УДК 519.682.1

Ю.Л. Костюк, М.С. Пожидаев

СБАЛАНСИРОВАННАЯ ЭВРИСТИКА ДЛЯ РЕШЕНИЯ ЗАДАЧИ МАРШРУТИЗАЦИИ ТРАНСПОРТА С УЧЕТОМ ГРУЗОПОДЪЕМНОСТИ

Предлагается эвристический алгоритм приближённого решения задачи маршрутизации транспорта с учетом грузоподъемности экипажей и потребностями в товаре вершин-клиентов для случая метрических расстояний. Алгоритм состоит из двух фаз: кластеризации вершин на группы и фазы вычисления маршрутов отдельно по группам. Порядок трудоемкости первой фазы около $O(n^2)$, а второй фазы при использовании алгоритма Лина – Кернигана – не выше $O(n^3)$. Проведенный вычислительный эксперимент показал, что предложенный алгоритм почти не уступает алгоритму Османа по качеству получаемого решения при уменьшении времени работы в десятки раз. Еще лучшее по качеству решение получается при комбинированном применении этих двух алгоритмов.

Ключевые слова: *дискретная оптимизация*, *приближённые алгоритмы*, *задача коммивояжера* (*TSP*), *задача маршрутизации транспорта* (*VRP*), *кластеризация*.

Задача маршрутизации транспорта (ЗМТ) (Vehicle Routing Problem, VRP) – одно из обобщений задачи коммивояжёра (ЗК) (Traveling Salesman Problem, TSP). Она предусматривает построение нескольких замкнутых маршрутов, проходящих через общую вершину-депо, при минимизации суммарной стоимости маршрутов. При одинаковой размерности эта задача гораздо сложнее, чем ЗК, которая, в свою очередь, является NP-трудной. Поэтому в практических приложениях, где число вершин достигает 100 и более, возможно применение лишь приближённых алгоритмов.

Известно несколько вариантов ЗМТ, различающихся дополнительными параметрами входных данных и ограничениями на получаемое решение. В самом простом случае кроме матрицы расстояний между всеми парами вершин задается также либо количество маршрутов, либо максимальное количество вершин, которые могут входить в любой из маршрутов. Авторами в работе [1] предложен ряд алгоритмов решения этого простого варианта ЗМТ для случая метрических расстояний. В настоящей статье рассматривается более сложный вариант ЗМТ, когда для каждой из вершин-клиентов задается величина потребности в товаре и задается максимальная грузоподъёмность экипажа – транспортного средства, развозящего товар из вершины-депо по вершинам-клиентам маршрута. Количество маршрутов (задействованных экипажей) определяется в процессе вычислений.

В последние годы при создании алгоритмов решения задач дискретной оптимизации, в том числе и ЗМТ, используются такие метаэвристические стратегии, как детерминированный и моделируемый отжиг, поиск с исключениями, генетические алгоритмы, алгоритмы муравьиных колоний, нейронные сети и др. Полученные с помощью этих стратегий алгоритмы в некоторых случаях дают хорошие результаты, но обычно требуют подбора большого количества управляющих параметров, зависящих как от вида задачи, так и от конкретного набора входных

данных, что существенно усложняет и ограничивает их практическое применение. Один из лучших по качеству алгоритмов решения ЗМТ, основанный на поиске с исключениями, алгоритм Османа [2], также имеет управляющий параметр – длину списка исключений, но диапазон изменений этого параметра и его наиболее вероятная величина вычисляется на основе входных данных, что является существенным преимуществом. Однако трудоемкость алгоритма растет очень быстро при увеличении размерности задачи, на обычном персональном компьютере его можно эффективно применять, если число вершин не превышает 200 – 300.

В статье рассматривается развитие лучшего из алгоритмов, рассмотренных в [1], – сбалансированного последовательного дихотомического разделения вершин на маршруты по критерию разности расстояний – для ЗМТ с учетом грузоподъемности экипажей и потребности в товаре. Сравнение с алгоритмом Османа проводится с помощью вычислительного эксперимента.

