ЭКОНОМИКА И МАТЕМАТИЧЕСКИЕ МЕТОДЫ, 2014, том 50, № 1, с. 117–126

МЕТОДЫ ОПТИМИЗАЦИИ

ПРИМЕНЕНИЕ МЕТАЭВРИСТИЧЕСКИХ АЛГОРИТМОВ ДЛЯ ЗАДАЧИ МАРШРУТИЗАЦИИ ТРАНСПОРТА

© 2013 г. С.А. Сластников[1]

(Москва)

Статья посвящена исследованию современных метаэвристик для задач маршрутизации транспорта. Приведен краткий обзор основных метаэвристических алгоритмов, подробно описан алгоритм муравьиных колоний. Предлагается модификация алгоритма муравьиных колоний, эффективность которой подтверждена результатами вычислительного эксперимента.

Ключевые слова: задача маршрутизации транспорта, метаэвристические алгоритмы, алгоритм муравьиных колоний.

Классификация JEL: C61, R42.

ВВЕДЕНИЕ

Проблема оптимизации поставок продукции всегда привлекала повышенное внимание экономистов, специалистов по логистике и математиков. Среди экономистов это обусловлено важностью транспортировки товаров и связанных с ней издержек (Бауэрсокс, Клосс, 2008, с. 47). Для математиков этот интерес вызван ее значительной сложностью. В литературе рассматриваемая задача известна как задача маршрутизации транспорта (Vehicle Routing Problem); ее не следует путать с транспортной задачей (иногда называемой задачей Монжа–Канторовича), которой посвящено много книг и работ, в том числе известная монография (Гольштейн, Юдин, 1969).

В разд. 1 дана постановка задачи, описана классификация существующих методов решения. В разд. 2 приведен краткий обзор основных метаэвристических методов: имитации отжига, детерминированного отжига, генетического алгоритма, метода поиска с запретами. В разд. 3 представлен классический алгоритм муравьиных колоний и его модификация. Результаты вычислительного эксперимента с использованием модифицированного алгоритма приведены в разд. 4.

1. ЗАДАЧА МАРШРУТИЗАЦИИ ТРАНСПОРТА

1.1. Постановка задачи. Задача маршрутизации транспорта (ЗМТ) является обобщением известной задачи коммивояжера и, следовательно, принадлежит к классу NP-полных задач. Это означает, что для нее не найден алгоритм решения за полиномиальное время и не доказано, что такого алгоритма не существует.

Опишем постановку задачи маршрутизации транспорта с ограничениями грузоподъемности (capacitated vehicle routing problem, CVRP). Задан граф $G = (V, A, D)$, где $V = \{v_0,\ldots,v_n\}$ – множество вершин (v_0 – склад, остальные вершины – клиенты); A – набор дуг, соединяющих соответствующие вершины графа; $D = \{d_{ij}, i, j = 0, 1,\ldots,n\}$ – множество неотрицательных чисел, которые чаще всего имеют смысл длины пути, времени или стоимости перевозки по дуге между вершинами v_i и v_j (далее для краткости будем называть их просто i и j). Для исследования не

[1] Автор выражает искреннюю благодарность Е.Г. Гольштейну за внимание к данной работе и ценные замечания по ее улучшению.

так важно конкретное содержание величин d_{ij}, поэтому их можно рассматривать как обобщение всех видов затрат на передвижение из i в j. В дальнейшем, если не оговорено противное, под d_{ij} будет пониматься расстояние между вершинами i и j. Для клиента в вершине i (клиента i) задан неотрицательный спрос c_i на некую продукцию или товар. На складе имеется m транспортных средств, грузоподъёмность каждого из которых ограничена числом C_k ($k = 1,\ldots, m$). Также для задачи заданы следующие ограничения:

– каждый клиент должен быть посещён ровно один раз;

– местом начала и окончания всех маршрутов транспортных средств является склад.

Целью задачи является построение маршрутов минимальной суммарной стоимости, удовлетворяющих спрос всех клиентов и не нарушающих ограничений, описанных выше.

