“Исследование настроечных параметров алгоритма поиска с запретами и его модификация для решения курьерской задачи маршрутизации.”

L. Lyubchik, R. Shafeyev

В статье рассматривается задача маршрутизации курьерской доставки с долгим сроком службы, для которой была построена дискретная модель и вычислительная схема, в основу которой был положен алгоритм поиска с запретами. Эффективность схемы предложенного алгоритма была проверена на задачах большой размерности, которые были составлены на базе известных модельных задачах маршрутизации на транспорте различных классов.

Introduction

Одним из средств экономии ресурсов при транспортировке грузов является применение систем поддержки принятия решений в сфере транспортной логистики. Одной из ключевых функций систем поддержки принятия решений в сфере транспортной логистики является возможность расчета и построения эффективных с точки зрения стоимости объезда маршрутов различного назначения на транспортной сети. Математическая формулировка этой задачи широко известно как задача маршрутизации транспорта. Существует ряд разновидностей ЗМТ с различными дополнительными условиями, которые позволяют учитывать грузоподъемность транспортных средств и другие ограничения для более полного представления деталей реальной действительности.

Данная статья посвящена рассмотрению задачи курьерской маршрутизации, которая является подзадачей ЗМТ. Данная задача состоит в нахождении маршрутов для посещения заданного множества адресов некоторым количеством единиц транспортных средств. Такая проблема наиболее часто встречается среди компаний, выполняющих развозки товара от отправителя к получателю.

Автоматизация процесса планирования маршрутов будет актуальной для: интернет-магазинов, крупных оптовых компаний (например, для развозки товаров, которые быстро портятся или оптимизации схемы доставки товаров в пункты назначения), фирм с целью организации допоставок в магазины. Практическое применение задачи можно встретить в сфере транспортировки пациентов между корпусами поликлиники[16, 18], доставке газет и журналов, доставки топлива в частные дома [19]. Решение рассматриваемой задачи также имеет практическое применение в системах вертолетных перевозок людей между морскими нефтяными платформами [17].

В последнее время все больше людей пользуются услугами курьерских служб. Обычно заявки формируются днем ранее и перед диспетчером полный

список заявок, которые необходимо выполнить в кратчайшие сроки. Тематическое исследование провел Attanasio(2007) [15], в котором описал преимущества использования компьютерных методов перед человеческой диспетчеризацией. В своей работе он рассмотрел примеры работы eCourier Ltd, лондонской компании, которая предлагает курьерские услуги. Их клиентами в основном являются юридические фирмы, финансовые учреждения, рекламные агентства и другие организации, которые заинтересованы в быстрой доставке товаров или оригинальных подписях на документах. После того как все заявки на конкретный момент времени сформированы, необходимо составить такой маршрут, который удовлетворял бы требованиям заказчиков. В зависимости от уровня обслуживания, который был задан клиентом, курьер может объединить или ни его доставку с другими, то есть это означает, что товары могут быть доставлены по очереди, или курьер может выполнить ускоренную доставку.

Компании, которые предлагают курьерские услуги достаточно часто имеют смешанный парк транспортных средств, который состоит из велосипедов, мотоциклов, автомобилей и небольших грузовиков. В зависимости от типа запроса, расположение и временных окон будет выбран тот вид транспортного средства, который сможет выполнить заказ в кратчайшие сроки. Исследования [15] показали, что использование компьютерных методов, в том числе алгоритмов оптимизации, были очень выгодными для курьерских компаний.

Таким образом, использование автоматизированной системы позволяет улучшить качество обслуживания, время доставки, повысить курьерскую эффективность и сократить расходы на доставку, тем самым обеспечить повышение конкурентного преимущества.

Постановка задачи

Пусть $C, dim(C) = n$ – множество транспортных средств, $Q, dim(Q) = m$ – множество запросов клиентов, которые требуется обработать на текущий момент времени.

Предположим, что следующая информация известна о транспортных средствах из множества C:
$\vec{P}_c$ – местоположение ТС, $c \in C$;
L_c – грузоподъемность ТС, $c \in C$.

Пусть S – множество отправителей груза, $(dim(S) = dim(Q) = m)$, R – множество получателей, $dim(R) = dim(Q) = m$. Тогда каждый клиентский запрос $q \in Q$ включает в себя следующую информацию:
s_q – отправитель груза клиента, $s \in S$;
r_q – получатель клиентского груза, $r \in R$;
$\vec{P}_{s_q}$ – Местоположение отправителя;
$\vec{P}_{r_q}$ – Местоположение получателя;
w_q – клиентский груз, который требуется перевезти от отправителя к получателю;
$[t_s^q, t_s^q + \Delta t_s^q]$ – временное окно, в течение которого нужно забрать груз у отпра-

вителя;
$[t_r^q, t_r^q + \Delta t_r^q]$ – временное окно, в течение которого нужно доставить груз получателю.

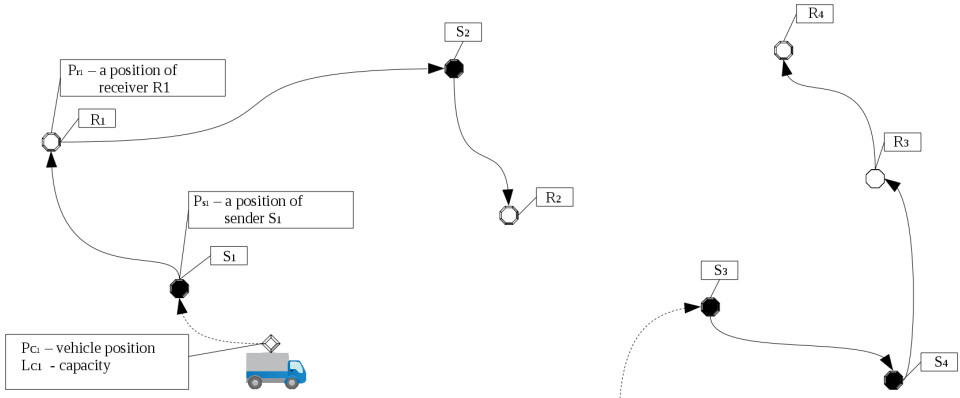
Таким образом, каждая заявка может быть представлена в виде кортежа:

$$\forall q \in Q : \exists q = (s_q, r_q, \vec{P}_{s_q}, \vec{P}_{r_q}, w_q, t_s^q, \Delta t_s^q, t_r^q, \Delta t_r^q) \tag{1}$$

Для того, чтобы оценить стоимость перевозки между пунктами назначения, определим функцию стоимости Ω:

$$\forall i, j \in S \cup R : \exists \Omega_{i,j} = \Omega(\vec{P}_i, \vec{P}_j) \tag{2}$$

Необходимо построить оптимальные маршруты движения транспортных средств для перевозки грузов от отправителя к получателю для всех клиентских запросов.



