ISSN 0005-1179 (print), ISSN 1608-3032 (online), Automation and Remote Control, 2023, Vol. 84, No. 3, pp. 341–364.
Trapeznikov Institute of Control Sciences, Russian Academy of Sciences, 2023.

Greedy and Adaptive Algorithms for Multi-Depot Vehicle Routing with Object Alternation

S. N. Medvedev

Voronezh State University, Voronezh, Russia
e-mail: s_n_medvedev@mail.ru

Received July 22, 2022
Revised September 28, 2022
Accepted October 26, 2022

Abstract—This paper considers the multi-depot vehicle routing problem with object alternation. We propose a formal statement of the problem with two types of objects and a mathematical model with two blocks of Boolean variables. First, the model is studied without gathering vehicles (mobile objects). Then, a special object (a single collection point) is introduced in the model. We show additional constraints of the mathematical model with this object. Special attention is paid to the condition of no subcycles. This condition is considered based on the adjacency matrix. Five greedy algorithms are proposed for solving the problem, two of which are iterative. One of the greedy algorithms is given a probabilistic modification based on the randomization of variables (an adaptive algorithm). Finally, the proposed algorithms are compared in terms of the average value of the objective function and the running time in a computational experiment. Also, the results of another experiment—the parametric tuning of the adaptive algorithm—are presented.

Keywords: vehicle routing problem, adaptive algorithm, greedy algorithm

DOI: 10.25728/arcRAS.2023.81.72.001

1. INTRODUCTION

There are two global lines of research on vehicle routing problems (VRP): modeling various modifications and supplements of the classical problem and developing efficient solution algorithms.

Possible VRP statements were surveyed in [1]. They include the graph statement, the statement based on mathematical integer programming, two- and three-index statements of the problem, and the statement based on scheduling theory. Analyzing the latter statement, the author came to the following conclusion: it is possible to apply heuristics related to the traveling salesman problem to determine an upper estimate for the number of vehicles required. In the paper [2], the territory design model of VRP was investigated by introducing additional constraints. This model divides the customers into zones to find the best routes in each zone. The study [3] was devoted to the continuous VRP model. Special attention was paid to the analysis of the speed of mobile objects. The paper [4] presented a mathematical model with the constraints of many well-known VRP statements. An algorithm based on ant colony optimization with an evolutionary strategy was developed therein.

The researchers [5] proposed an ant colony optimization algorithm for VRP with an adaptive gradient descent-based learning mechanism. This approach integrates the classical method (calculating and putting the pheromone on the route of the best ant in the population) with adaptation (updating the pheromone matrix adaptively using gradients). According to the results of the calculations, the proposed algorithm yields the best solution compared to other algorithms on all the

test data considered. In [6], a genetic algorithm was developed for solving a multi-depot VRP with time windows. Note that it includes a special population initialization technique. Another approach provided in [7] combines genetic algorithms with ant colony optimization.

In [8, 9], the authors described various mathematical models for VRP with object alternation and a single collection point, including heuristic algorithms for solving them based on ant colony optimization and genetic algorithms. The paper [10] proposed an exact branch-and-bound algorithm for solving this problem.

Many VRP studies have an application-oriented nature since the scope of such problems is extremely wide. VRP variations can be used, e.g., when grouping geographical objects into different clusters [2] for rational planning of sales and transportation processes, maintenance, garbage collection, medical care, etc. Other applications include optimal routing for manned and unmanned vehicles in urban and intercity transport networks, harvesting in fields, and extinguishing fires by air and ground means [1, 5, 8]. Also, such problems arise when designing calculations, when certain processes need to be assigned to several devices [11]. Thus, research focused on the models and algorithms of various VRP modifications is topical both theoretically and practically.

In this study, we propose a discrete optimization model for the VRP with object alternation and a single collection point under the condition of no subcycles. (Subcycles are also called subtours in the literature.) The emphasis below is on constructing a mathematical model with two blocks of two-index variables and deriving an alternative condition of no subcycles. Also, the problem statement does not involve a fixed number of mobile objects (vehicles). We develop iterative and noniterative greedy algorithms for solving the problem and an adaptive algorithm representing a probabilistic modification of one of the greedy algorithms. With the two blocks of two-index variables in the mathematical model, we select a block of the adaptive algorithm that does not require updating the probabilities at each step.