Построим математическую модель задачи. Ее можно сформулировать следующим образом:

$$F = \sum_{k=1}^{m}\sum_{i=0}^{n}\sum_{j=0}^{n} d_{ij}X_{ij}^{k} \to \min \tag{1}$$

при ограничениях

$$\sum_{i=1}^{n} c_i \sum_{j=0}^{n} X_{ij}^{k} \le C_k \quad \forall k = 1, \ldots, m, \tag{2}$$

$$\sum_{j=1}^{n} X_{0j}^{k} \le 1 \quad \forall k = 1, \ldots, m, \tag{3}$$

$$\sum_{i=1}^{n} X_{i0}^{k} \le 1 \quad \forall k = 1, \ldots, m, \tag{4}$$

$$\sum_{k=1}^{m}\sum_{i=0}^{n} X_{ij}^{k} = 1 \quad \forall i = 1, \ldots, n, \tag{5}$$

$$\sum_{i=0}^{n} X_{ih}^{k} - \sum_{j=0}^{n} X_{hj}^{k} = 0 \quad \forall h = 1, \ldots, n, \quad \forall k = 1, \ldots, m, \tag{6}$$

$$X_{ij}^{k} \in \{0,1\} \quad \forall i, j = 0, \ldots, n, \quad \forall k = 1, \ldots, m, \tag{7}$$

$$X_{ii}^{k} = 0 \quad \forall i = 0, \ldots, n, \quad \forall k = 1, \ldots, m, \tag{8}$$

если $S_k = \{(i, j)\colon X_{ij}^{k} = 1\}$, то

$$\forall k = 1, \ldots, m \quad \forall (i, j) \in S_k \ne \varnothing\, \exists (j_p, j_{p+1}) \in S_k (p = 0, \ldots, l):$$

$$j_0 = j_{l+1} = 0, \quad (i, j) = (j_r, j_{r+1}), \quad r \le l. \tag{9}$$

Таким образом, X_{ij}^{k} принимает значение 1, если транспортное средство k следует от клиента i к клиенту j, и 0 в противном случае. S_k – множество пар вершин, определяющих маршрут машины k. Целевая функция (1) минимизирует стоимость всех маршрутов всех транспортных средств. Неравенство (2) гарантирует, что выполняются ограничения грузоподъемности каждого транспортного средства. Ограничения (3) и (4) определяют, что каждое транспортное средство не может покинуть склад и вернуться на склад более 1 раза. Равенство (5) показывает, что каждый клиент обслуживается только одним транспортным средством и только один раз. Условие (6) означает, что если транспортное средство прибывает в вершину, то оно так же и покидает данную вершину. Условие (9) исключает возможность расщепления маршрута транспортного средства на несвязные циклы.

1.2. Классификация методов решения. Существующие на сегодняшний день методы решения задач маршрутизации транспорта можно в самом широком смысле классифицировать следующим образом:

– точные методы;

– классические эвристические методы;

– метаэвристические методы.

Поскольку ЗМТ является обобщением задачи коммивояжера, идеи многих *точных алгоритмов* решения ЗМТ были заимствованы из методов решения этой задачи. До конца 1980-х годов наиболее эффективными точными методами решения ЗМТ являлись методы ветвей и границ, основанные на базовых комбинаторных упрощениях для задачи о назначениях, задачи минимального обхода графа, а также на методе сокращения пространства состояний. К настоящему времени предложены более сложные границы, основанные на лагранжевых релаксациях и аддитивном подходе, что позволило решать задачи существенно большей размерности (Toth, Vigo, 2002, p. 29). Однако для решения практических задач точные методы используются крайне редко, так как они применимы при некоторых дополнительных предположениях, чувствительны к любым дополнительным ограничениям, а время вычислений (в силу NP-полноты задачи) растет слишком быстро при увеличении размерности задачи.

Классические эвристики (разработаны в 1960–1990-х годах) можно разбить на следующие группы: конструктивные алгоритмы, двухфазные алгоритмы и улучшающие алгоритмы (Laporte, Semet, 1999).

Эвристические методы проводят поиск в относительно ограниченном пространстве и в целом дают неплохое качество решений за небольшое количество вычислений. Кроме того, большинство из них может быть легко приспособлено для учета разнообразных ограничений, возникающих в реальных условиях. Поэтому эвристики по-прежнему широко используются, особенно в коммерческих программных пакетах. Подробный обзор эвристик можно найти в работе (Сластников, 2012).

