Муравьиный алгоритм с ослаблением ограничений по временны́м окнам в решении задачи маршрутизации транспорта

О. Э. Долгова, В. В. Пересветов*
Вычислительный центр ДВО РАН, Хабаровск, Россия
*Контактный e-mail: peresvv@mail.ru

Рассмотрена задача маршрутизации транспорта с ограничениями по временны́м окнам. Требовалось составить план доставки товара клиентам, построив маршруты движения идентичных транспортных средств так, чтобы общая длина пройденного пути была минимальной. Для решения задачи разработан гибридный алгоритм. Он состоит из методов построения исходных решений, муравьиного алгоритма и локального поиска. В муравьином алгоритме в процессе формирования маршрутов разрешается нарушение временны́х ограничений при условии добавления штрафа в целевую функцию. Предложенный метод показал высокую эффективность при решении задач кластерного типа и задач с долгосрочным горизонтом планирования.

Ключевые слова: маршрутизация транспорта, временны́е окна, общая длина пройденного пути, гибридный алгоритм, муравьиный алгоритм, локальный поиск.

Библиографическая ссылка: Долгова О.Э., Пересветов В.В. Муравьиный алгоритм с ослаблением ограничений по временны́м окнам в решении задачи маршрутизации транспорта // Вычислительные технологии. 2018. Т. 23, № 5. С. 49–62. DOI: 10.25743/ICT.2018.23.5.005.

Введение

Целью решения задач маршрутизации транспорта (Vehicle Routing Problem — VRP) является составление маршрутов движения транспортных средств (ТС) с минимальными ценовыми затратами. Классическую VRP — задачу маршрутизации с ограничением на грузоподъемность — для расширения области ее практического использования в ряде случаев усложняют путем добавления дополнительных ограничений.

В настоящей работе рассматривается задача маршрутизации транспорта с ограничениями по временны́м окнам (VRP with Time Windows — VRPTW). Эта задача оптимизации принадлежит к классу сложности NP [1]. VRPTW обеспечивает математическую основу для многих возникающих на практике задач о наилучшем распределении некоторого числа работ или ресурсов. В связи с этим предложен широкий спектр различных вариаций VRPTW с разнообразными точными и приближенными алгоритмами их решения. Обзор литературных источников, опубликованных до 2000 г., представлен в [2],

а после 2000 г. — в [1]. В [3] показано, что для ряда методов (поиска с запретами, генетических алгоритмов, алгоритмов имитации отжига, эволюционных алгоритмов и др.) удается повысить эффективность решения задач VRPTW, если ослабить ограничения по временны́м окнам. Ослабление временны́х ограничений используется в методе поиска по окрестности, который предложен в [4] для решения задачи оптимизации автопарка и маршрутов ТС размерности 1000.

Временно́е окно представляет собой отрезок времени, в котором может начаться разгрузка товара у клиента. Если ТС прибывает к клиенту слишком рано, то появляется время ожидания, которое не добавляется к целевой функции (ЦФ). Возвращение ТС на склад должно быть в пределах заданного временно́го срока. Требуется составить маршруты движения ТС и доставить товар всем клиентам так, чтобы общая длина пройденного пути была минимальной. Число доступных ТС не ограничено. Такая постановка задачи, как правило, встречается в литературе по точным алгоритмам решения VRPTW.

Используя современные точные методы [5–7], можно находить решения всех описываемых далее тестовых задач размерности до 100. Для большей размерности (до 1000) также получены некоторые результаты, однако это касается в основном задач с определенной выраженной структурой и/или узкими временны́ми окнами. Тем не менее точные методы редко используются для решения практических задач [8], так как они чувствительны к любым дополнительным ограничениям, а время решения задач большой размерности оказывается неприемлемым. Подробным исследованиям точных методов посвящены работы [9, 10] и др.

Приближенные муравьиные алгоритмы (Ant Colony Optimization — ACO) [11] успешно применяются в составе гибридных методов для решения задач маршрутизации транспорта [8, 12–15], включая задачи с ограничениями по временны́м окнам [14, 15]. В [13] нами предложена гибридная схема, включающая лучевой поиск с муравьиным алгоритмом и метод локальных улучшений, для решения классической задачи маршрутизации. Этот метод оказался эффективным в решении задач кластерного типа и задач небольшой размерности случайного типа. В [14] для решения VRPTW с минимизацией ЦФ затрат на доставку товара представлен минимаксный алгоритм муравьиной колонии [11] с комбинированной схемой локального поиска. В [15] предложены муравьиный алгоритм и метод локальных улучшений, для кластерных задач исследуется связь между качеством исходных решений и качеством получаемых локальных оптимумов.

В настоящей работе разработан гибридный алгоритм, включающий методы построения исходных решений, муравьиный алгоритм и локальный поиск. Если в [14, 15] и других работах ACO служит для построения только решений, не нарушающих ограничений задач, то здесь предложено построение решений с ослаблением временны́х ограничений по схеме *returns in time* [16]. На первом этапе работы алгоритма находятся несколько пробных решений, а в дальнейшем поиск решения продолжается с лучшим из них. Представлены результаты вычислительных экспериментов с часто используемыми тестовыми задачами размерностей 25, 50, 100 [17].

1. Формулировка задачи

Задача VRPTW определена на графе с множеством вершин $\{0, 1, \ldots, N\}$. Вершина 0 представляет на графе склад, где находятся транспортные средства и товар. Возвращение ТС на склад должно быть в пределах временно́го отрезка $[e(0), l(0)]$. Вершинам

$i = 1, \ldots, N$, представляющим на графе клиентов, соответствуют спрос товара $q(i)$, продолжительность его разгрузки $s(i) \geq 0$ и временны́е окна $[e(i), l(i)]$, в течение которых разрешается начинать разгрузку.