1. Сбалансированный алгоритм кластеризации

Предлагаемый алгоритм содержит две фазы: вначале решается задача кластеризации (разделение общего множества вершин на группы для каждого экипажа), затем выполняется построение конечных маршрутов путём решения традиционной ЗК. Ниже приведено подробное описание сбалансированной процедуры кластеризации; последующий способ решения ЗК может быть выбран произвольно среди известных методов. Алгоритм работает без заданного заранее числа экипажей, вычисляя его в ходе работы.

Алгоритм содержит дополнительные условия, определяющие требование сбалансированности. Такой подход позволяет уменьшить объём вычислений, но необходимо решить вопрос о его влиянии на качество получаемых результатов. Основная сложность в реализации алгоритма связана с тем, что в его работе постоянно учитываются две величины: стоимость маршрута и суммарная потребность в товаре вершин-клиентов, включенных в маршрут. Стремление к наилучшему распределению загрузки экипажей может негативно сказываться на стоимости маршрута, и наоборот. Другая сложность заключается в том, что потребность в товаре у клиентов может быть достаточно неравномерной, и существует риск, что добавление вершин на основе оценки стоимости решения в самый последний момент нарушит ограничение грузоподъёмности.

Рассмотрим описание алгоритма. Дано множество из n вершин $V = \{v_1, \ldots, v_n\}$. Для каждой вершины v_i, принадлежащей V, задана величина потребности в товаре $Q(v_i) > 0$. Все экипажи имеют заданную заранее одинаковую грузоподъёмность q. Также задаётся особая вершина – депо v_0, представляющая точку начала и конца всех построенных маршрутов. Для общности полагаем, что $Q(v_0) = 0$. Для любой пары вершин i и j из $\{v_0, v_1, \ldots, v_n\}$ определены расстояния (стоимость переезда) $D(i, j)$ из i в j. Будем считать, что расстояния метрические, т.е. они симметричны и для них выполняется условие треугольника.

Алгоритм строит множество $R = \{r_1, \ldots, r_k\}$ из k замкнутых маршрутов (k не задано), таких, что:

1) вершина v_0 принадлежит каждому из k маршрутов;

2) каждая из вершин v_i, принадлежащая множеству V, принадлежит одному и только одному из k маршрутов;

3) для каждого из k маршрутов r_i, которому принадлежат $m(i)$ вершин $\{v_0, v_{i_1}, v_{i_2}, \ldots, v_{i_{m(i)}}\}$, должно выполняться следующее условие:

$$\sum_{j=1}^{m(i)} Q(v_{i_j}) \le q\,. \qquad (1)$$

Еще раз заметим, что первая фаза алгоритма является всего лишь алгоритмом кластеризации, и для получения окончательных маршрутов необходимо решить ЗК для каждого из k множеств r_i.

В идеале алгоритм должен получать такое множество маршрутов R, чтобы их суммарная длина была минимальной. Обычно такое требование недостижимо, поэтому алгоритм должен хотя бы приближаться к идеалу. Так, чаще всего, если удается разделить все множество вершин на меньшее число маршрутов, то и их суммарная длина будет меньше. Кластеризацию будем производить следующим образом: вначале разделим все вершины множества V на две группы, затем каждую из групп снова на две группы и т.д., до тех пор, пока каждая из групп не будет удовлетворять условию (1).

2. Рекурсивная процедура дихотомического сбалансированного разделения вершин

В ходе вычислений используется рекурсивная процедура *divide* (V',k'), выполняющая разделение вершин. Для соблюдения однозначности параметры процедуры обозначаем буквами с апострофами. Процедура должна выполнить разделение множества вершин V' на k' групп. При использовании процедуры множество V' будет некоторым подмножеством из V. На каждом этапе производится деление только на две группы, а затем выполняется рекурсивный вызов с целью продолжить разделение. Процедуру *divide*(V',k') можно представить в следующем виде.

