-1}) - \sum_i \ln\frac{n_i}{n_{i-1}}(n_i - n_{i-1})\right) / \sum_i (n_i - n_{i-1})^2\right); \tag{6}$$

$$b = \exp\left(\frac{1}{K}\sum_{i=1}^{K}(\ln U_i - n_i \ln c - 2\ln n_i)\right). \tag{7}$$

Вычисленные по формулам (6) и (7) оценки параметров b и c для формулы (5) у трёх рассмотренных алгоритмов для трёх видов распределений приведены в табл. 5.

Таблица 5

Оценки параметров b и c в формуле (5) для времени

Вид распределения	Алгоритм 1		Алгоритм 2		Алгоритм 3	
	Параметр b	Параметр c	Параметр b	Параметр c	Параметр b	Параметр c
1	$5{,}1052 \cdot 10^{-8}$	1,1857	$18{,}646 \cdot 10^{-8}$	1,0904	$22{,}076 \cdot 10^{-8}$	1,0885
2	$0{,}38013 \cdot 10^{-8}$	1,6549	$1{,}5407 \cdot 10^{-8}$	1,6255	$3{,}6702 \cdot 10^{-8}$	1,4113
3	$1{,}7476 \cdot 10^{-8}$	1,3478	$0{,}14256 \cdot 10^{-8}$	1,3391	$0{,}14172 \cdot 10^{-8}$	1,3449

Рассмотренные алгоритмы были также протестированы на известной ЗК с 48 городами [5] с симметричной матрицей. Алгоритм 1 получил результат за 1,5 мин, алгоритм 2 — за 45 мин, а алгоритм 3 — за 11 с. Все они вычислили один и тот же оптимальный маршрут длиной 11461, такой же, как в работе [2]: 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.

Программы, реализующие алгоритмы 2 и 3, включая модуль с описанием класса для решения ЗК, написаны на Паскале в системе Delphi. Исходные тексты программ в виде архивов доступны в сети интернет по ссылкам [6, 7]. Кроме того, в архив [7] помещен файл с матрицей расстояний для упомянутой ЗК с 48 городами [5].

Заключение

Как показал вычислительный эксперимент, время выполнения рассмотренных алгоритмов существенно различается для матриц расстояний с распределениями разных видов. Модифицированный алгоритм по быстродействию существенно превосходит алгоритм Литтла на случайных матрицах, однако уступает ему на матрицах евклидовых расстояний. Предложенный способ вычисления более точной нижней границы наилучшего решения ЗК на промежуточных шагах работы алгоритма, когда часть рёбер включена в маршрут, позволил создать новый алгоритм, который существенно превосходит алгоритм Литтла, в том числе на матрицах евклидовых расстояний. Применение нового алгоритма позволяет заметно увеличить размер матриц с различными распределениями, для которых можно за относительно небольшое время получить точное решение ЗК.

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. *Костюк Ю. Л.* Эффективная реализация алгоритма решения задачи коммивояжера методом ветвей и границ // Прикладная дискретная математика. 2013. № 2 (20). С. 78–90.
3. *Данциг Дж.* Линейное программирование, его применения и обобщения: пер. с англ. М.: Прогресс, 1966.
4. *Ахо А., Хопкрофт Дж., Ульман Дж.* Построение и анализ вычислительных алгоритмов: пер. с англ. М.: Мир, 1979. 536 с.
5. *Хелд М., Карп Р. М.* Применения динамического программирования к задачам упорядочивания // Киб. сборник, старая серия. М.: Мир, 1964. Вып. 9. С. 202–218.
6. *Костюк Ю. Л.* Модифицированный алгоритм решения задачи коммивояжера методом ветвей и границ. Версия 2. Программа и модуль с описанием класса: язык Паскаль в системе Delphi. 2013, июль. Электронный ресурс: `www.inf.tsu.ru/Decanat/Staff.nsf/people/KostjukJuL`.
7. *Костюк Ю. Л.* Модифицированный алгоритм решения задачи коммивояжера методом ветвей и границ с улучшенной нижней границей. Программа и модуль с описанием класса: язык Паскаль в системе Delphi. 2013, август. Электронный ресурс: `www.inf.tsu.ru/Decanat/Staff.nsf/people/KostjukJuL`.