2. PROBLEM STATEMENT

Let us recall the multi-depot VRP statement with object alternation [10].

Consider a set of fixed objects (vertices) of two types: targets (type A) and depots (type B). The cost of moving between objects is known. Several mobile objects (vehicles) in the aggregate must visit all the fixed objects of type A with the minimum total cost. Moreover, the targets and depots must alternate in the routes of vehicles. In addition, each target may be visited once in the aggregate, whereas any depot may be visited an unlimited number of times. In this case, all mobile objects start and end their route at some fixed collection point.

The cost can be understood as distance, time, price, etc.

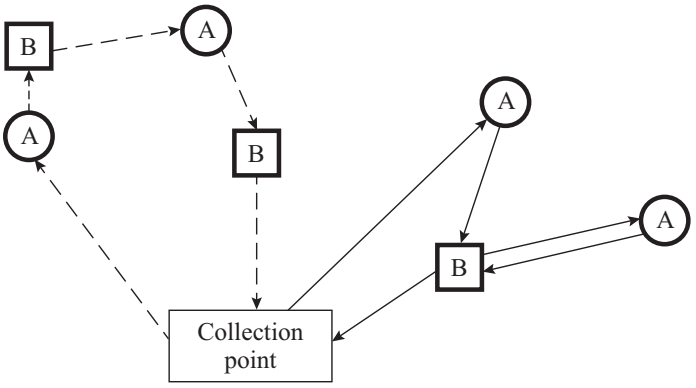

Fig. 1. Multi-depot VRP with object alternation and a single collection point.

Note that the number of mobile agents in this problem statement is not specified: several or only one.

Figure 1 shows a graphical interpretation of this problem: the targets are indicated by circles, depots by squares, and the collection point by a rectangle. The arcs (dashed and solid) correspond to some route.

The route in Fig. 1 has two visits to the collection point. According to the classical VRP interpretation, there are two mobile objects, one moving along dotted arcs and the other along solid ones. Another interpretation is when only one mobile object visits the collection point twice. This paper focuses on the number of visits to a single collection point (by one or more vehicles) rather than on the number of mobile objects.

Note that the need to consider the exact number of available mobile objects will lead to a different mathematical model and other solution algorithms.

3. THE MATHEMATICAL MODEL

First, we construct a mathematical model of the problem without requiring that all mobile objects start and end their route at some fixed collection point (Fig. 2).

Let us introduce the following notations:

n is the number of targets (the fixed objects of type A);

m is the number of depots (the fixed objects of type B);

$(c^1_{ij})_{m\times n}$ is a matrix specifying the cost of moving between the ith depot (type B) and the jth target (type A);

$(c^2_{ji})_{n\times m}$ is a matrix specifying the cost of moving between the jth target (type A) and the ith depot (type B).

We define two blocks of variables:

$$x_{ij}, y_{ji} = \{0, 1\}, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n.$$

Here, $x_{ij} = 1$ if the mobile object from the ith depot arrives at the jth target and $x_{ij} = 0$ otherwise.

By analogy, $y_{ji} = 1$ if the mobile object from the jth target arrives at the ith depot and $y_{ji} = 0$ otherwise.

The objective function is to minimize the total cost:

$$L(X, Y) = \sum_{i=1}^{m}\sum_{j=1}^{n}(c^1_{ij}x_{ij} + c^2_{ij}y_{ij}) \to \min, \tag{1}$$

where $X = (x_{ij})_{m\times n}$ and $Y = (y_{ji})_{n\times m}$ are the matrices of the corresponding variables.

We proceed to the problem constraints, which form an admissible set of solutions.

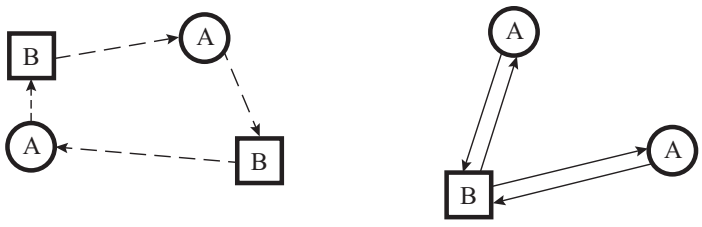

Fig. 2. Multi-depot VRP with object alternation and without any collection point.