1. Инициализация некоторых параметров. Переменная q_t должна содержать величину общей потребности в товаре для всех вершин из V'. Переменные k'_1 и k'_2 хранят значения, вершины для какого количества экипажей должны быть отнесены к первой и второй группам. Они вычисляются по формулам:

$$k'_2 = \llcorner k'/2 \lrcorner,\ k'_1 = k' - k'_2.$$

Переменные Δ_1 и δ_1 обозначают верхнюю и нижнюю границы суммы количества товара тех вершин, которые могут быть отнесены к первой группе при разделении, аналогично это справедливо и для Δ_2 и δ_2 для второй группы. Их значения вычисляются следующим образом:

$$\Delta_1 = k'_1 \cdot q,\ \Delta_2 = k'_2 \cdot q,\ \delta_1 = q_t - \Delta_2,\ \delta_2 = q_t - \Delta_1.$$

Обратим внимание, что значения δ_1 или δ_2 могут оказаться отрицательными. В этом случае величина k' уменьшается на 1 и вычисления повторяются.

2. Далее ищутся две вершины s_1 и s_2 с максимальной стоимостью переезда между ними. Вершины s_1 и s_2 являются начальным наполнением будущих групп. Если k'_1 не равно k'_2, то приводимое ниже описание алгоритма необходимо выполнить два раза с обменом значений s_1 и s_2 и выбором наилучшего варианта. Вопрос, как сравнить качество двух вариантов, обсуждается ниже.

3. Сначала рассмотрим разбиение в общем случае, исключая $k' = 2$ и $k' = 3$, разбиение для последних описано отдельно. Итак, вершины s_1 и s_2 являются начальным наполнением будущих групп. Далее на основе различных критериев мы будем относить оставшиеся вершины из V' в первую или вторую группу. Важную роль играет понятие близости вершины к группе. Используется оценка, равная разности расстояний между некоторой вершиной и вершиной, ближайшей из первой группы, и вершиной, ближайшей из второй группы. При практической реали-

зации вычисления такой оценки для каждой вершины, еще не отнесенной к какой-либо группе, должны храниться ссылки на ближайшие вершины из обеих групп, и после перемещении вершины в группу эти ссылки должны обновляться.

4. Если $k'_1 \neq k'_2$, то выполняем добавление к первой группе ближайших к ней вершин до тех пор, пока не будет достигнута величина средней загрузки экипажа, которая вычисляется как q_t / k'.

5. В дальнейших вычислениях потребуются значения желаемого количества товара для каждой группы. Получить эти значения можно путём умножения чисел k'_1 и k'_2 на величину средней загрузки экипажа. До тех пор пока количество товара у вершин, отнесенных к первой группе, меньше желаемого количества для первой группы, и количество товара вершин, отнесенных к второй группе, меньше желаемого уровня для второй группы, производим следующие действия:

(a) если запас для первой группы, равный разнице желаемого уровня её загрузки и суммы текущего наполнения, больше запаса для второй группы, вычисленного аналогичным способом, и добавление к первой группе ближайшей к ней вершины не приведёт к превышению желаемого уровня, то относим к первой группе ближайшую к ней вершину;

(b) если добавление ко второй группе ближайшей к ней вершины не приведёт к превышению желаемого уровня наполнения, то относим ко второй группе ближайшую к ней вершину;

(c) вычислим два значения для каждой группы, равных оставшемуся резерву в каждой из них после добавления к ней текущей ближайшей вершины. Необходимо обратить внимание, что в общем случае каждое из значений может оказаться отрицательным, поэтому обязательно используем их модули. Если полученное значение для первой группы меньше, чем для второй, то относим к первой группе ближайшую к ней вершину, в противном случае — ко второй.

6. Пока остаются нераспределённые вершины, относим их к той группе, где не произошло превышения желаемого уровня загрузки.