C_i	Vehicle
P_{C_i}	Vehicle position
S_q	Sender
P_{s_q}	Sender posinion
R_q	Receiver
P_{r_q}	Receiver position
W_q	Client shipment that is required to deliver from the sender

Рис. 1. Example of the Routing Delivery Problem.

Дискретная модель

Задачу маршрутизации можно представить в виде ориентированного графа $G = G(V, E)$. Пусть множество $V = C \cup S \cup R$ – узлы графа G, состоящее из элементов множеств транспортных средств C, отправителей S и получателей R. E – динамическое множество дуг графа G, такое, что:

$$\forall e(X) \in E : e(X) = (v_i, v_j), \exists v_i \in V, v_j \in V/C \tag{3}$$

Пусть $X = \{X^k\}_{k=1}^n$ представляет собой последовательность матриц для каждого ТС $k \in C$. Элементы матрицы принимают следующие значения:

$$x_{i,j}^{(k)} = \begin{cases} 1, & \text{АТС } k \in C \text{ движется с узла } i \text{ в вершину } j \\ 0, & \text{в противном случае.} \end{cases} \tag{4}$$

где: $i \in V/C \cup k, j \in V/C$.

Введем вектор переменных $\vec{Y}^k(X)$ для каждого транспортного средства $k\ inC$. Векторные элементы принимают следующие значения:

$$\vec{y}_j^{(k)}(X) = \begin{cases} 1, & \text{Запрос } j \in Q \text{ обслуживается ТС } k \in C \\ 0, & \text{в противном случае} \end{cases} \tag{5}$$

где: $i \in V/C \cup k, j \in V/C$.

Пусть $t_j^k(X^{(k)})$ – время прибытия автомобиля $k \in C$ в пункт назначения $j \in S \cup R$.

Целевая функция принимает следующий вид:

$$F(X) = \sum_{k \in C} \sum_{i,j \cup V} \Omega_{ij} \cdot x_{ij}^{(k)} \to min \tag{6}$$

Наложим ограничения на целевую функцию (6) для обеспечения непрерывности маршрутов:

$$\sum_{k \in C} \sum_{j \in S \cup R} x_{i,j}^{(k)} \leq 1, \forall i \in V \tag{7}$$

$$\sum_{k \in C} \sum_{i \in S \cup R \cup \{k\}} x_{i,j}^{(k)} = 1, \forall j \in S \cup R \tag{8}$$

$$\sum_{i \in S \cup R \cup \{k\}} x_{i,\omega}^{(k)} - \sum_{j \in S \cup R} x_{\omega,j}^{(k)} \leq 1, \forall \omega \in S \cup R, \forall k \in C \tag{9}$$

$$\sum_{i \in S \cup R/Z} \sum_{j \in Z} x_{i,j}^{(k)} > 0, Z = \{z \in Z : \sum_{j \in S \cup R} x_{j,z}^{(k)} > 0\}, \forall k \in C \tag{10}$$

Ограничение (7) запрещает, чтобы у узла графа G было более чем одна выходящая дуга. Ограничение (8) запрещает узлу иметь более чем одну входную

дугу. Ограничение (9) указывает на то, что у узла количество входных дуг не может быть меньше количества выходных дуг (это ограничение учитывает тот факт, что транспортное средство может покинуть пункт назначения, если только он посетил этот узел). Ограничение (10) исключает локальные циклы.

Следующие ограничения синхронизируют значения переменных X и $\bar{}$для каждого запроса Q Q и запрещают посещение получателя перед отправителем:

$$x_{s_q}^{(k)} + x_{r_q}^{(k)} = 2 \cdot y_q^{(k)}(X), \forall k \in C, q \in Q \tag{11}$$

$$y_q^{(k)}(X) \cdot (\tilde{t}_{r_q}^k(X^{(k)}) - \tilde{t}_{s_q}^k(X^{(k)}) \geq 0, \forall k \in C, q \in Q \tag{12}$$

Определим ограничения для учета грузоподъемности транспортных средств и временные окна:

$$\sum_{j \in Q} \omega_j \cdot y_j^k \leq L_k, \forall k \in C \tag{13}$$

$$t_s^q \leq \tilde{t}_{s_q}^k(X^{(k)} \leq t_s^q + \Delta t_s^q, \forall q \in Q, \tag{14}$$

$$t_r^q \leq \tilde{t}_{r_q}^k(X^{(k)} \leq t_r^q + \Delta t_r^q, \forall q \in Q, \tag{15}$$

Описание алгоритма

Представим множество матриц $\{X^k\}$ в виде вектора, определенного на гиперкубе $E^\eta = \{0.1\}^\eta$, где $\eta = n \cdot (2m+1) \cdot 2m$. Цель задачи состоит в минимизации следующей целевой функции:

$$F(\vec{u}) \to min, \vec{u} \in E^\eta \tag{16}$$

Обозначим через $\delta(\vec{u}, \vec{v})$ расстояние Хэмминга между $\vec{u}$ и $\vec{v}$. Через $N_l(\vec{u})$ обозначим окрестность точки $\vec{u}$ дариусом l [4]:

$$N_l(\vec{u}) = \{\vec{v} \in E^\eta : \delta(\vec{u}, \vec{v}) \leq l\}, l = 1, \bar{}\eta \tag{17}$$

При $= ETA$ множество $N_l(\ VEC)$ для любого вектора VEC совпадает с множеством E и быть в этой окрестности вектора с минимальным значение целевой функции эквивалентна решению исходной задачи. Стандартный алгоритм Локальный алгоритм поиска начинается с выбранного случайным образом вектор VEC^0. На I шаг алгоритма перехода от текущего вектора в соседней, что минимальное значение целевой функции в окрестности данного вектора:

Когда $l = \eta$, тогда множество $N_l(\vec{u})$ для любого вектора $\vec{u}$ совпадает с множеством E^η и поиск в этой окрестности минимума эквивалентно решению исходной задачи. Классический алгоритм локального поиска начинает поиск

оптимума с инициализации вектора $\vec{u^0}$. На i шаге алгоритма алгоритма выполняется перемещение текущего вектора в минимум его окрестности:

$$F(\vec{u}^{i+1}) = min\{F(\vec{v}) : \vec{v} \in N_l(\vec{u}^i)\} \tag{18}$$

Алгоритм завершается в локальном оптимуме, когда $F(\vec{x^{i+1}}) = F(\vec{x^i})$. В задачах маршрутизации, как правило, имеется большое множество локальных оптимумов и только один из них носит глобальный характер:

$$F_{opt} = min\{F(\vec{v}) : v \in E^\eta\} \tag{19}$$

To ensure that the algorithm did not stop in a local minimum, and passed from one local minimum to another, with edges removed the Central point and when searching for the minimum of the following rule applies. Let $l = 2$ and in the transition from $\vec{u^{i+1}}$ to $\vec{u^i}$ change the values in the coordinate $(u^i_\lambda, u^i_\omega)$. The algorithm stores such pair for the last h couple of steps and in the next step prohibits the movement in these directions. An ordered list of such pairs

$$\phi^i = \{(u^i_\lambda, u^i_\omega), (u^{i-1}_\lambda, u^{i-1}_\omega), \cdots, (u^{i-h+1}_\lambda, u^{i-h+1}_\omega)\} \tag{20}$$

called the list of prohibitions. When building the list of all pairs of different and a pair of $(u_\lambda, u_\omega), \lambda \neq \omega$ does not prohibit the movement of pairs (u_λ, u_λ) and (u_ω, u_ω). When $l > 2$ similarly constructed three coordinates, fours, etc. A set of non-restricted vectors denote by $N_l(\vec{u^i}, \vec{\phi^i})$. In order that the search was effectively, it is advisable to use small values h and to control this parameter in the course of the algorithm.

```
1  function TabuSearch(u^0, l, p, h)
2  // Initialize variables:
3  u^opt = u^0  F^opt = F^0  φ^0 = ∅  i = 0
4  while the breakpoint is not triggered do
5  |  N_l = N_l(u^i, φ^i, p, h)
6  |  if N_l ≠ ∅ then
7  |  |  u^{i+1} = u^i
8  |  |  i = i + 1
9  |  |  goto 5
10 |  else
11 |  |  // find optimum into the neighborhood N_l:
12 |  |  u^{i+1} : F(u^{i+1}) = min{F(y) : y ∈ N_l}
13 |  end
14 |  if F(u^{i+1}) < F_opt) then
15 |  |  F_opt = F(u^{i+1})
16 |  |  u^opt = u^{i+1}
17 |  end
18 |  φ^{i+1} = update(φ^i)
19 |  i = i + 1
20 end
21 return u^opt
```

Algorithm 1: Pseudo-code for probabilistic Tabu Search algorithm.

Denote by $N_l(\vec{u^i}, \vec{\phi^i}, p)$ probabilistic edge that goes from a deterministic $N_l(\vec{u^i}, \vec{\phi^i})$ as follows. Each vector $\vec{v} \in N_l(\vec{u^i}, \vec{\phi^i})$ with probability p included in the neighborhood $N_l(\vec{u^i}, \vec{\phi^i}, p)$ regardless of other points. Note that this set may be empty or contain only one point. The General scheme of the probabilistic search algorithm with a list of prohibitions referred to as pseudo-code in algorithm scheme 1.

As the stopping criterion is based on the total number of steps N_stop, in the course of which does not change the value F_{opt}. Values l, p, h are the control parameters of the algorithm. Their choice depends on the problem dimension.

In the presented scheme, it was assumed that the value of h (the dimension of the deny list) does not change in the course of the algorithm. This creates certain difficulties in the implementation of the scheme, as it is unknown how long is to take a list of prohibitions. At small h the algorithm can start walking cycle. At large h the search becomes inefficient. One of the simple rules regulating the length of the list of prohibitions is the following.

If at the next step of the algorithm, the edge $N_l(\vec{u^i}, \vec{\phi^i}, p)$ turned out to be empty, it is added to the list of fictitious vector $\vec{0}$. This vector nothing forbids, but reduces the number of bans per unit.

Ініціалізація

Для того, щоб використовувати алгоритм імітації відпалу необхідно побудувати початкове рішення. Для отримання початкового рішення $\vec{u^0}$ можна скористатися евристичним методом конструювання маршруту. Суть цих методів полягає в наступному. Для кожного автомобіля послідовно будується маршрут шляхом додавання ще не розглянутих клієнтських заявок за заданими правилами. У нашому випадку, для збереження порядку відвідування вершин, під замовленням розуміється пара "відправник-одержувач".

Найвідоміший метод конструювання маршруту був запропонований Соломоном у 1987 році [**?**, **?**] і в літературі зустрічається як метод Соломона. Даний алгоритм є досить швидким (обчислювальна складність $O(n^3)$ [**?**]), тому його зручно використовувати для ініціалізації вектора $\vec{u^0}$.

На початку роботи алгоритму відбувається додавання по одному запиту на маршрут у відповідності з тимчасовими обмеженнями. Вибір першої заявки відбувається випадковим чином, або вибирається та, яку необхідно виконати раніше за інших. Далі кожна можлива заявка u з множини нерозглянутих заявок Q (множина $Q' \in Q$) оцінюється на предмет вставити її в початок маршруту, кінець або між двома сусідніми вузлами маршруту. Для вибору місця вставки використовується наступний критерій:

$$c(k, v_i, u, v_{i+1}) = \min = \begin{cases} \min_{q=2,\dots,n^k}(\Omega^k_{q-1,u} + \Omega^k_u + \Omega^k_{u,q} - \mu\Omega^k_{q-1,q}), \\ \Omega^k_{k,u} + \Omega^k_{u,q} + \Omega^k_u\mu\Omega^k_{q,q}), q = 1 \\ \Omega^k_{q,u} + \Omega^k_u, q = n^k \end{cases} \quad (21)$$

де:
k – маршрут транспортного засобу $k \in C$;
n^k - кількість вузлів у маршруті;
μ – довільний параметр, $\mu \geq 0$;
$\Omega^k_u = \Omega^k[\vec{P}_{s_q}, \vec{P}_{r_u}]$ – варіант вставки нової заявки між вже існуючими відправником і одержувачем;
$\Omega^k_{u,q} = \Omega^k[\vec{P}_{r_u}, \vec{P}_q]$ – варіант вставки нової заявки на початок маршруту;
$\Omega^k_{q,u} = \Omega^k[\vec{P}_q, \vec{P}_{s_u}]$ – варіант вставки нової заявки в кінець маршруту.