Имеется парк идентичных ТС с грузоподъемностью $Q \geq \max\{q(i) : i = 1, \ldots, N\}$. Транспортные средства начинают и заканчивают свои маршруты на складе. Все клиенты посещаются однократно в одном из $K \geq 1$ маршрутов. Пусть в некотором маршруте V^k посещаются n_k клиентов:

$$V^k = (v_0^k, v_1^k, \cdots, v_{n_k}^k, v_{n_k+1}^k), \tag{1}$$

$v_0^k = v_{n_k+1}^k = 0$ (первый и последний пункты посещения соответствуют складу). Длина этого маршрута находится как сумма известных расстояний между назначенными на него пунктами: $\sum_{j=0}^{n_k} d(v_j^k, v_{j+1}^k)$. В маршруте (1) не нарушается ограничение на грузоподъемность, если $\sum_{j=1}^{n_k} q(v_j^k) \leq Q$.

Если все маршруты удовлетворяют ограничениям задачи, то их совокупность есть допустимое решение, которое обозначим $\zeta = \{V^k : k = 1, \ldots, K\}$. В общем случае имеется множество допустимых решений Z, $\zeta \in Z$. Общая длина $F(\zeta)$ всех маршрутов решения ζ является целевой функцией:

$$F(\zeta) = \sum_{k=1}^{K} \sum_{j=0}^{n_k} d(v_j^k, v_{j+1}^k).$$

Требуется найти $\zeta^* = \arg\min_{\zeta \in Z} F(\zeta)$.

2. Алгоритм решения

Для решения рассматриваемой задачи предложен гибридный алгоритм (далее HYB), который включает методы построения исходных решений, муравьиный алгоритм и локальный поиск. В алгоритме 1 дана его общая схема.

Алгоритм 1. HYB

1: инициализация N_{tr} и других данных;
2: **for** $i = 1 \to N_{tr}$ **do**
3: построить исходное решение ζ_i^{tr};
4: LocalSearch(ζ_i^{tr}) — локальный поиск — см. алгоритм 2;
5: ACO(ζ_i^{tr}) — муравьиный алгоритм — см. алгоритм 3;
6: **end for**
7: найти $\zeta^{bsf} = \arg\min_{i=1,\ldots,N_{tr}} F(\zeta_i^{tr})$;
8: ACO(ζ^{bsf}) — муравьиный алгоритм — см. алгоритм 3;
9: **return** ζ^{bsf}.

В алгоритме HYB находятся $N_{tr} \geq 1$ пробных решений (строки 2–6), затем производится попытка улучшить решение, которое оказалось лучшим среди них. На поиск пробных решений отводится время T_{tr}, а общее время решения задачи ограничивается значением $T_{\max}$. Пробное решение ζ^{tr} представляет собой некоторое допустимое решение задачи; ζ^{bsf} — лучшее решение, найденное на данный момент. Использование пробных решений позволяет уменьшить негативное влияние основного недостатка приближенных методов, заключающегося в частом попадании в одну из субоптимальных областей,

в которой вычислительный процесс может находиться на протяжении многих итераций. Чем больше число пробных решений, тем лучше удается исследовать наиболее перспективные области пространства поиска, что повышает надежность методов.

Для построения исходных решений (строка 3) используются два алгоритма, которые основаны на "жадной" стратегии: метод вставки I1 [17] (далее m1) и эвристика "ближайшего по расстоянию" (m2). Во втором случае вершины упорядочиваются по возрастанию расстояний и выбирается вершина, ближайшая к последней добавленной в текущий маршрут. Локальный поиск (строка 4) улучшает исходные решения, что для многих сложных задач позволяет сразу найти удовлетворительные начальные приближения.

Ниже представлена схема алгоритма локального поиска LocalSearch(ζ).

Алгоритм 2. LocalSearch(ζ)

```
1: input: ζ;
2: N_imp ← K; решение ζ состоит из K маршрутов
3: while (N_imp ≠ 0) do
4:     выбрать случайно маршрут с номером k среди N_imp;
5:     for v = v_1^k → v_{n_k}^k do
6:         for i = 1 → C_ls do
7:             из числа C_ls найти w — ближайший пункт к v;
8:             поиск по окрестности Θ(v, ζ);
9:             if (найдено некоторое ζ^1 ∈ Z и F(ζ^1) < F(ζ)) then
10:                ζ = ζ^1;
11:                перейти на строку 2;
12:            end if
13:        end for
14:    end for
15:    N_imp = N_imp − 1;
16: end while
17: return ζ.
```

Локальный поиск производится из некоторого начального решения ζ. Когда удается найти улучшенное решение, оно запоминается, затем поиск продолжается в окрестности этого нового решения, пока локальный оптимум не будет найден. Как было предложено в [16], мы рассматриваем окрестность $\Theta(v, \zeta)$ как множество решений, которые можно получить вблизи клиента $v \in V^k$ данного маршрута V^k решения ζ. За основу взяты следующие операторы обмена: 2-opt* [18]; Or-Exchange, Relocation и Exchange [19].

2-opt*(v, ζ):

- удалить (v, v^-) и (w, w^+), добавить (v, w) и (v^-, w^+);
- удалить (v, v^+) и (w, w^-), добавить (v, w) и (v^+, w^-).

Out-Relocate(v, ζ):

- вставить v между w и w^-, соединить v^- с v^+;
- вставить v между w и w^+, соединить v^- с v^+.

In-Relocate(v, ζ):

- вставить w между v и v^-, соединить w^- с w^+;
- вставить w между v и v^+, соединить w^- с w^+.

Exchange(v, ζ):

- вставить v между w и $(w^-)^-$, вставить w^- между v^- и v^+;
- вставить v между w и $(w^+)^+$, вставить w^+ между v^- и v^+.