7. Полученные уровни загрузки групп должны быть обязательно не меньше чем δ и не больше чем Δ, которые были вычислены выше (Δ_1, δ_1 для первой группы и Δ_2, δ_2 –- для второй). Если какая либо группа нарушает это условие, то необходимо произвести процедуру балансировки:

(a) просматриваем вершины той группы, где произошло превышение Δ. Обратим внимание, что превышение этого значения у обеих групп одновременно невозможно. Порядок просмотра должен быть обратным тому, как происходило приписывание вершин группе;

(b) если перенос некоторой вершины в противоположную группу не приводит к превышению значения Δ в ней, то такой перенос выполняем. Если после переноса нарушение границ было устранено, то операцию балансировки завершаем. В противном случае переходим к следующей вершине;

(c) если после просмотра всех вершин устранить нарушение не удалось, завершаем работу процедуры *divide* (V',k') аварийно.

8. Когда все вершины распределены по группам без нарушения границ Δ и δ, то необходимо последовательно произвести рекурсивные вызовы процедуры для каждой из полученных групп, указывая соответствующие значения V' и k'. Если какой-нибудь из них завершиться аварийно, то также завершаем работу аварийно.

Необходимо описать два частных случая при $k = 2$ и $k' = 3$. Приведём сначала случай для $k' = 2$.

1. Пока есть нераспределённые вершины и уровни загрузки каждой группы не

достигли соответствующего этой группе значения δ, относим ближайшую вершину к той группе, уровень наполнения которой дальше отстоит от значения δ.

2. После выполнения предыдущего шага возможно превышение границ Δ и δ. Если это произошло, то необходимо выполнить операцию балансировки, как она была описана выше. Если она завершилась неуспешно, то завершаем работу процедуры аварийно.

3. Оставшиеся вершины распределяем по принципу близости к некоторой группе в следующем порядке: если ближайшая к первой группе вершина приводит к меньшему росту количества товара, чем аналогичная вершина для второй, то добавляем вершину к первой группе, и наоборот.

Случай для $k' = 3$ может быть представлен следующим образом.

1. Пока есть нераспределённые вершины и уровни загрузки каждой группы не достигли соответствующего этой группе значения δ, относим ближайшую вершину к той группе, уровень которой дальше отстоит от значения δ.

2. После выполнения предыдущего шага возможно превышение границ Δ и δ. Если это произошло, то необходимо выполнить операцию балансировки, как она была описана выше. Если она завершилась неуспешно, то завершаем работу процедуры аварийно.

3. Все оставшиеся вершины относим ко второй группе.

Необходимо решить вопрос, как оценивать качество получаемых групп при выборе наилучшего порядка s_1 и s_2. Существует несколько способов. Можно, например, решать для этого ЗК для всей группы вершин каким-либо алгоритмом, не требующим большого времени для работы. В вычислительном эксперименте использовалась оценка снизу длины возможного пути коммивояжёра, полученная путём суммирования длин рёбер минимального остова, построенного на полном графе вершин группы, с добавлением длины самого длинного ребра.

В целом работу алгоритма можно представить в следующем виде.

1. Вычисляем начальное значение количества экипажей $k = \left\lceil \sum_{i=1}^{n} Q(v_i) \right\rceil / q$.

Если $k = 1$, то задача является ЗК в её обычной постановке и нет необходимости проводить кластеризацию.

2. Произведём вызов процедуры *divide*(V, k), передав ей в качестве параметров все множество вершин V и вычисленное текущее значение $k > 1$. Процедура может возвратить в виде результата множество групп, меньшее, чем k, это допустимое поведение. Если процедура завершила свою работу аварийно, то увеличиваем значение k на единицу и повторяем попытку до тех пор, пока результат не будет получен.

3. Вычисляем совокупность маршрутов, решая ЗК для вершин каждой из полученных групп. В вычислительном эксперименте использован один из лучших приближенных алгоритмов – алгоритм Лина – Кернигана [3].