Додається та заявка, у якій $c(k, v_i, u, v_{i+1})$ буде мінімальною:

$$u^* : c(k, v_i, u^*, v_{i+1}) = \min_u[c(k, v_i, u, v_{i+1})] \quad (22)$$

В результаті u^* додається до поточного маршруту k на найвигіднішу позицію.

На рис. 2 представлений зважений граф, на прикладі якого показано виконання одного кроку методом Соломона:
a) вихідний граф, на якому показані варіанти додавання нових заявок на маршрут;
b) маршрут, який був побудований методом Соломона після виконання одного кроку.

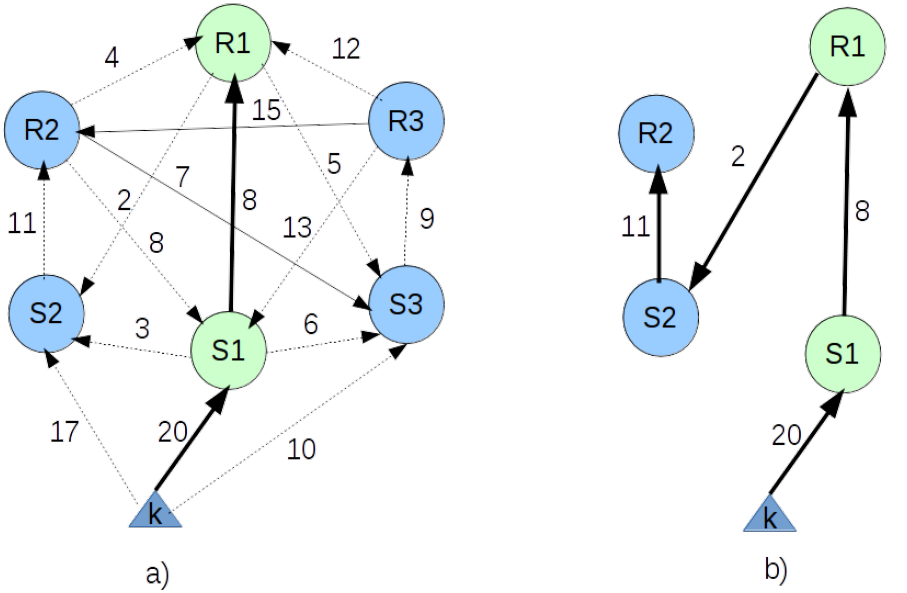


Рис. 2. Приклад виконання одного кроку методом Соломона

Приклад. Спочатку дана заявка (яка складається з відправника і одержувача). Значення функції вартості Ω представлені на дугах графа (рис.2). Необхідно додати нові заявки в маршрут, використовуючи критерії 21 і 22. Коефіцієнт $\mu = 0.5$.

Рішення. Розглянемо процес додавання нових заявок. З двох запропонованих заявок $(\{S_2, R_2\}, \{S_3, R_3\})$ виберемо ту, яку потрібно додати в першу чергу. В залежності від місця куди можна додати заявку (початок, кінець, середина) виберемо потрібний критерій. Таким чином, знайдемо оптимальне рішення вставки нової заявки $\{S_2, R_2\}$:
$\min[c(k, S_1, \{S_2, R_2\}, R_1), c(k, \{S_2, R_2\}, S_1, R_1), c(k, S_1, R_1, \{S_2, R_2\})] = \min[3 + 11 + 4 - 0.5 \cdot 8; 17 + 11 + 8 - 0.5 \cdot 20; 2 + 11] = \min[14; 26; 13]$
Знайдемо оптимальне рішення вставки нової заявки $\{S_3, R_3\}$:
$\min[c(k, S_1, \{S_3, R_3\}, R_1), c(k, \{S_3, R_3\}, S_1, R_1), c(k, S_1, R_1, \{S_3, R_3\})] = \min[6 + 9 + 12 - 0.5 \cdot 8; 10 + 9 + 13 - 0.5 \cdot 20; 5 + 9] = \min[23; 22; 14]$.
Із знайдених рішень виберемо рішення з найменшим значенням $c(k, v_i, u, v_{i+1})$. У нашому прикладі це $c(k, S_1, R_1, \{S_2, R_2\} = 13$.

Таким чином, новий маршрут буде складатися з початкових заявок і нової заявки $\{S_2, R_2\}$, яка буде додана в кінець маршруту.

Відповідь: в маршрут $\{S_1, R_1\}$ необхідно додати заявку $\{S_2, R_2\}$, яка буде розміщена в кінці.