Метаэвристические алгоритмы впервые начали разрабатываться после 1990-х годов. Большинство этих алгоритмов были предложены на основе идей, возникших при наблюдении процессов живой и неживой природы. В метаэвристиках акцент делается на проведение глубоких исследований в наиболее перспективных областях пространства решений. Эти методы обычно сочетают сложные правила поиска, структуры данных и рекомбинации решений. Качество получаемых с помощью этих методов решений, как правило, намного выше, чем у классических эвристик, но и время вычислений при этом возрастает. Более того, некоторые метаэвристики зависят от контекста конкретной задачи и требуют тонко настроенные управляющие параметры, что может сделать трудным их распространение на другие задачи. В некотором смысле, метаэвристики это не более чем усложненные улучшающие алгоритмы, и они могут рассматриваться как естественные усовершенствования классических эвристик (Laporte, Gendreau et al., 2000).

2. МЕТАЭВРИСТИЧЕСКИЕ АЛГОРИТМЫ

Метаэвристические алгоритмы составляют основу современных исследований в области приближенных методов решения ЗМТ.

2.1. Имитация отжига. Идея метода имитации отжига (simulated annealing) появилась при наблюдении процессов, происходящих в металле в ходе охлаждения после достижения им температуры плавления (Kirkpatrick, Gelatt, Vecchi, 1983). Этот метод является модификацией вероятностного метода градиентного спуска. Известно множество его вариаций для решения различных задач оптимизации. Опишем общую идею метода. В первую очередь необходимо определить понятие окрестности решения $O(x)$. Это те решения, которые могут быть получены из решения x путём выполнения фиксированного набора преобразований. Преобразования могут различаться для разных вариантов алгоритма. Алгоритм является итеративным и начинается с выбора начального решения задачи, которое на первом шаге будет являться текущим. Имея текущее решение задачи x_i^* на шаге i, произвольно выбирается новое решение $x_{i+1} \in O(x_i^*)$. Выбранное решение становится текущим со следующей вероятностью:

$$P = \begin{cases} 1, \text{ если } F(x_{i+1}) < F(x_i^*): \\ \exp\left(-\left(F(x_{i+1}) - F(x_i^*)\right)/\theta_i\right), \text{ если } F(x_{i+1}) \geq F(x_i^*), \end{cases}$$

где $F(x_i)$ – значение целевой функции на решении x_i, θ_i – элементы произвольной убывающей, сходящейся к нулю положительной последовательности (которая задаёт аналог падающей температуры). Алгоритм останавливается после заранее предопределенного числа итераций.

Алгоритм имитации отжига похож на градиентный спуск, но за счёт случайного выбора промежуточной точки решения будут попадать в локальные экстремумы реже, чем при простом градиентном спуске.

2.2. Детерминированный отжиг. Несмотря на успешное применение в различных комбинаторных задачах методы имитации отжига часто работают медленно в силу стохастического механизма выбора текущего решения. Алгоритмы детерминированного отжига (deterministic annealing) пытаются преодолеть этот недостаток, используя детерминированные критерии выбора текущего решения. Наиболее известными из них являются алгоритм порогового принятия (Dueck, Scheurer, 1990) и алгоритм движения от рекорда к рекорду (Dueck, 1993).

В этих алгоритмах новое решение $x_{i+1} \in O(x_i^*)$ становится текущим, если $F(x_{i+1}) - F(x_i^*) < \theta$, где $\theta > 0$ – величина порога (для алгоритма порогового принятия), или $F(x_{i+1}) < \theta F(x_i^*)$, где значение параметра θ обычно чуть больше единицы (для алгоритма движения от рекорда к рекорду).

2.3. Генетические алгоритмы. Генетические алгоритмы – это алгоритмы, которые позволяют найти решение для аналитически трудно решаемых задач путем последовательного подбора и комбинирования искомых параметров, используя механизмы, напоминающие биологическую эволюцию. Впервые такой подход был предложен Джоном Холландом (Holland, 1975).