Пункт w является ближайшим по расстоянию к v среди C_{ls} (параметр алгоритма) соседних пунктов. Вершины v и w могут принадлежать как одному, так и разным маршрутам. Для обмена 2-opt*(v, ζ) принадлежность к разным маршрутам необходима. В маршруте посещений пункт v^- непосредственно предшествует пункту v, а за v следует v^+ (аналогично для $w^-,\ w,\ w^+$). В случае, когда w является складом, клиент v удаляется из выбранного маршрута и вставляется в пустой маршрут — увеличивается число ТС. Множества ближайших пунктов ко всем клиентам задачи $\{1, \ldots, N\}$ формируются в начале работы программы, чтобы не расходовать процессорное время на повторные вычисления. Выбор маршрута для улучшения, как и способа изменения решения, происходит случайно. Поиск по окрестности производится по принципу первого улучшения, при котором выгодное изменение производится сразу же.

В алгоритме 3 дана схема работы муравьиного метода ACO(ζ).

Алгоритм 3. ACO(ζ)

input: ζ;
инициализация $\tau_{\max}$, $\tau_{\min}$, матрицы следов феромона T;
while (не превышено время выполнения) **do**
    построение допустимых и недопустимых решений;
    восстановление недопустимых решений;
    локальный поиск для N_{ls} лучших допустимых решений по алгоритму 2;
    if (найдено $\zeta^t \in Z$ такое, что $F(\zeta^t) < F(\zeta)$) **then**
        $\zeta = \zeta^t$;
        пересчет $\tau_{\max}$, $\tau_{\min}$;
    end if
    if (нет улучшения $F(\zeta)$ на протяжении N_{it} итераций) **then**
        $T = \tau_{\max}$;
    else
        обновить T;
    end if
end while
return ζ.

Имитируя поведение колонии муравьев в природе, муравьиные алгоритмы задействуют многоагентные системы. Искусственный муравей (агент) в ACO — это стохастический способ пошагового построения решения с учетом следов феромона. За счет скоординированного взаимодействия агентов удается исследовать наиболее перспективные области пространства решений и находить оптимальные или близкие к оптимальным маршруты за время, приемлемое для практического использования.

Совокупность следов феромона представляется в виде матрицы T размерности $(N+1)^2$ с элементами τ_{ij}. Значения τ_{ij} ограничиваются отрезком $[\tau_{\min}, \tau_{\max}]$, где $\tau_{\max} = 1/[(1-\rho)F(\zeta)]$, $\tau_{\min} = \tau_{\max}/[2(N+1)]$, ρ — заданный коэффициент испарения феромона, $F(\zeta)$ — ЦФ решения ζ, N — число клиентов. Перед первой итерацией следы феромона инициализируются: $T = \tau_{\min}$. Кроме того, на дугах (i, j), входящих в пробное

решение ζ^{tr}, след феромона усиливается: $\tau_{ij} = \tau_{ij} + 1/F(\zeta^{tr})$. При вызове ACO($\zeta^{bsf}$) используется матрица T лучшего пробного решения.

Следы феромона обновляются после каждой итерации t алгоритма ACO. В обновлении T используется лучшее допустимое решение ζ^t на данной итерации. Если в решении ζ^t за вершиной i следует вершина j, то $\tau_{ij}(t+1) = \rho\,\tau_{ij}(t) + 1/F(\zeta^t)$. Для всех ребер, не принадлежащих этому решению, применяется только процедура испарения феромона: $\tau_{ij}(t+1) = \rho\,\tau_{ij}(t)$. Если на протяжении N_{it} итераций не было улучшения $F(\zeta)$, то следы феромона повторно инициализируются следующим образом: $T = \tau_{\max}$.

На каждой итерации муравьи строят $N_{fs} > 0$ допустимых решений и $N_{inf} \geq 0$ недопустимых. Агент отправляется со склада и добавляет в текущий маршрут клиентов до тех пор, пока он их всех не посетит. При выборе следующего пункта для перехода учитывается текущая загруженность. При построении допустимых решений также учитываются временны́е ограничения: разгрузка товара у клиентов не может быть начата вне временно́го окна, а возвращение на склад должно быть не позже $l(0)$. Если ни один из непосещенных клиентов не может быть добавлен в текущий маршрут из-за нарушения ограничений задачи, то начинается новый маршрут со склада.

Пусть J — список вершин, куда может перейти муравей, находясь в вершине i; p — случайное число, равномерно распределенное на отрезке $[0,1]$, которое генерируется на каждом шаге перехода итерации t. Это число сравнивается с параметром p_0, и если $p \leq p_0$, то следующий пункт $j \in J$ для посещения определяется однозначно по формуле $j = \arg\max_{h \in J} \{\tau_{ih}(t) g_{ih}\}$. Здесь и далее соответствующие элементы g_{ij} матрицы G размерности $(N+1)^2$ пересчитываются, если переходы (i,j) встретились в решении муравья. Перед каждой итерацией $g_{ij} = 1\ \forall g_{ij} \in G$. Значения g_{ij} обновляются перед запуском следующего муравья того же типа (допустимого или недопустимого) по следующей формуле: $g_{ij} = g_{ij}\rho_a$. Величина ρ_a является параметром алгоритма. Используются две матрицы G, каждая для своего типа агентов. Если $p > p_0$, то следующий пункт выбирается “методом рулетки”. Величины P_{ij} сопоставляются “секторам колеса рулетки” и обозначают вероятности, с которыми муравей, находясь в данный момент в пункте i, выбирает пункты $j \in J$ для перехода: $P_{ij} = [\tau_{ij}(t) g_{ij}] / \sum_{h \in J} [\tau_{ih}(t) g_{ih}]$.