На рис. 3 представлені варіанти додавання в маршрут нової заявки $\{S_2, R_2\}$:

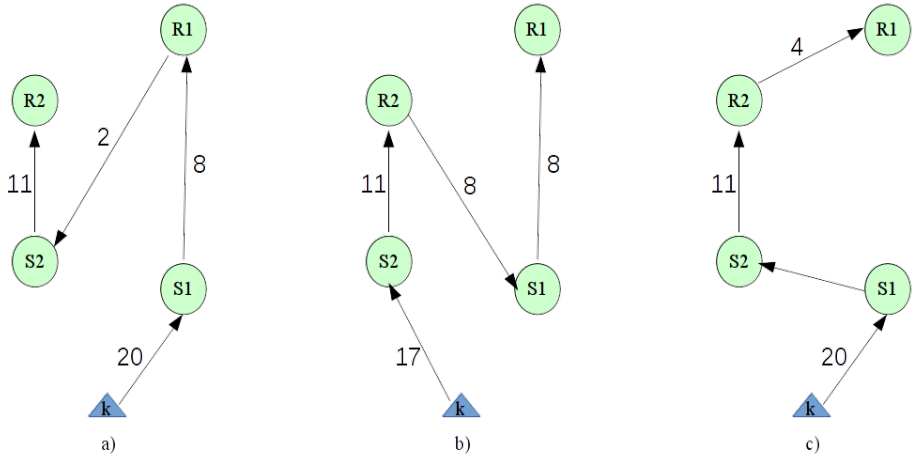


Рис. 3. Варіанти додавання в маршрут заявки $\{S_2, R_2\}$

a) варіант додавання заявки $\{S_2, R_2\}$ в кінець маршруту;
b) варіант додавання заявки $\{S_2, R_2\}$ в початок маршруту;
c) варіант додавання заявки $\{S_2, R_2\}$ між вершинами $\{S_1, R_1\}$.

На рис. 4 представлені варіанти додавання в маршрут нової заявки $\{S_3, R_3\}$:
a) варіант додавання заявки $\{S_3, R_3\}$ в кінець маршруту;
b) варіант додавання заявки $\{S_3, R_3\}$ в початок маршруту;
c) варіант додавання заявки $\{S_3, R_3\}$ між вершинами $\{S_1, R_1\}$.

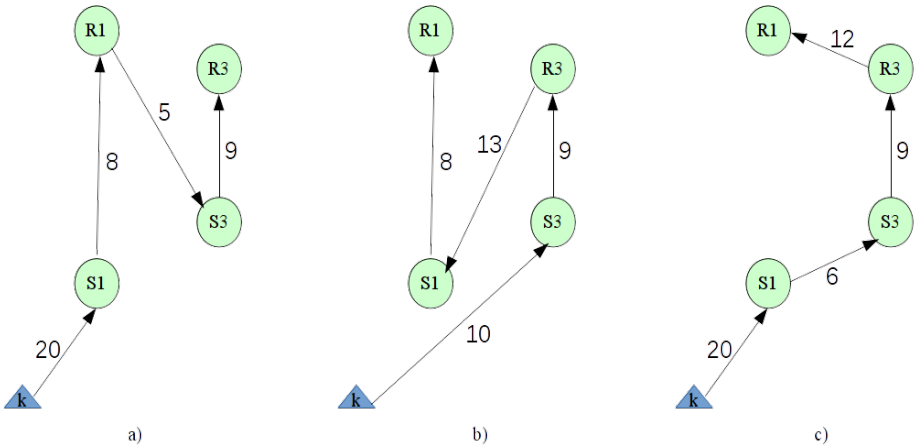


Рис. 4. Варіанти додавання в маршрут заявки $\{S_3, R_3\}$

Метод формування околиці

Під час роботи алгоритму імітації відпалу необхідно проводити перегляд околиці $N_l(\vec{u^i})$ поточної точки $\vec{u^i}$.В ході обчислення маршруту в околицю потрапляє досить велика кількість точок. Але в зв'язку з особливостями обмежень нашої моделі багато з цих точок не є рішенням задачі. Тому в зв'язку з особливостями розглянутої дискретної задачі ми використали наступний прийом визначення околиці.

Нехай $x^k(v)$ – індекс вузла v, $v \in R \cup S$ в маршруті автотранспортного засобу $k \in C$. Кожна заявка складається з пари $\{S, R\}$, де S – це відправник, а R – одержувач. Під час визначення околиці, повинна зберігатися цілісність маршруту, тобто виконуватися обмеження (7 –10). Над заявками маршруту $Route_i$, $i = 1 \dots n$ виконуються операції перестановки. Отримані під час перестановок вектори $\vec{u}$ і буде околицею $N_l(\vec{u^i})$. Для перестановки використовувалися операції переміщення і поглинання.

Операція поглинання представлена на рис.5(a), на якому представлено два маршрути $Route_1$ і $Route_2$. Маршрут $Route_1$ складається з вузлів $\{C_1, S_1, R_1, S_2, R_2, S_3, R_3\}$, а маршрут $Route_2$ складається з вузлів $\{C_2, S_4, R_4, S_5, R_5, S_6, R_6\}$. Під час виконання операції поглинання заявки $\{S_1, R_1\}$ в маршруті $Route_1$, вставка даної заявки в маршрут $Route_2$ може бути виконана на $n + 1$ позицій.

Операція переміщення представлена на рис.5(b) На даному рисунку представлений маршрут $Route_1$, який складається з вузлів $\{C_1, S_1, R_1, S_2, R_2, S_3, R_3\}$. Для збереження цілісності маршруту вузол S_i (відправник) має бути відвіданий до R_i (одержувач), виходячи з цього під час операції переміщення, вузол S_i може переміщатися поки буде виконуватися умова $0 < x^{new}(S_i) < \widetilde{u}(R_i)$, а вузол R_i може переміщатися поки буде виконуватися умова $n^k \geq x^{new}(R_i) > \widetilde{u}(S_i)$.

Analysis of the algorithm parameters

The Tabu Search algorithm has two main parameters: h – size of the list of prohibitions and $N_{neighbors}$ – accepted count limit of size of neighbordhood $N_l(\vec{u})$. For the experiments, we constructed the test tasks on the basis of the capacity routing problems developed by Breedam, Fisher, Christofides and Eilon. To determine trends in time and cost of the decision we used exponential smoothing by a factor of 0.1.

In this experiment we use two different breakpoint strategies:

1. *StepCount* termination strategy – terminates when an amount of steps has been reached;
2. *TimeSpent* termination strategy – terminates when an amount of time has been reached.

As should have been expected, for large values of the parameter a jam occurs in local optimum due to the large number of restrictions of movement in space.