Опишем общую схему работы алгоритма. Сначала произвольно генерируется начальная популяция допустимых решений задачи $X^0 = \{x_1^0, \ldots, x_N^0\}$. Далее на каждой итерации (шаге эволюции) выполняются следующие действия.

1. С помощью оператора селекции выбираются два родителя x_i и x_j.

2. Оператором скрещивания (в биологии его принято называть кроссинговером) строится новое решение-потомок x' на основе решений-родителей.

3. Полученный потомок подвергается “мутациям”, которые представляют собой небольшие стохастические модификации данного решения.

4. После мутации решение добавляется в популяцию, а решение с наихудшим показателем приспособленности (значением целевой функции) из популяции удаляется.

5. Процесс повторяется до тех пор, пока не выполнен критерий остановки.

Если в качестве начального множества решений взять решения, полученные с помощью одной из классических эвристик, то это может значительно сократить время исполнения алгоритма. Подробнее описание генетических алгоритмов можно найти, например, в работе (Батищев, 1995).

2.4. Методы поиска с запретами. Впервые метод поиска с запретами (tabu search) был предложен автором термина “метаэвристика” Ф. Гловером (Glover, 1986) и полностью описан им же в 1989 г. (Glover, 1989a, 1989b). С тех пор было создано множество различных вариантов данного алгоритма, наиболее эффективные из которых представлены в работах (Gendreau, Hertz, Laporte, 1991; Rego, Roucairol, 1996; Toth, Vigo, 1998; Xu, Kelly, 1996; Rochat, Taillard, 1995).

Общая схема метода поиска с запретами состоит из следующих шагов.

1. Формируется множество *Tabu* (список запретов), изначально оно пусто.

2. Методом локального поиска строится начальное решение задачи X_0.

3. В список запретов *Tabu* добавляется полученное решение.

4. Формируется некая окрестность полученного решения $O(X_0)$.

5. Проводится локальный поиск в множестве $O(X_0)\backslash Tabu$ и получается новое решение X_1.

6. Процесс повторяется до тех пор, пока не выполнен критерий остановки.

Таким образом, основная идея метода поиска с запретами состоит в том, чтобы не останавливать поиск, попадая в локальный экстремум, а, двигаясь от одного локального экстремума к другому, попытаться достичь глобального минимума. Список запретов при этом позволяет избежать попадания в уже пройденные локальные минимумы.

3. МУРАВЬИНЫЕ АЛГОРИТМЫ

3.1. Классический алгоритм муравьиных колоний. Идея алгоритма оптимизации, подражающего муравьиной колонии (Ant Colony Optimization), была впервые высказана в 1991 г. (Colorni, Dorigo, Maniezzo, 1991) и подробно описана в (Dorigo, 1992). Суть подхода заключается в использовании модели поиска пищи в колониях муравьёв, которые помечают пройденный путь, выбрасывая специальные ароматические эссенции, называемые феромонами. Оставленные следы привлекают запахом других муравьев, которые, проходя по помеченным путям, в свою очередь, усиливают запах феромона. Таким образом, муравьи всё чаще проходят пути, ведущие к источнику пищи.

При применении алгоритма муравьиных колоний к решению ЗМТ каждый "муравей" рассматривается как модель транспортного средства. Изначально, каждый "муравей" k начинает свой маршрут со склада, причем множество M_k (клиенты, включенные в его маршрут) пусто. Далее, по вероятностному критерию (Dorigo, Gambardella, 1997) "муравей" выбирает следующего клиента j, который будет посещен:

$$j = \begin{cases} \arg\max\limits_{u \notin M_k} [\tau_{iu}(\eta_{iu})^{\beta}] \text{ с вероятностью } q_0; \\ S \text{ с вероятностью } 1 - q_0, \end{cases} \tag{10}$$

где i – текущий клиент; τ_{iu} – количество феромона на пути между клиентами i и u; η_{iu} – некоторая "эвристическая функция", убывающая с ростом расстояния между клиентами (ее иногда называют "видимостью" между клиентами i и u, для простоты можно взять $\eta_{iu} = d_{iu}^{-1}$); β – параметр, характеризующий относительную "важность" расстояния по сравнению с количеством феромона (при $\beta = 0$ "муравей" ориентируется только на количество феромона); q_0 – вероятность использования детерминированного принципа при выборе следующего клиента; S – случайная величина, подчиняющаяся следующему закону распределения вероятностей:

$$P\{S = s\} = p_k(i, s) = \begin{cases} \tau_{is}(\eta_{is})^{\beta} / \sum\limits_{v \notin M_k} \tau_{iv}(\eta_{iv})^{\beta}, \text{ если } s \notin M_k; \\ 0, \text{ если } s \in M_k, \end{cases} \tag{11}$$

т.е. $p_k(i, s)$ – вероятность, с которой "муравей" k выбирает передвижение от клиента i к клиенту s.