Для недопустимых решений, найденных муравьями, находится функция штрафа F_{pen}. Пусть ξ — недопустимое решение задачи с нарушениями временны́х ограничений. Функция штрафа $TW(k)$ маршрута (1) решения $\xi \notin Z$ вычисляется по формуле [16]

$$TW(k) = \sum_{j=0}^{n_k+1} \max[\bar{t}^{\,'}(v_j^k) - l(v_j^k), 0],$$

$$\bar{t}(v_0^k) = \bar{t}^{\,'}(v_0^k) = e(0), \quad s(v_0^k) = 0,$$

$$\bar{t}^{\,'}(v_j^k) = \bar{t}(v_{j-1}^k) + s(v_{j-1}^k) + d(v_{j-1}^k, v_j^k), \quad j = 1, \ldots, n_k + 1,$$

$$\bar{t}(v_j^k) = \begin{cases} \max[\bar{t}^{\,'}(v_j^k), e(v_j^k)], & \text{если } \bar{t}^{\,'}(v_j^k) \leq l(v_j^k), \\ l(v_j^k), & \text{если } \bar{t}^{\,'}(v_j^k) > l(v_j^k), \end{cases}$$

а функция штрафа решения ξ определяется как сумма штрафных функций по всем K маршрутам: $F_{pen}(\xi) = \sum_{k=1}^{K} TW(k)$. Если $l(v_j^k) < \bar{t}^{\,'}(v_j^k)$, то момент начала разгрузки товара у клиента v_j^k (или когда ТС возвращается на склад, если $j = n_k + 1$) происходит

после самого позднего срока, когда это разрешается. В этом случае будем считать, что опоздания не было и происходит "возврат" к моменту $l(v_j^k)$ с целью без задержки начать разгрузку или вернуться на склад, но со штрафом $\bar{t}^{'}(v_j^k) - l(v_j^k)$.

Для решений с $F_{pen}(\xi) > 0$ используется процедура восстановления допустимости — алгоритм минимизации $F_{pen}(\xi) + F(\xi)$, основанный на поиске по окрестности, который работает по похожему принципу, что и алгоритм 2 локального поиска. На вход подается недопустимое решение ξ. Случайным образом выбирается маршрут, в котором нарушены временны́е ограничения. Далее решается задача поиска лучшего решения $\xi^{'}$ в окрестности $\Theta(v, \xi)$ при условии, что $\triangle F_{pen} = F_{pen}(\xi^{'}) - F_{pen}(\xi) < 0$. Рассматриваются C_{rep} ближайших пунктов к v. Если лучшее решение не может быть найдено (функция штрафа не уменьшается), то поиск прекращается, решение ξ остается недопустимым и оно отбрасывается. Поиск производится по принципу наискорейшего спуска, когда просматривается вся окрестность и выбирается решение с наименьшей стоимостью. Допустимость восстановлена, если $F_{pen}(\xi) = 0$. Локальный поиск выполняется только для N_{ls} лучших допустимых решений.

Просмотр окрестности требует больших вычислительных затрат. В работе [16] предложен способ, в котором величина штрафа для новых маршрутов находится за время $O(1)$ с использованием операторов 2-opt*, Out-Relocate, In-Relocate, Exchange, если изначально пункты v и w принадлежат разным маршрутам. Для операторов в пределах одного маршрута (всех из перечисленных выше, кроме 2-opt*) это условие не выполняется и временна́я сложность алгоритма в худшем случае равна $O(n_k)$, где n_k — число клиентов, назначенных на маршрут с номером k. Этот же способ используется в настоящей работе.

3. Результаты вычислительных экспериментов

Для настройки параметров алгоритма HYB и оценки его эффективности решались 168 тестовых задач размерностей $N = 25, 50, 100$ из стандартного набора [17]. Задачи каждой размерности распределены по классам. Классы C1(9), R1(12), RC1(8) включают задачи с краткосрочным горизонтом планирования, а классы C2(8), R2(11), RC2(8) — с долгосрочным. Для задач первого типа характерно посещение только нескольких клиентов одним и тем же ТС, в задачах второго типа соотношения грузоподъемности ТС, объемов заказов клиентов, предельного времени возвращения на склад позволяют назначать на один маршрут многих клиентов (до 60 в некоторых случаях). Обозначения C, R и RC указывают на соответственно кластерное, случайное и смешанное расположение клиентов; в скобках приведено число задач в каждом классе.

К 2012 г. точными методами были найдены решения всех задач из тестового набора [17]. В настоящей работе, как и в случае с точными методами [5–7], мы оперируем числовыми данными целого типа. Для этого элементы целочисленной матрицы расстояний с координатами (x_i, y_i) и (x_j, y_j) соответствующих пунктов рассчитываются по формуле $D_{ij} = \mathrm{int}(10\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2})$, где int — функция преобразования к целому типу с отбрасыванием дробной части числа. Входные данные $[e(i), l(i)]$, $s(i)$, $i = 0, \ldots, N$, преобразуются к целочисленным значениям аналогичным образом.

Для любой задачи, возникающей на практике, по координатам, временны́м окнам и другим входным данным можно заранее определить класс задач, к которому она относится. Вследствие этого с целью повышения эффективности решения задач целе-

сообразно найти подходящие параметры алгоритмов для каждого класса и соответствующих размерностей $N = 25, 50, 100$.

Параметры алгоритма HYB для задач $N = 25$: $N_{fs} = 3$, $N_{inf} = 7$, $N_{ls} = 10$, $N_{it} = 10$, $N_{tr} = 5$, $T_{tr} = 25$. Значения C_{ls}, C_{rep} для классов указаны в скобках после их названий: C1 и RC2 (15,10); C2 (15,5); R1 (5,10); R2 (10,5); RC1 (5,5). Для построения исходных решений задач класса C1 используется алгоритм m2, для остальных — метод m1 со значениями μ-λ-α_1-α_2 [17]: C2 и R2 — 1-1-0-1; R1 — 1-1-1-0; RC1 и RC2 — 1-2-0-1.