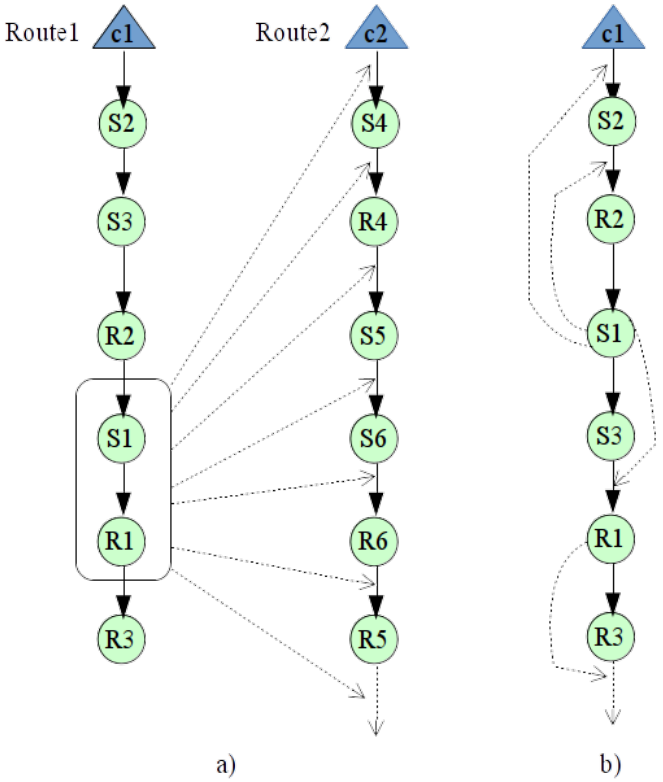


Рис. 5. Приклад операції a)поглинання та b)переміщення

Otherwise if you select a too small tabu size, algorithm can still get stuck in a local optimum. In the first computational experiment we made finding the best solutions for different values of the tabu size parameter from interval $h \in [0, 40]$ (pict. 6 an pict. 7).

In the second experiment (see figure 8) we built a relationship between the number $N_{neigh.}$ of viewed solutions neighborhood size $N_l(\vec{u})$ using $TimeSpent$ termination strategy. In this experiment we made finding the best solutions for different values of the neighborhood size parameter from interval $N_{neigh.} \in [750, 850]$.

Modification

The first generalized diagram is as follows. First, the set of solutions is constructed by Solomon. Then every solution is improved by tabu search and choose the best solution according to the objective function.

Modification of the scheme is based on the hypothesis of "On a large valley"[9]. According to this hypothesis, the average local optima are located much closer to

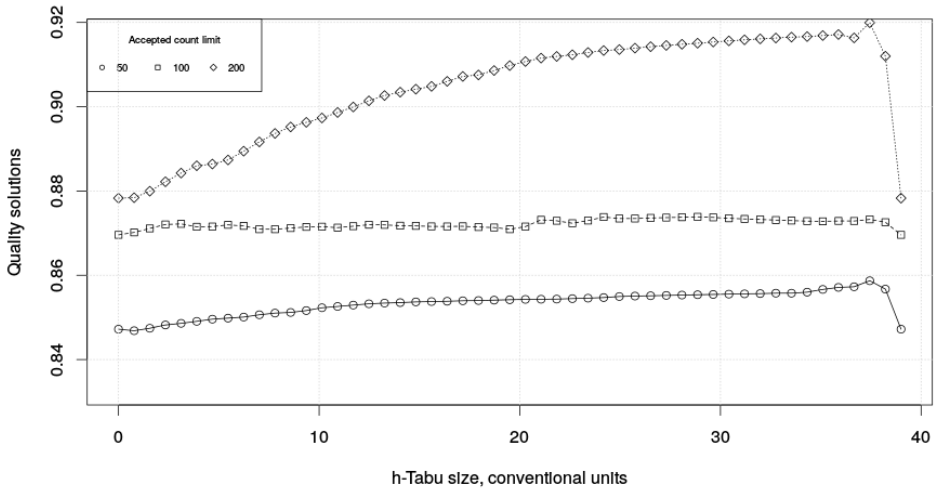


Рис. 6. The dependence of the quality of solutions on the size of the tabu list when using *StepCount* termination strategy

the global than a randomly chosen point. There is a certain concentration of local optimum in a small part of the feasible region, which is figuratively called a large valley. If this assumption is true, then it is advisable to remember the best solutions and based on them design new original decision. We use this idea to solve the Routing Courier Delivery Problem.

We proceed to the description of the algorithm in scheme 2. Let U^{opt} is a sorted array of optimal solutions by value of the objective function ascending, ie:

$$F(u_1^{opt}) \leq F(u_2^{opt}) \leq \ldots < F(u_i^{opt}) \leq \ldots \leq F(u_{sizeof(U^{opt})}^{opt}) \tag{23}$$

Each solution u_i^{opt} gets into the population with a given probability p_u^i, the probability of selection decreases with increasing sequence number:

$$p_u^1 = 1 > p_u^2 > \ldots > p_u^i > \ldots > p_u^{sizeof(U^{opt})}, p_u^{sizeof(U^{opt})} > 0 \tag{24}$$

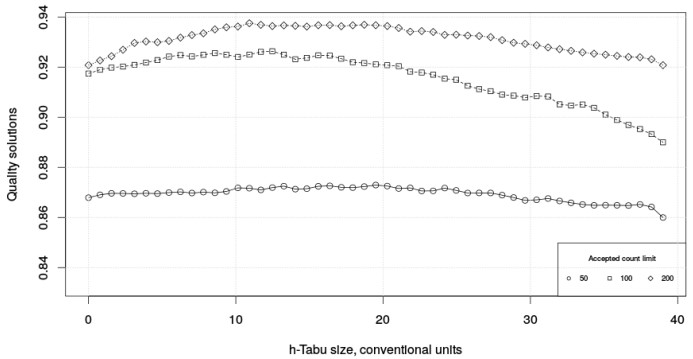


Рис. 7. The dependence of the quality of solutions on the size of the tabu list when using *TimeSpent* termination strategy

```
1  function constructSolution( U^opt, p_u, n_s^min)
2  // U^opt – a sorted set array of optimal solutions
3  if sizeof(U^opt) < n_s^min then
4  |  // construct solution using heuristics(for example, Solomon alg.)
5  |  u^0 = heuristicsConscruction()
6  |  return u^0
7  end
8  while i < sizeof(U^opt) do
9  |  if random(0.0, 1.0) > p_u^i then
10 |  |  i = i + 1
11 |  |  goto 8
12 |  end
13 |  R_i =randomly select a route from u_i^opt solution; u^0 = u^0 ∪ R_i i = i + 1
14 end
15 if u^0 isnotcomplete then
16 |  u^0 = heuristicsConscruction(u^0)
17 end
18 return u^0
```

Algorithm 2: Pseudo-code for heuristics construction algorithm.