"Муравей" возвращается в депо, когда исчерпана его "грузоподъемность" либо все клиенты посещены. Алгоритм строит полный маршрут для первого "муравья" и лишь затем начинает строить для второго. Это происходит до тех пор, пока для каждого из заранее заданного числа "муравьев" m не построен выполнимый маршрут.

Для улучшения последующих решений необходимо обновлять следы феромона в зависимости от качества получаемых решений. Локальное обновление феромона моделирует его естественное испарение и гарантирует, что никакой маршрут не станет слишком превалирующим. Это обновление происходит после построения полного маршрута каждым "муравьем" и выражается формулой:

$$\tau_{ij}^{new} = (1 - \alpha)\tau_{ij}^{old} + \alpha\tau_0, \tag{12}$$

где α – параметр, характеризующий скорость испарения феромона, τ_0 – начальное значение феромона.

После того как все m "муравьев" проложили допустимые маршруты, происходит глобальное обновление феромона, заключающееся в добавлении феромона ко всем дугам лучшего из решений, найденного одним "муравьем". След феромона на этих ребрах обновляется следующим образом:

$$\tau_{ij}^{new} = (1-\alpha)\tau_{ij}^{old} + \frac{\alpha}{L}, \tag{13}$$

где L – суммарные затраты лучшего маршрута. Такое обновление поощряет использование более "дешевых" маршрутов, увеличивая вероятность того, что будущие маршруты будут задействовать дуги, содержавшиеся в лучших решениях. Этот процесс повторяется предопределенное число раз, и лучшее из всех решений дает хорошее приближение оптимального решения задачи.

3.2. Модифицированный алгоритм муравьиных колоний. Для применения алгоритма муравьиных колоний, описанного выше, к решению задачи маршрутизации транспорта необходимо определить параметры α, β, q_0, τ_0 для каждой конкретной задачи или же сформулировать правила их вычисления. Например, в работе (Bell, McMullen, 2004) предлагается в качестве начального значения феромона τ_0 брать величину, обратную к затратам на лучшем из известных решений для данной задачи. Однако такой подход приемлем лишь для модельных задач, где известны оптимальные или субоптимальные решения.

Предлагаемая в настоящей работе модификация определяет начальное значение феромона по формуле:

$$\tau_0 = \left((n+1)\min_{i\neq j} d_{ij}\right)^{-1}, \tag{14}$$

где n – число клиентов. Кроме того, распределение вероятностей (11), определяющее выбор следующего клиента, изменяется следующим образом:

– после построения полного маршрута каждым "муравьем" предлагается запоминать лучший и худший маршруты одного муравья (среди всех муравьев), L и R соответственно затраты на этих маршрутах;

– в случае, если путь от клиента i к клиенту j входит в худший маршрут, то вероятность выбора этого передвижения уменьшается пропорционально отношению затрат лучшего маршрута к худшему, т.е.

$$\tilde{p}_k(i,j) = \frac{L}{R}\tau_{ij}(\eta_{ij})^{\beta} \Big/ \left(\sum_{v\neq j,\, v\notin M_k} \tau_{iv}(\eta_{iv})^{\beta} + L\tau_{ij}(\eta_{ij})^{\beta}/R\right); \tag{15}$$

– в остальных случаях ($s \neq j$) распределение (11) заменяется на

$$\tilde{p}_k(i,s) = \begin{cases} \tau_{is}(\eta_{is})^{\beta} \Big/ \left(\sum\limits_{v\neq j,\, v\notin M_k} \tau_{iv}(\eta_{iv})^{\beta} + L\tau_{ij}(\eta_{ij})^{\beta}/R\right), & \text{если } s \notin M_k; \\ 0, & \text{если } s \in M_k. \end{cases} \tag{16}$$

Стоит заметить, что для задачи с одним транспортным средством (т.е., по-существу, для задачи коммивояжера) модифицированный алгоритм совпадет с классическим.