By the problem statement, only one mobile object must arrive at each target and only one mobile object must depart from each target. This requirement can be written as the nonlinear constraint

$$\left(\sum_{i=1}^{m} x_{ij}\right)\left(\sum_{i=1}^{m} y_{ji}\right) = 1, \quad j = 1, \ldots, n.$$

It can be replaced by the equivalent constraints

$$\sum_{i=1}^{m} x_{ij} = 1, \quad j = 1, \ldots, n, \tag{2}$$

$$\sum_{i=1}^{m} y_{ij} = 1, \quad j = 1, \ldots, n, \tag{3}$$

belonging to the class of constraints of the classical traveling salesman problem. As a result, the mathematical model of the problem becomes linear.

The following conditions must be considered to make the route of each mobile object inseparable: if a mobile object arrives at a target, it must also leave this target; if a mobile object departs from a target, it must also have arrived at this target. Such conditions can be expressed as the nonlinear constraints

$$\left(\sum_{i=1}^{m} x_{ij}\right)\left(1 - \sum_{i=1}^{m} y_{ji}\right) = 0, \quad j = 1, \ldots, n,$$

$$\left(\sum_{i=1}^{m} y_{ji}\right)\left(1 - \sum_{i=1}^{m} x_{ij}\right) = 0, \quad j = 1, \ldots, n.$$

They are equivalently transformed to the linear constraint

$$\sum_{i=1}^{m} x_{ij} = \sum_{i=1}^{m} y_{ji}, \quad j = 1, \ldots, n. \tag{4}$$

This condition is redundant due to (2) and (3).

Similarly, the following conditions must be considered to ensure the inseparability of a route: if a mobile object arrives at a depot, it must also leave from this depot; if a mobile object departs from a depot, it must also have arrived at this depot. Such conditions are written as follows:

$$\left(\sum_{j=1}^{n} y_{ji}\right)\left(\sum_{j=1}^{n} y_{ji} - \sum_{j=1}^{n} x_{ij}\right) = 0, \quad i = 1, \ldots, m,$$

$$\left(\sum_{j=1}^{n} x_{ij}\right)\left(\sum_{j=1}^{n} x_{ij} - \sum_{j=1}^{n} y_{ji}\right) = 0, \quad i = 1, \ldots, m.$$

These nonlinear constraints are equivalently transformed to the linear constraint

$$\sum_{j=1}^{n} x_{ij} = \sum_{j=1}^{n} y_{ji}, \quad i = 1, \ldots, m. \tag{5}$$

It means that the number of arrivals at a depot coincides with the number of departures from this depot.

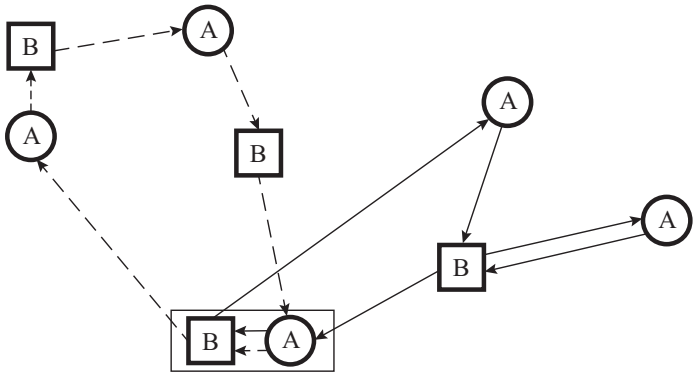

Fig. 3. Multi-depot VRP with object alternation and a single collection point defined by a "target–depot" pair.

The constraints (2), (3), and (5) on the variables x_{ij}, $y_{ji} = \{0, 1\}$, $i = 1, \ldots, m$, $j = 1, \ldots, n$, form the admissible set of solutions Ω.