Параметры алгоритма HYB для задач $N = 50$: $N_{fs} = 5$, $N_{inf} = 10$, $N_{ls} = 15$, $N_{it} = 30$, $N_{tr} = 5$, $T_{tr} = 500$. Лучшие значения C_{ls} и C_{rep} следующие: C1 и RC2 (20,10); C2 (30,10); R1 (10,30); R2 (10,10); RC1 (10,20). Метод m1 применялся для классов C1, C2 и R1 с 1-1-1-0; для классов R2 и RC1 с 1-2-0-1. Для RC2 использовался алгоритм m2.

Параметры алгоритма HYB для задач $N = 100$: $N_{fs} = 20$, $N_{inf} = 20$, $N_{ls} = 20$. По классам C1 и C2 результаты получены со значениями $N_{it} = 50$, $C_{ls} = C_{rep} = 20$, а по R1, R2, RC1, RC2 — со значениями $N_{it} = \infty$, $C_{ls} = C_{rep} = 40$. Для построения исходных решений задач класса C2 применялся метод m1 с 1-1-0-1. Для всех остальных — алгоритм m2.

Одинаковыми для всех классов и размерностей N были значения параметров $\rho = 0.1$, $p_0 = 0.9$, $\rho_a = 0.9$. Алгоритм HYB является стохастическим, поэтому каждая задача решалась N_{run} раз. Значения $N_{run} = 500, 500, 100, 25$ были выбраны соответственно для задач $N = 25$, $N = 50$, кластерных задач $N = 100$, для задач остальных классов размерности $N = 100$ (в этом случае время решения ограничивалось значением $T_{\max}$). Всюду время дано в секундах. В случае обнаружения оптимального решения поиск прекращался.

Все результаты получены на вычислительных ресурсах [20] с использованием процессора Intel Xeon E5450 3.0 ГГц 2007 г. выпуска. В дальнейшем при сравнении времени счета с опубликованными результатами других авторов применялись коэффициенты производительности процессоров, найденные по результатам тестов SPECint_base2006 и SPECint_rate_base2000 (`https://www.spec.org/`). В JPSP [5] использовался процессор Intel Pentium 4 3.0 ГГц, коэффициент его производительности по отношению к Intel Xeon E5450 3.0 ГГц составил $c_1 = 0.32$. Соответственно для BMR [6] (Intel Xeon X7350 2.93 ГГц 2007 г. выпуска) — $c_2 = 0.89$ и PCDU [7] (Intel Xeon E5-2637 v2 3.5 ГГц 2014 г. выпуска) — $c_3 = 2.28$.

В табл. 1 приведены обобщенные результаты решения задач всех классов размерностей $N = 25, 50$. Все задачи в каждом запуске были решены методом HYB. Показаны максимальное $t_{\max}$ и среднее t_{avg} времена решения, найденные по общему количеству

Т а б л и ц а 1. Время решения задач, с

Класс (N_p)	$N = 25$			$N = 50$			
	HYB		JPSP	HYB		JPSP	BMR
	$t_{\max}$	t_{avg}	$t_{avg}c_1$	$t_{\max}$	t_{avg}	$t_{avg}c_1$	$t_{avg}c_2$
C1 (9)	0.1	0.005	0.22	1.7	0.09	56	—
C2 (8)	4.2	0.07	0.85	23	0.48	25 (7)	7
R1 (12)	1.1	0.06	0.05	511	16	44	—
R2 (11)	3.7	0.16	1.20	952	28	2268 (9)	110
RC1 (8)	1.7	0.09	0.08	202	7.3	13	—
RC2 (8)	0.3	0.03	3.37	579	16	86 (7)	24

запусков всех задач N_p одного класса: $N_{prun} = N_p N_{run}$. Для сравнения в этой таблице также представлены опубликованные результаты решения этих задач точными методами JPSP [5] и BMR [6] с коэффициентами c_1, c_2. В тех случаях, когда не все задачи в классе были решены методом JPSP, число решенных указано в скобках. Из табл. 1 можно видеть, что в среднем методом HYB получены результаты, существенно превосходящие опубликованные в литературе для задач $N = 25, 50$. В настоящее время такие задачи не считаются сложными, однако необходимо уметь их быстро решать: задачи небольшой размерности часто встречаются на практике.

В табл. 2 представлены результаты решения задач всех классов размерности $N = 100$ приближенными методами HYB, CGH [21] и точными JPSP [5], BMR [6], PCDU [7]. Для упрощения изложения методы с $T_{\max} = 3600$ помечены окончанием (1h), а с $T_{\max} = 14400$ — (4h). Для алгоритма CGH(1h) приведены результаты по трем запускам. Для HYB(1h) выбраны $N_{tr} = 5$, $T_{tr} = 1800$; для HYB(4h) использовались $N_{tr} = 3, 3, 5, 8, 16$, $T_{tr} = 500, 900, 1800, 3600, 7200$ соответственно. В табл. 2 показаны $\delta_{\max}$ и δ_{avg} — максимальная и средняя относительные погрешности ЦФ, найденные по N_{prun} запускам, $\delta = 100\,\%(F(\zeta) - F(\zeta^*))/F(\zeta^*)$, где $F(\zeta^*)$ — ЦФ известного оптимального решения. Для точных методов приводится среднее (по классам) время решения задач. Для метода HYB(4h) указано среднее время решения только тех задач, для которых получен оптимальный результат в каждом запуске (число таких задач указано в скобках). Остальные задачи оцениваются по погрешности δ_{avg}.