It then crosses solutions by the following rule: with the first solution is randomly selected route R_1. Then another solution is selected from a route R_2 that does not include client requests from the first route etc. If the client requests were not included in the solution u^0, then these requests are added by the construction heuristics algorithm (for example, Solomon algorithm).

Now we describe the primary function $SolveCDP()$ of the modified algorithm presented in pseudocode form in Scheme 3. In this function using a loop is a

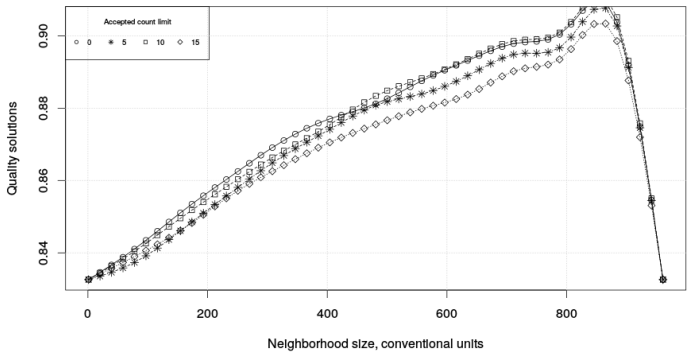


Рис. 8. The dependence of the quality of solutions on the neighborhood size when using *TimeSpent* termination strategy

sequential formation of an array of the best solutions U^{opt} by means TabuSearch algorithm.

Based on the results of the experiments described in the preceding section, we decided to set the parameters of the algorithm are not fixed values, but as a random variable of predetermined period. For example, the size of the tabu list h is defined as the interval $[10, 35]$(see pict. 6 and pict. 7) and the neighbordhood size $l = N_{neighbordhood}$ is defined as the interval $[600 \cdot n, 850 \cdot n]$, where n – problem size (see pict. 8).

Let $\vec{\psi}_i = (\psi_i^1, \psi_i^2, \psi_i^3) = (\tilde{l}_i, \tilde{p}_i, \tilde{l}_h)$ – a set of values of the TabuSearch configuration parameters which was using for solving the problem in the step i.

Let $[\psi^j_{begin}, \psi^j_{end}]$ – an optimal interval of the TabuSearch j -parameter. (For example: if $j = 3$, then: $[\psi^3_{begin}, \psi^3_{end}] = [h_{begin}, h_{end}] = [10, 40]$). These intervals are initial data and they are set in the initialization block.

The configuration parameters ψ_i^j are randomly selected from the interval $[\psi^j_{begin}, \psi^j_{end}]$ according to a distribution law. We proposed to build the distribution density $f_j(\psi_i^j)$ of the random variable ψ_i^j as follows. At first, we define anchor points:

$$\rho_i^j = \frac{F(u_1^{opt})}{avg[F(u_k^{opt})]}, \forall k : \psi_k^j = \psi_i^j \tag{25}$$

The number of anchor points n_ρ is smaller than the set of optimal solutions, because a set of anchor points are removed the same points ($n_\rho \leq n_u$).

The main idea is that the distribution density should be larger at those points (parameters ψ_i^j) in which high quality solutions than the points at which the low quality of solutions. Therefore, we presented the distribution function $f_j(z)$ as a sum

of kernel functions:

$$f_j(z) = \alpha_j * \sum_{i=1}^{n_\rho} \rho_i^j * K(z - \psi_i^j) \tag{26}$$

where: $K(x)$ – kernel function, α – a normalizing parameter

We have chosen as the kernel the function of parabolic type(known as an Epanechnikov function), because during the experiments the best results were obtained using this function:

$$K(z) = 3/4 \cdot (1 - z^2) \tag{27}$$

```
1  function SolveCDP( n_s^min)
2  // Initialize variables:
3  ψ = ∅, p_u = ∅, U^opt = ∅, i = 0
4  while the breakpoint is not triggered do
5      // set TabuSearch parameters
6      for j = 1 ... 3 do
7          // f_j – the density distribution of the random
8          // variable ψ^j (TabuSearch parameter l, p or h)
9          ψ_i^j = random(ψ_begin^j, ψ_end^j, f_j)
10     end
11     // Make solution using euristics construction algorithms
12     u^0 = constructSolution(U^opt, n_s^min)
13     // Improve solution u^0 using TabuSearch algorithm
       u_i^opt = TabuSearch(u^0, ψ_i⃗)
14     U^opt = U^opt ∪ {u_i^opt}
15     // Sort ascending optimums U of the objective function
16     U^opt = sort(U^opt)
17     p_u^{i+1} = update(p_u^i)
18     i = i + 1
19 end
20 return u_1^opt
```

Algorithm 3: Pseudo-code for modified tabu-search algorithm.

The parameter α_j was introduced in the density function to perform the normalization condition:

$$\int_{\psi_{begin}^j}^{\psi_{end}^j} f_j(z)dz = 1 => (\alpha_j)^{-1} = \sum_{i=1}^{n_\rho} \rho_i^j \cdot \int_{\psi_{begin}^j}^{\psi_{end}^j} K(z - \psi_i^j)dz \tag{28}$$

The parameter α_j can be written in explicit form:

$$(\alpha_j)^{-1} = \frac{3}{4} \cdot \sum_{i=1}^{n_\rho} \rho_i^j \cdot [(1 - (\psi_i^j)^2) \cdot (\psi_{end}^j - \psi_{begin}^j) + \\ + \psi_i^j \cdot ((\psi_{end}^j)^2 - (\psi_{begin}^j)^2) - 1/3 \cdot (\psi_{end}^j)^3 - (\psi_{begin}^j)^3)] \tag{29}$$

In figure 9 the results of an experiment in which were presented a dependence of the quality of the solutions obtained by the classical and modified algorithms and the breaking point time. As seen from the graph, with an increase in operating time of the computational scheme, the quality of solutions obtained using the modified algorithm is growing faster than using classical Tabu Search algorithm.