Приведем упрощенное описание модифицированного алгоритма.

1. Задаем значения параметров α, β, q_0.

2. Вычисляем начальное значение феромона τ_0 по формуле (14).

3. Цикл по всем итерациям (пока не выполнено правило остановки).

3.1. Цикл по всем "муравьям" ($k = 1,\ldots,m$).

3.1.1. Множество M_k клиентов, включенных в маршрут муравья k, полагаем пустым.

3.1.2. Цикл по одному муравью (пока не исчерпана грузоподъемность “муравья” k или не осталось необслуженных клиентов).

Пополняем множество M_k (следующих клиентов для посещения) по формулам (10), (15), (16).

3.1.3. Конец цикла по одному муравью.

3.1.4. Обновляем феромон локально по формуле (12).

3.2. Конец цикла по всем “муравьям”.

3.3. Находим лучший и худший (среди всех “муравьев”) маршруты.

3.3. Обновляем феромон глобально по формуле (13).

4. Конец цикла по итерациям.

5. Определяем лучший результат из всех итераций.

4. ВЫЧИСЛИТЕЛЬНЫЙ ЭКСПЕРИМЕНТ

Для оценки качества алгоритмов решения ЗМТ с ограничениями грузоподъемности использовались задачи из набора специально разработанных тестовых примеров[2]. Для этих задач известно оптимальное значение целевой функции и минимальное число используемых транспортных средств (ТС), грузоподъемность которых одинакова в рамках каждой задачи.

Эксперимент проводился на задачах *P-n*16-*k*8 (число клиентов – 15), *A-n*32-*k*5 (31 клиент), *P-n*60-*k*10 (59 клиентов) и *P-n*101-*k*4 (100 клиентов). Созданная для генерации решений программа на языке *C++* исполнялась на персональном компьютере с процессором Dual-Core AMD Opteron 2.8 GHz. При написании программы использовалась библиотека точных вычислений *GMP* (свободно распространяемая по лицензии *GNU LGPL*). Это позволило свести вычислительные потери к минимуму без увеличения времени вычислений.

В табл. 1 представлены результаты решения указанных задач классическим алгоритмом муравьиных колоний (АМК) и его модификацией (МАМК), описанной выше. Для классического

Таблица 1. Результаты эксперимента

Задача / Алгоритм	*P-n*16-*k*8		*A-n*32-*k*5		*P-n*60-*k*10		*P-n*101-*k*4	
	АМК	МАМК	АМК	МАМК	АМК	МАМК	АМК	МАМК
Полученное значение целевой функции	451,3	451,3	867,7	856,2	876,6	877,6	900,2	870,4
Полученное число ТС	8	8	5	5	10	10	4	4
α	0,8	0,8	0,8	0,2	0,4	0,4	1	0,6
β	0,7	0,7	0,8	0,7	1	0,9	0,1	0,6
q_0	0,6	0,6	0,9	0,7	0,9	0,9	0,9	0,9
Время выполнения (секунды)	5	5	15	15	60	61	194	197
Число итераций	10000		10000		10000		10000	
Оптимальное значение целевой функции	450		784		744		681	
Оптимальное число ТС	8		5		10		4	

[2] См. раздел *Augerat et al.* на сайте http://neo.lcc.uma.es/vrp/vrp-instances/capacitated-vrp-instances/.

Таблица 2. Относительное отклонение полученных решений от оптимальных, в %

Задача	*P-n*16-*k*8	*A-n*32-*k*5	*P-n*60-*k*10	*P-n*101-*k*4
Классический алгоритм	0,3	10,68	17,83	32,19
Модифицированный алгоритм	0,3	9,21	17,96	27,81

алгоритма в качестве начального значения феромона τ_0 выбиралось значение, предложенное в (Bell, McMullen, 2004), а для модифицированного – в соответствии с формулой (14). Остальные параметры алгоритмов – α, β, q_0 варьировались в пределах от 0 до 1 с шагом 0,1. Число доступных транспортных средств принималось равным 20. Для каждой задачи и каждого алгоритма в таблице представлены значения параметров, при которых получены наилучшие значения целевой функции. В последних двух строках таблицы представлены известные для рассматриваемых задач оптимальные значения целевой функции и числа транспортных средств.