В табл. 3, 4 представлены результаты для отдельных задач. В этот список включены задачи, которые являются сложными для точных методов согласно [7], а также

Т а б л и ц а 2. Результаты решения всех задач, $N = 100$

Класс (N_p)	HYB(1h)		CGH(1h)	HYB(4h)		JPSP	BMR	PCDU
	$\delta_{\max}$	δ_{avg}	δ_{avg}	δ_{avg}	t_{avg}	$t_{avg}c_1$	$t_{avg}c_2$	$t_{avg}c_3$
C1 (9)	0	0	0	0	2.2	150	22	34
C2 (8)	0	0	0.05	0	8.8	894 (7)	36	748
R1 (12)	1.6	0.16	0.6	0.09	1027 (9)	8772	223	71
R2 (11)	1.3	0.28	6.4	0.16	1618 (4)	11293 (4)	25525 (10)	14665
RC1 (8)	1.2	0.22	0.8	0.05	2234 (3)	3521	246	119
RC2 (8)	1.4	0.17	3.9	0.07	1889 (4)	1025 (5)	3353	768

Т а б л и ц а 3. Время решения сложных задач, $N = 100$, с

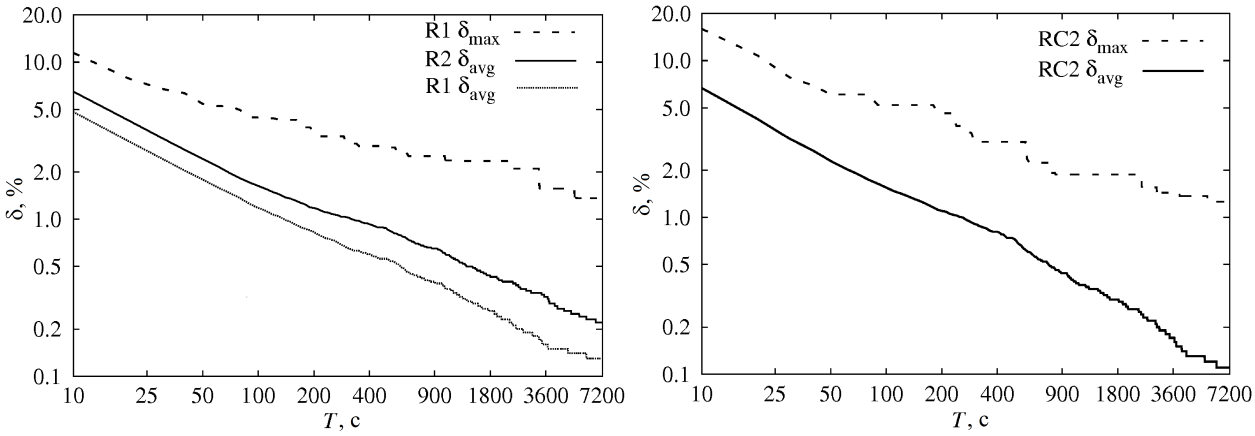
Задача	HYB(4h)			JPSP	BMR	PCDU
	$t_{\max}$	$t_{\min}$	t_{avg}	$t_{exact}c_1$	$t_{exact}c_2$	$t_{exact}c_3$
C203	58	0.3	14	4182	28	832
C204	126	11	44	—	162	3757
R201	6768	328	3909	44	160	—
R202	712	34	193	2650	3031	720
R203	527	48	177	17340	1904	1288
R204	7506	85	2194	—	192567	3409
RC203	9068	322	3347	4773	224	—
RC205	5091	149	1516	71	153	—
RC206	6722	116	1516	108	202	—
RC207	2885	95	1177	—	23944	791

Т а б л и ц а 4. Погрешность решения сложных задач, $N = 100$

Задача	HYB(1h)			CGH(1h)		HYB(4h)	
	$\delta_{\max}$	$\delta_{\min}$	δ_{avg}	$\delta_{\min}$	δ_{avg}	$\delta_{\min}$	δ_{avg}
R205	1.2	0	0.42	2.4	3.9	0	0.28
R206	1.1	0	0.26	6.7	7.8	0	0.08
R207	1.3	0	0.30	8.3	9.4	0	0.08
R208	1.3	0	0.39	6.1	7.9	0	0.06
R209	0.1	0	0.11	3.9	5.6	0	0.04
R210	1.1	0.03	0.83	5.7	6.1	0	0.64
R211	0.9	0.54	0.69	9.2	10	0	0.55
RC201	0.6	0	0.12	1.0	1.5	0	0.04
RC202	0.4	0	0.13	2.5	4.3	0	0.01
RC204	0.5	0	0.27	3.1	4.9	0	0.21
RC208	1.4	0.05	0.63	5.7	7.0	0	0.33

те, которые вообще считаются сложными для решения — все задачи классов R2, RC2. Отобранные таким образом задачи разделены на 2 группы: те, что решены HYB(4h) в каждом запуске, и для них указаны максимальное $t_{\max}$, минимальное $t_{\min}$ и среднее t_{avg} времена решения (табл. 3), и те, что не были решены в каждом запуске, и для них приведены максимальная $\delta_{\max}$, минимальная $\delta_{\min}$ и средняя δ_{avg} погрешности (табл. 4). Время решения точными методами обозначено как t_{exact}. Погрешности найдены по $N_{run} = 25$ запускам решения задач методами HYB(1h), HYB(4h) и по трем запускам — методом CGH(1h).

Отметим, что без ослабления ограничений ($N_{inf} = 0$) результаты решения тестовых задач оказались в целом хуже. В случае задач размерности $N = 25$ среднее время решения осталось на прежнем уровне, кроме одной задачи RC102, для которой среднее время решения возросло в 580 раз. Для задач $N = 50$ среднее время решения по классу C1 осталось на прежнем уровне, по классам C2 и R2 увеличилось соответственно в 1.2 и 1.8 раза, а в классах R1, RC1, RC2 при ограничении времени решения $T_{\max} = 3600$ не все задачи были решены в каждом запуске. При решении задач размерности $N = 100$ погрешность нахождения оптимальных значений ЦФ методом HYB(1h) без ослабления ограничений увеличилась более чем в 2 раза.



Зависимость погрешностей от времени решения задач классов R1, R2, RC2

Включение методов локального поиска для исходных решений (строка 4 в алгоритме 1) позволило улучшить результаты решения задач $N = 100$ классов RC1 и RC2 в среднем более чем на 30 %. Эффективность решения оставшихся задач размерностей $N = 25, 50, 100$ без включения локального поиска для исходных решений осталась на том же уровне.