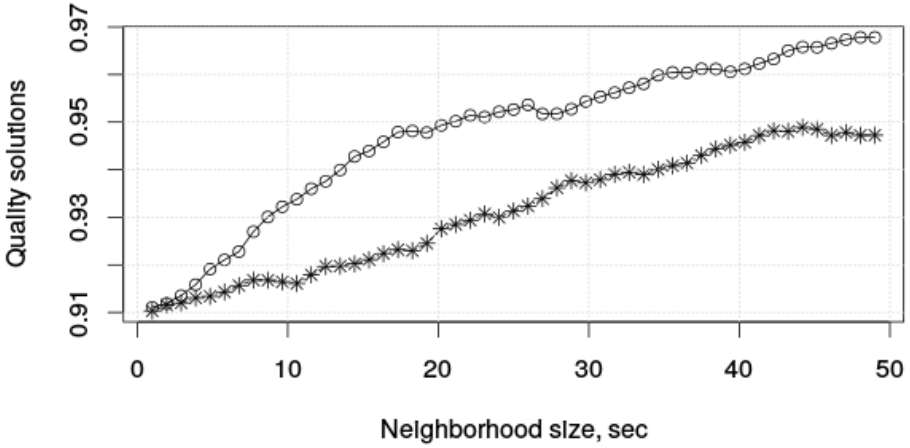


Рис. 9. The dependence of the quality of solutions and the breaking point time by classic and modified algorithms using *TimeSpent* termination strategy

Conclusion

During the research has been investigated and implemented tabu search algorithm to solve the Static Routing Courier Delivery problem with time constraints. Also a modified algorithm was developed based on the Tabu Search algorithm to improve the quality of the solutions. Modified algorithm gave the best solutions in terms of the balance between the number of vehicles and the cost traveled. In practice this algorithm can be used for decision support in intelligent systems for improving the quality of customer service and for reducing waiting the time to use vehicle, this will reduce fuel costs and depreciation of transport.

The analysis of the parameters of implemented algorithms allowed us to determine their optimal values for this class of routing problems. With the modified algorithm was found solutions of model problems, which in most cases have an acceptable deviation from the global optimum.

Список литературы

[1] F. Ordonez, Chen Wang, A New Approach for Routing Courier Delivery Services //METRANS Transportation Center:University of Southern California Los Angeles, 2012, 81–115.

[2] R. Masson, F. Lehued, O. Peton,The dial-a-ride problem with transfers. // Computers and Operations Research, Vol. 41, 2014, p. 12–23.

[3] T. Babb, Pickup and Delivery Problem with Time Windows // Coordinated Transportation Systems: The State of the Art. Department of Computer Science University of Central Florida Orlando, Florida, 2005, 38 p.

[4] R. Shafeyev ,L. Lyubchik, A some realization of Tabu Search algorithm for Solving the Transportation Problem with Time Constraints // Vestnik NTU "KhPI". – Kharkov: NTU "KhPI 2013. – No3 (977). – p. 35–39.

[5] R. Shafeyev. Java-based optimixation framework for solving routing problems. url: http://jlogistics.net, 2015.

[6] W Barnes, Solving the pickup and delivery problem with time windows using reactive tabu search // Transportation Research Part B: Methodological, Vol. 34 Issue 2, 2000, p. 107–121.

[7] B. Coltin, M. Veloso, Scheduling for Transfers in Pickup and Delivery Problems with Very Large Neighborhood Search // The Twenty-Eighth Conference on Artificial Intelligence, Quebec City, Canada, 2014, 7 p.

[8] E. Goncharov, Y Kochetov, Probabilistic search with exclusions for discrete unconstrained optimization //Discrete Analysis and Operations Research, Moscow, Serial 2. Vol. 9, 2002, p. 13-30.

[9] Y Kochetov, Probabilistic methods of local search for discrete optimization problems //Discrete Mathematics and Its Applications: Proceedings of youth lectures and scientific schools in discrete mathematics and its applications, Publishing House of the Center for Applied Research at the Mechanics and Mathematics. Faculty of Moscow State University, 2001, p. 84-117.

[10] O. Braysy, M. Gendreau, Vehicle Routing Problem with Time Windows, Part I: Route Constuction and local algorithms // Transportation science Vol.39 No. 1, 2005, p. 104-118.

[11] V. Goldberg, R. Kennedy, An Efficient cost scaling algotirhm for the assignment problem, Math. Program., 1995, p. 153–177.

[12] N. Christofides, S. Eilon, An algorithm for the vehicle dispatching problem //Operational Research Quarterly, 20, 1969, p. 309–318.

[13] B. Golden, E. Wasil, J. Kelly, I-M. Chao. The impact of metaheuristics on solving the vehicle routing problem: Algorithms, problem sets, and

computational results. In T. Crainic and G. Laporte, editors // Fleet Management and Logistics, Kluwer, Boston, 1998 p. 33–56.
[14] E. Taillard. VRP benchmarks.
url: http://mistic.heig-vd.ch/taillard/problemes.dir/vrp.dir/vrp.html, 1993.
[15] A. Attanasio, J. Bregman,G. Ghiani, E. Manni. Real-time fleet management at Ecourier Ltd. Dynamic Fleet Management, volume 38 of Operations Research/Computer Science Interfaces, chapter 10, p.219–238, 2007.
[16] A. Beaudry, G. Laporte, T. Melo, Nickel. Dynamic transportation of patients in hospitals. Berichte des Fraunhofer ITWM, Nr. 104 :p.1–34, 2010.
[17] M.Romero, L.Sheremetov and A.Soriano. A genetic algorithm for the pickup and delivery problem: An application to the helicopter offshore transportation. In Theoretical Advances and Applications of Fuzzy Logic and Soft Computing, volume 42 of Advances in Soft Computing, 2007, p. 35–44.
[18] M.Romero, L.Sheremetov and A.Soriano. Yannick Kergosien, Christophe Lent, D. Piton, Jean-Charles Billaut. A tabu search heuristic for a dynamic transportation problem of patients between care units. 29 pages. 2010.
[19] H. Sarak, A. Satman. The degree-day method to estimate the residential heating natural gas consumption in Turkey: a case study. Pergamon, Energy 28 (2003) 929–939.