Сравнительным критерием для обработки результатов эксперимента и анализа качества решений служит показатель отклонения полученного значения целевой функции (F) относительно оптимального значения (F_{opt}), рассчитанный по формуле:

$$k = \frac{F - F_{opt}}{F_{opt}} 100\%.$$

Значения этого показателя для классического и модифицированного алгоритма муравьиных колоний приведены в табл. 2.

Из представленных результатов видно, что при сравнительно небольшом числе клиентов оба алгоритма дают по существу одинаковые результаты, практически не отличающиеся от оптимального, а с ростом числа клиентов модифицированный алгоритм становится все более эффективным по сравнению с классическим.

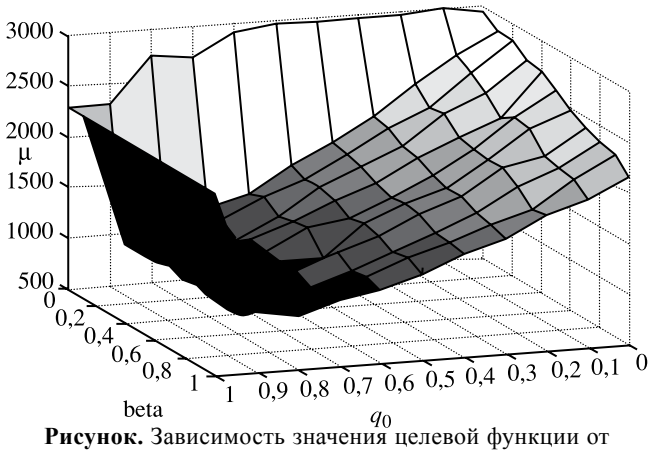

Рисунок. Зависимость значения целевой функции от параметров для задачи *P-n*101-*k*4

На рисунке приведен трехмерный график зависимости значения целевой функции от параметров β и q_0 при фиксированном значении $\alpha = 0{,}5$, построенный по решениям задачи *P-n*101-*k*4 с помощью модифицированного алгоритма. Из графика видна тенденция уменьшения значения целевой функции при увеличении значений параметров β и q_0 (за исключением граничного значения $q_0 = 1$). Это же подтверждается результатами эксперимента для всех четырех задач, где оптимальные пары параметров (β, q_0) получились равными (0,7; 0,6), (0,7; 0,7), (0,9; 0,9) и (0,6; 0,9) соответственно (см. табл. 1). На основании этого можно сделать общий вывод об областях изменения параметров алгоритма, где наиболее вероятно получить хорошее приближение оптимального решения задачи: оптимальные значения параметров β и q_0 с большой вероятностью лежат на отрезке [0,6; 0,9]. Результаты экспериментов показали, что при $q_0 < 0{,}4$ и $q_0 = 1$ полученные значения целевой функции (независимо от значений остальных параметров) очень далеки от оптимального. Кроме того, оптимальные значения параметра q_0 свидетельствуют о том, что при не слишком большом числе клиентов роль стохастической компоненты в алгоритме достаточно велика, а также подтверждают целесообразность правила выбора клиента для посещения с помощью формул (15)–(16). С увеличением числа клиентов вероятность детерминированного выбора очередного клиента будет расти, однако чисто детерминированный алгоритм (соответствующий случаю $q_0 = 1$) будет давать на порядок худший результат, чем стохастический. Относительно параметра α можно сказать лишь

о том, что по результатам проведенных экспериментов не удалось однозначно выявить его оптимальную область: как видно из табл. 1, оптимальные значения α распределились достаточно равномерно на всем отрезке [0; 1]. По мнению автора, оптимальная область для α должна в большой степени зависеть от геометрического расположения множества обслуживаемых клиентов относительно склада (равномерное, кластерное, случайное), что, впрочем, нуждается в дополнительном исследовании.