Проведенные вычислительные эксперименты показали, что алгоритм HYB наиболее эффективно решает задачи кластерного типа. По сравнению с результатами, полученными нами в [15], удалось улучшить среднее время решения задач $N = 25$ классов C1 и C2 в 4 и 2.4 раза соответственно; задач $N = 50$ классов C1 и C2 — в 2 и 1.3 раза; задач $N = 100$ классов C1 и C2 — в 2 и 1.6 раза.

Задачи классов C2, R2, RC2 с долгосрочным горизонтом планирования считаются более сложными для решения. Это подтверждается результатами многих авторов: число решенных задач из этих классов меньше и время их решения значительно хуже, чем для задач классов C1, R1, RC1. Однако для алгоритма HYB такого различия нет: результаты примерно на одном уровне. Отметим, что все задачи размерности $N = 100$ хотя бы в одном из запусков были решены методом HYB.

На рисунке показано уменьшение погрешностей $\delta_{\max}$, δ_{avg} (найденных по результатам числа запусков N_{prun} при $N_{run} = 25$) в процессе решения задач классов R1, R2, RC2 ($N_{tr} = 8$, $T_{tr} = 3600$). Временно́й отрезок от 0 до 7200 был разбит на интервалы по 10 с, и учитывалось ближайшее по времени к конечной точке каждого интервала одно значение δ.

Заключение

Для решения задачи маршрутизации транспорта с ограничениями по временны́м окнам (VRPTW) разработан и реализован приближенный гибридный алгоритм, в котором использование пробных решений позволяет исследовать наиболее перспективные области пространства поиска и уменьшает вероятность преждевременной сходимости к субоптимальным решениям. В составе гибридного метода предложен муравьиный алгоритм с ослаблением временны́х ограничений: при построении решений некоторыми муравьями эти ограничения не учитываются. Метод восстановления недопустимых решений основан на поиске по окрестности с эффективной процедурой ее просмотра.

Результаты проведенных вычислительных экспериментов показали следующее.

1. Все 112 задач размерностей 25 и 50 были решены в каждом запуске. Из всех 56 задач размерности 100 решены в каждом запуске 66 %, а оставшиеся решены хотя бы в одном. Относительные погрешности нахождения оптимальных значений целевой функции для самых сложных задач уменьшаются с увеличением времени решения.

2. По отдельным задачам имеются большие различия с результатами, полученными другими авторами. Многие задачи решены предложенным в настоящей работе алгоритмом существенно быстрее.

3. Предложенный алгоритм оказался наиболее эффективным для кластерных задач.

4. Эффективность решения задач с краткосрочным и долгосрочным горизонтами планирования оказалась примерно на одном уровне. Этот результат отличается от опубликованных данных других авторов: их результаты решения задач с долгосрочным горизонтом планирования значительно хуже, чем с краткосрочным.

5. Предложенный метод может рассматриваться в качестве дополнительного или альтернативного для решения задач VRPTW кластерного типа и задач с долгосрочным горизонтом планирования.

Благодарности. Для численных экспериментов использованы вычислительные ресурсы ЦКП "Центр данных ДВО РАН" (`http://lits.ccfebras.ru/`).

Список литературы / References

[1] **Desaulniers, G., Madsen, O.B.G., Ropke, S.** The vehicle routing problem with time windows // Vehicle Routing: Problems, Methods, and Applications. 2nd Edition / P. Toth, D. Vigo (Eds). SIAM: MOS-SIAM Series on Optimization, 2014. P. 119–159.

[2] **Cordeau, J.-F., Desaulniers, G., Desrosiers, J.** VRP with time windows // The Vehicle Routing Problem / P. Toth, D. Vigo (Eds). SIAM: MOS-SIAM Series on Optimization, 2002. P. 157–193.

[3] **Vidal, T., Crainic, T.G., Gendreau, M., Prins, C.** Time-window relaxations in vehicle routing heuristics // J. of Heuristics. 2015. Vol. 21, No. 3. P. 329–358.

[4] **Хмелев А.В.** Трехфазный алгоритм оптимизации автопарка и маршрутов транспортных средств // Дискретный анализ и исследование операций. 2015. Т. 22, № 6. С. 55–77.
Khmelev, A.V. A three-phase heuristic for the vehicle fleet and route optimization // Diskretnyi Analiz i Issledovanie Operatsii. 2015. Vol. 22, No. 6. P. 55–77. (In Russ.)

[5] **Jepsen, M., Petersen, B., Spoorendonk, S., Pisinger, D.** Subset-row inequalities applied to the vehicle-routing problem with time windows // Operations Research. 2008. Vol. 56(2). P. 497–511.

[6] **Baldacci, R., Mingozzi, A., Roberti, R.** New route relaxation and pricing strategies for the vehicle routing problem // Operations Research. 2011. Vol. 59(5). P. 1269–1283.

[7] **Pecin, D., Contardo, C., Desaulniers, G., Uchoa, E.** New enhancements for the exact solution of the vehicle routing problem with time windows // INFORMS J. on Computing. 2017. Vol. 29, No. 3. P. 489–502.

[8] **Сластников С.А.** Применение метаэвристических алгоритмов для задачи маршрутизации транспорта // Экономика и матем. методы. 2014. Т. 50, № 1. С. 117–126.
Slastnikov, S.A. Application of metaheuristic algorithms for the vehicle routing task // Economics and Math. Methods. 2014. Vol. 50, No. 1. P. 117–126. (In Russ.)

[9] **Baldacci, R., Mingozzi, A., Roberti, R.** Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints // Europ. J. of Operational Res. 2012. Vol. 218. P. 1–6.

[10] **Kallehauge, B.** Formulations and exact algorithms for the vehicle routing problem with time windows // Computers & Operations Res. 2008. Vol. 35. P. 2307–2330.