Оценим трудоемкость алгоритма. При работе процедуры *divide* каждый из шагов 2, 3, …, 7 имеет порядок трудоемкости не выше $O(n^2)$. С учетом рекурсивных вызовов на шаге 8 получаем рекуррентное соотношение для трудоемкости $T(n) = T(n/2) + c \cdot n^2$. Решив его, получаем величину общей трудоемкости $O(n^2)$. При выводе не было учтено, что выполнение процедуры *divide* может закончиться аварийно, и тогда все вычисления повторяются с самого начала. Если количество таких ситуаций не превысит константы, то трудоемкость будет иметь тот же порядок $O(n^2)$. Трудоемкость алгоритма Лина – Кернигана близка к $O(n^3)$, но так как

этот алгоритм применяется по отдельности к каждой из k групп вершин, то общая трудоемкость на этой фазе алгоритма – $O(k \cdot n^3/k^3) = O(n^3/k^2)$. Если с ростом n количество групп k растет пропорционально n, то трудоемкость 2-й фазы будет $O(n)$, а если при росте n количество групп k остается константой, то трудоемкость – $O(n^3)$. Таким образом, порядок общей трудоемкости всего алгоритма будет между $O(n^2)$ и $O(n^3)$.

3. Вычислительный эксперимент

Качество предложенного алгоритма оценивалось путём постановки вычислительного эксперимента. Одни и те же наборы входных данных подавались на вход алгоритма Османа, описанного в работе сбалансированного алгоритма, и комбинированного алгоритма, когда на первом этапе исполняется алгоритм Османа, а на втором – сбалансированный алгоритм. В алгоритме Османа параметру, определяющему количество обмениваемых между маршрутами вершин, задано значение 1. Как отмечено в [3], хотя большее значение этого параметра и позволяет несколько повысить качество решения, однако при этом существенно возрастает время выполнения алгоритма. В алгоритме Османа имеется другой важный целочисленный параметр: длина списка исключений. Его наиболее предпочтительное значение вычисляется по эмпирической формуле [3]

$$t_s = \max\{7; 9{,}6 \cdot \ln(n \cdot k) - 40\}, \qquad (2)$$

где t_s – значение параметра, n – общее количество вершин, k – текущее количество маршрутов.

В первой части вычислительного эксперимента алгоритмы испытывались на задачах (наборах данных), предложенных Кристофидесом, Мингоззи и Тоссом [4], на которых испытывались также многие другие алгоритмы решения ЗМТ. Всех задач 14, из них выбраны те, которые соответствуют постановке ЗМТ с учетом грузоподъемности. Для получения более высокого качества получаемых решений алгоритм Османа выполнялся многократно с изменением параметра длины списка исключений в диапазоне от 7 до $2 \cdot t_s$ и выбирался самый лучший из полученных результатов. Результаты вычислений, приведенные в табл. 1, содержат суммарные длины вычисленных алгоритмами маршрутов, количества получившихся при этом маршрутов (экипажей), а также общее затраченное время.

Таблица 1

Качество и время решения задач Кристофидеса, Мингоззи и Тосса

№ задачи, число вершин	Качество решения, получившееся число экипажей, время работы в секундах		
	Алгоритм Османа	Сбалансированный	Комбинированный
1 (50)	537,6 (5 эк.), 9 с.	559,7 (5 эк.), 0,07 с.	558,2 (5 эк.), 6 с.
2 (75)	885,1 (11 эк.), 35 с.	937,9 (11 эк.), 0,11 с.	881,6 (11 эк.), 26 с.
3 (100)	867,2 (8 эк.), 161 с.	1088,5 (8 эк.), 0,16 с.	863,2 (8 эк.), 201 с.
4 (150)	1122,5 (12 эк.), 993 с.	1171,9 (12 эк.), 0,41 с.	1078,9 (12 эк.), 682 с.
5 (199)	1419,6 (17 эк.), 2749 с.	1462,7 (17 эк.), 1,8 с.	1379,4 (17 эк.), 3715 с
11 (120)	1223,7 (7 эк.), 227 с.	1170,2 (7 эк.), 1,2 с.	1059,2 (7 эк.), 203 с.
12 (100)	905,2 (10 эк.), 220 с.	1066,2 (10 эк.), 0,6 с.	841,4 (10 эк.), 167 с.