СПИСОК ЛИТЕРАТУРЫ

Батищев Д.И. (1995). Генетические алгоритмы решения экстремальных задач. Воронеж: ВГТУ.

Бауэрсокс Д., Клосс Д. (2008). Логистика: интегрированная цепь поставок. М.: ЗАО "Олимп-бизнес".

Гольштейн Е.Г., Юдин Д.Б. (1969). Задачи линейного программирования транспортного типа. М.: Наука.

Сластников С.А. (2012). Анализ эвристических и метаэвристических методов для решения задачи распределения автомобильного топлива // *Качество. Инновации. Образование.* № 11.

Bell J., McMullen P. (2004). Ant Colony Optimization Techniques for the Vehicle Routing Problem // *Advanced Engineering Informatics.* No. 18.

Colorni A., Dorigo M., Maniezzo V. (1991). Distributed Optimization by Ant Colonies. In: "*Actes de la Première Conférence Européenne sur la Vie Artificielle*". Paris: Elsevier Publishing.

Dorigo M. (1992). Optimization, Learning and Natural Algorithms. PhD thesis, Politecnico di Milano, Italie.

Dorigo M., Gambardella L.M. (1997). Ant Colonies for the Traveling Salesman Problem // *BioSystems*. No. 43.

Dueck G. (1993). New Optimization Heuristics: The Great Deluge Algorithm and the Record-to-Record Travel // *J. of Computational Physics*. No. 104.

Dueck G., Scheurer T. (1990). Threshold Accepting: A General Purpose Optimization Algorithm // *J. of Computational Physics*. No. 90.

Gendreau M., Hertz A., Laporte G. (1991). A Tabu Search Heuristic for the Vehicle Routing Problem // *Management Science*. Vol. 40. No. 10.

Glover F. (1986). Future Paths for Integer Programming and Links to Artificial Intelligence // *Computer and Operations Research*. Vol. 13. No. 5.

Glover F. (1989). Tabu Search // *INFORMS J. on Computing.* Part 1: Vol. 1. № 3; Part 2: Vol. 2. No. 1.

Holland J. (1975). Adaptation in Natural and Artificial Systems. Ann Arbor: The University of Michigan Press.

Kirkpatrick S., Gelatt C., Vecchi M. (1983). Optimization by Simulated Annealing // *Science*, *New Series*. Vol. 220. No. 4598.

Laporte G., Gendreau M., Potvin J., Semet F. (2000). Classical and Modern Heuristics for the Vehicle Routing Problem // *International Transactions in Operation Research.* No. 7.

Laporte G., Semet F. (1999). Classical Heuristics for the Vehicle Routing Problem // *Les Cahiers du Gerad.* G-98-54.

Rego C., Roucairol C. (1996). A Parallel Tabu Search Algorithm Using Ejection Chains for the Vehicle Routing Problem. In: "Meta-Heuristics: Theory and Applications" Osman I.H., Kelly J.P. (eds.). Boston: Kluwer Academic Publishers.

Rochat Y., Taillard E. (1995). Probabilistic Diversification and Intensification in Local Search for Vehicle Routing // *J. of Heuristics.* No. 1.

Toth P., Vigo D. (1998). The Granular Tabu Search (and its Application to the Vehicle Routing Problem). Technical Report OR/98/9, Dipartimento di Elettronica, Informatica e Sistemistica, Universita di Bologna.

Toth P., Vigo D. (2002). The vehicle routing problem. Philadelphia: SIAM Monographs on Discrete Mathematics and Applications.

Xu J., Kelly J. (1996). A Network Flow-Based Tabu Search Heuristic for the Vehicle Routing Problem // *Transportation Science.* Vol. 30. No. 4.

Поступила в редакцию
29.04.2013 г.

Application of Metaheuristic Algorithms for the Vehicle Routing Problem

S. Slastnikov

The paper observes modern metaheuristics for the vehicle routing problem. Given a brief survey of the principle metaheuristic algorithms, as well as a detailed description of the ant colony optimization algorithm. The modification of ant colony optimization algorithm proposed, the effectiveness of which is confirmed by the computational results.

Keywords: vehicle routing problem, metaheuristic algorithms, ant colony optimization.

JEL Classification: C61, R42.