[11] **Dorigo, M., Stützle, T.** Ant colony optimization. Cambridge, MA: MIT Press, 2004. 306 p.

[12] **Кажаров А.А., Курейчик В.М.** Муравьиные алгоритмы для решения транспортных задач // Изв. РАН. Теория и системы управления. 2010. Т. 49, № 1. С. 32–45.
Kazharov, A.A., Kureichik, V.M. Ant colony optimization algorithms for solving transportation problems // J. of Comput. and Syst. Sci. Intern. 2010. Vol. 49, No. 1. P. 30–43.

[13] **Долгова О.Э., Пересветов В.В.** Лучевой поиск и муравьиный алгоритм в решении задачи маршрутизации транспорта // Информатика и системы управления. 2016. Т. 48, № 2. С. 47–57.
Dolgova, O.E., Peresvetov, V.V. Beam search with ant colony optimization algorithm for solving the vehicle passage problem // Informatika i Sistemy Upravleniya. 2016. Vol. 48, No. 2. P. 47–57. (In Russ.)

[14] **Долгова О.Э., Пересветов В.В.** Задача маршрутизации транспортных средств с временны́ми окнами // Матер. Всерос. науч.-практ. конф. "Информ. технологии и высокопроизводительные вычисления". Хабаровск, 2013. С. 119–124.
Dolgova, O.E., Peresvetov, V.V. The vehicle routing problem with time windows // Information Technologies and High Performance Computations: Proc. All Russ. Sci. and Appl. Conf. Khabarovsk, 2013. P. 119–124. (In Russ.)

[15] **Долгова О.Э., Пересветов В.В.** Муравьиный алгоритм и метод локальных улучшений в решении задач маршрутизации транспорта с временны́ми окнами кластерного типа // Тр. IV Всерос. науч.-практ. конф. "Информ. технологии и высокопроизводительные вычисления". Хабаровск: ТОГУ, 2017. С. 50–54.
Dolgova, O.E., Peresvetov, V.V. An ant colony optimization algorithm and local search in solving vehicle routing problems of the cluster type with time windows // Information Technologies and High Performance Computations: Proc. All Russ. Sci. and Appl. Conf. Khabarovsk: TOGU, 2017. P. 50–54. (In Russ.)

[16] **Nagata, Y., Bräysy, O., Dullaert, W.** A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows // Computers and Operations Res. 2010. Vol. 37. P. 724–737.

[17] **Solomon, M.M.** Algorithms for the vehicle routing and scheduling problem with time window constraints // Operations Res. 1987. Vol. 35. P. 254–265.

[18] **Potvin, J.-Y., Rousseau, J.-M.** An exchange heuristic for routeing problems with time windows // J. of the Operational Res. Soc. 1995. Vol. 46. P. 1433–1446.

[19] **Kindervater G.A.P., Savelsbergh M.W.P.** Vehicle routing: handling edge exchanges // Local Search in Combinatorial Optimization / E.H. Aarts, J.K. Lenstra (Eds). Chichester, U.K.: John Wiley & Sons, 1997. P. 311–336.

[20] Центр коллективного пользования "Центр данных ДВО РАН". Адрес доступа: `http://lits.ccfebras.ru` (дата обращения 08.11.2017).
Shared Facility Center "Data Center of FEB RAS" Available at: `http://lits.ccfebras.ru` (accessed 08.11.2017) (In Russ.)

[21] **Alvarenga, G.B., Mateus, G.R., de Tomi, G.** A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows // Computers & Operations Res. 2007. Vol. 34. P. 1561–1584.

Поступила в редакцию 19 января 2018 г.,
с доработки — 18 июня 2018 г.

An ant colony optimization algorithm with time window constraints relaxation for solving the vehicle routing problem

Dolgova, Olga E., Peresvetov, Vladimir V.*

Computing center of Far Eastern Branch of the RAS, Khabarovsk, 680000, Russia
*Corresponding author: Peresvetov, Vladimir V., e-mail: `peresvv@mail.ru`

The purpose of this paper is to improve the performance of a hybrid method based on ant colony optimization (ACO) that finds approximate solutions of the vehicle routing problem with time windows (VRPTW). In order to solve this problem it is required

to design a plan for goods delivery to the customers generating the routes of identical vehicles so that the total travelled distance is minimal.

For the VRPTW solving, the hybrid method is developed in which a usage of trial solutions makes it possible to explore the most promising parts of the search space. The initial methods for solution construction, an ant colony optimization (ACO) algorithm and local search are proposed in the framework of the hybrid method. In the ACO algorithm, when generating the routes, it is allowed to violate the time window constraints. A method to restore the feasibility of solutions is implemented within the relaxation scheme under "returns in time" principle.

Numerical results for solving all problems with 25, 50 and 100 customers from the Solomon test set are obtained. We provide the results on the time and deviation of the solution of these problems in comparison with the results of other authors. Some problems and their classes were solved much faster by the algorithm proposed in this paper. Relative deviations from optimal values of the objective function for the most complex tasks decrease with increasing decision time.

The proposed approach can be considered to be an additional or an alternative algorithm for solving the cluster type and the long-term planning horizon problems of the VRPTW.

Keywords: vehicle routing, time windows, total travelled distance, hybrid algorithm, ant colony optimization, local search.

Cite: Dolgova, O.E., Peresvetov, V.V. An ant colony optimization algorithm with time relaxed window constraints for solving the vehicle routing problem // Comput. Technologies. 2018. Vol. 23, No. 5. P. 49–62. (In Russ.) DOI: 10.25743/ICT.2018.23.5.005.

Acknowledgements. The resources of the Shared Facility Center "Data Center of FEB RAS" (`http://lits.ccfebras.ru/`) were used for numerical experiments.

Received 19 January 2018
Received in revised form 18 June 2018