Из табл. 1 видно, что сбалансированный алгоритм уступает алгоритму Османа по качеству решения на шести примерах из семи, но затрачивает на вычисления в сотни раз меньше времени. В то же время, если решение, полученное сбалансиро-

ванным алгоритмом, использовать как начальное для алгоритма Османа, то во всех случаях достигается наилучшее качество.

Во второй части вычислительного эксперимента алгоритмы испытывались на случайно сгенерированных наборах данных. Вершины в виде точек на плоскости размещались согласно равномерному распределению внутри единичного квадрата, депо помещалось в центре. Матрица расстояний вычислялась в виде евклидовых расстояний между точками. Веса вершин (потребности в товаре) генерировались согласно дискретному равномерному распределению в диапазоне от 1 до 10. Для наборов данных из 50 вершин алгоритмы запускались по 100 раз, для 100 вершин – 50 раз, для 150 вершин – 25 раз и для 200 вершин – 10 раз. Результаты вычислений, приведенные в табл. 2, содержат отношения суммарных длин полученных маршрутов к их оценке снизу – сумме длин рёбер минимального остова на всех вершинах и длины самого длинного ребра остова. В табл. 2 приведены минимальные, средние и максимальные величины этих отношений по всем выборкам, а также среднее время работы алгоритмов на одном наборе данных. При этом, в отличие от табл. 1, алгоритм Османа выполнялся только при одном значении параметра длины списка исключений, вычисляемом по формуле (2).

Таблица 2

Качество и время решения на случайных выборках

Число вершин, грузоподъемность	Качество алгоритма: минимум, среднее значение, максимум; среднее время работы в секундах		
	Алгоритм Османа	Сбалансированный	Комбинированный
50 (50)	1,60/1,83/2,17; 0,5 с.	1,58/1,84/2,14; 0,04 с.	1,46/1,73/2,10; 0,5 с.
100 (50)	1,97/2,19/2,52; 3,6 с.	1,96/2,23/2,56; 0,2 с.	1,87/2,10/ 2,33; 2,7 с.
150 (50)	2,16/2,43/2,68; 23 с.	2,34/2,52/2,69; 0,5 с.	2,17/2,37/2,53; 17 с.
200 (50)	2,53/2,73/2,86; 87 с.	2,60/2,83/2,99; 0,9 с.	2,48/2,66/2,79; 58 с.
100 (100)	1,53/1,73/1,99; 5,5 с.	1,49/1,60/1,73; 0,3 с.	1,40/1,51/1,67; 4,1 с
150 (150)	1,53/1,66/1,82; 33 с.	1,42/1,52/1,63; 0,8 с.	1,36/1,43/1,50; 19 с.
200 (200)	1,51/1,62/1,79; 144 с.	1,37/1,45/1,52; 2,1 с.	1,33/1,38/1,43; 59 с.

Из табл. 2 видно, что на первых четырех выборках, когда количество вершин в каждом из полученных маршрутов было в среднем менее 10, сбалансированный алгоритм оказался хуже алгоритма Османа по качеству решения в среднем от 0,5 до 3 %, при времени работы в десятки раз меньшем. На последних трех выборках, когда количество вершин в каждом из полученных маршрутов росло с ростом общего числа вершин, картина обратная, при этом преимущество сбалансированного алгоритма достигает более 10 % при $n = 200$. При этом сбалансированный алгоритм затрачивает на вычисления в десятки раз меньше времени. Применение же комбинации этих двух алгоритмов всегда дает выигрыш (в среднем) по сравнению с алгоритмом Османа, который при $n = 200$ и грузоподъемности 200 доходит до 15 % при заметном уменьшении времени вычислений.

Для вынесения решения о том, какой из алгоритмов оказался лучшим на той или иной случайной выборке, в табл. 3 приведены результаты сравнения алгоритмов между собой по качеству получаемых решений в таком виде, чтобы было можно применить знаковый статистический критерий.

Применение критерия знаков к табл. 3 подтверждает преимущество сбалансированного алгоритма над алгоритмом Османа для последних трех выборок и преимущество комбинированного алгоритма над алгоритмом Османа для всех семи

выборок. Согласно статистическим таблицам [5, с. 354], это утверждение имеет вероятность ошибки менее 0,005.

Таблица 3

Сравнение алгоритмов по качеству получаемых решений

Число вершин, грузоподъемность	Количество проигрышей алгоритма Османа из общего количества выборок	
	Проигрыши сбалансированному алгоритму	Проигрыши комбинированному алгоритму
50 (50)	45 из 100	89 из 100
100 (50)	17 из 50	45 из 50
150 (50)	4 из 25	23 из 25
200 (50)	2 из 10	10 из 10
100 (100)	45 из 50	50 из 50
150 (150)	24 из 25	25 из 25
200 (200)	10 из 10	10 из 10

Заключение

Предложенный в статье сбалансированный алгоритм кластеризации путем многократного дихотомического разделения вершин в сочетании с алгоритмом Лина – Кернигана получает приближенное решение задачи маршрутизации транспорта с учетом грузоподъемности экипажей и потребностями в товаре вершин-клиентов. Как показал вычислительный эксперимент, качество его решения в некоторых ситуациях незначительно уступает одному из лучших алгоритмов – алгоритму Османа, однако при увеличении количества вершин до 150 и более при пропорциональном увеличении вершин в отдельных маршрутах предложенный алгоритм начинает превосходить алгоритм Османа при одновременном уменьшении времени вычислений в десятки раз. Благодаря относительно невысокому порядку трудоемкости (менее $O(n^3)$) алгоритм можно применять для задач размерности до 1000 вершин и более, что недоступно алгоритму Османа.

Особенно эффективно применение предложенного алгоритма, дающего начальное приближение, для алгоритма Османа. Интересно, что при этом время работы алгоритма Османа на второй стадии заметно уменьшается. На практике применение комбинированного алгоритма удобно тем, что если время работы на второй стадии оказывается чрезмерным, процесс вычислений можно прервать и использовать полученный к этому моменту вариант решения задачи.

ЛИТЕРАТУРА

1. *Костюк Ю.Л., Пожидаев М.С.* Приближённые алгоритмы решения сбалансированной задачи *k* коммивояжёров // Вестник Томского госуниверситета. Управление, вычислительная техника и информатика. 2008. №1(2). С. 106 – 112.
2. *Osman I.H.* Metastrategy Simulated annealing and tabu search algorithms for the vehicle routing problem // Annals of Operations Research. 1993. V. 41. P. 421 – 451..
3. *Lin S., Kernighan B.* An effective heuristic algorithm for the traveling salesman problem // Operations Research. 1973. V. 21. P. 498 – 516..
4. *Christofides N., Mingozzi A., and Toth P.* The vehicle routing problem // Combinatorial Optimization / ed. N. Christofides, A. Mingozzi, P. Toth and C. Sandi. N.Y.: Wiley, 1979.
5. *Большев Л.Н., Смирнов Н.В.* Таблицы математической статистики. М.: Наука, 1983. 416 с.

Костюк Юрий Леонидович, Пожидаев Михаил Сергеевич
Томский государственный университет,
E-mail: kostyuk_y_l@sibmail.com; msp@altlinux.ru

Поступила в редакцию 15 июня 2010 г.