Thus, the mathematical model (1)–(3), (5) describes the multi-depot VRP with object alternation.

Now we consider the case where all mobile objects start and end their route at some fixed collection point.

For example, the collection point can be a dummy target (an object of type A), a dummy depot (an object of type B), a pair of dummy objects with the "target–depot" passage direction (A–B), and a pair of dummy objects with the "depot–target" passage direction (B–A). (For pairs of dummy objects, the distance between them is 0.)

Each of these cases can be dictated by a particular application. For example, it is convenient to use the "target–depot" pair (Fig. 3) for vehicles harvesting stacks from a field, the "depot–target" pair for dump trucks working in a sand pit, a dummy object of type B for airplanes extinguishing landscape fires, and a dummy object of type A for couriers delivering documents.

This study will use the "target–depot" pair. (As has been already mentioned, a possible application is harvesting stacks from a field.) Let targets be stacks and depots be the places or machines into which stacks are loaded. Obviously, from the collection point the mobile objects (tractors) must move to the stacks, not to the machines. Similarly, after the last stack is loaded onto the machine, the tractor must move from it to the collection point. For preserving the alternation, the route must be as follows: "dummy stack–dummy machine–stack–machine–. . . –stack–machine–dummy stack."

Thus, the problem under consideration has a collection point given by the "target–depot" pair (Fig. 3).

We add a dummy target and a dummy depot, assigning the indices $j = 0$ and $i = 0$ to them, respectively. The following conditions are imposed on this pair:

1) A mobile object from the dummy target may arrive at the dummy depot only.

2) A mobile object may not arrive at the dummy target from the dummy depot.

3) A mobile object may not arrive at the dummy depot from a common target.

4) A mobile object may not arrive at a common depot from a dummy target.

It is necessary to introduce the new variables x_{0j}, x_{i0}, y_{j0}, and y_{0i}, $j = 0, \ldots, n$, $i = 0, \ldots, m$. Here, $x_{0j} \in \{0, 1\}$, $j = 0, \ldots, n$, and $x_{0j} = 1$ if the mobile object from the dummy depot arrives at the jth target and $x_{0j} = 0$ otherwise. The variables $x_{i0} \in \mathbb{N} \cup \{0\}$, $i = 0, \ldots, m$, show the number of visits to the dummy target from the ith depot. Note that $x_{00} = 0$ by condition 2. The variables $y_{j0} = 0$ and $y_{0i} = 0$, $j = 1, \ldots, n$, $i = 1, \ldots, m$, by conditions 3 and 4, and $y_{00} \in \mathbb{N}$.

The integer variables x_{i0} and y_{00} are used instead of the Boolean ones due to the constraint (5) extended for $i = 0$ and $j = 0$. Writing this constraint for $i = 0$, we obtain

$$\sum_{j=0}^{n} x_{0j} = \sum_{j=0}^{n} y_{j0}.$$

If y_{00} is a Boolean variable ($y_{00} = 1$ since the route must be closed), condition 3 leads to

$$\sum_{j=0}^{n} x_{0j} = \sum_{j=0}^{n} y_{j0} = y_{00} = 1,$$

i.e.,

$$\sum_{j=0}^{n} x_{0j} = 1.$$

Thus, only one mobile object may depart from the dummy depot. This feature does not reflect the essence of the collection point of mobile objects; in general, it is not true. Therefore, we introduce a positive integer variable y_{00}.

Similar reasoning yields $x_{i0} \in \mathbb{N} \cup \{0\}$, $i = 0, \ldots, m$. Here, the zero element is added because a mobile object will not necessarily arrive at the dummy target from each depot.

Consider the additional constraints imposed on dummy objects.

Note that the number of departures from the collection point and the number of arrivals at the collection point are unknown from the problem statement. Hence, it is impossible to write the analogs of the constraints (2) and (3) for $j = 0$. However, we can write an analog of the constraint (4):

$$\sum_{i=0}^{m} x_{i0} = \sum_{i=0}^{m} y_{0i}.$$

It has the following interpretation: the number of arrivals at the dummy target equals the number of departures from it.

The extension of the constraint (5) has been described above:

$$\sum_{j=0}^{n} x_{0j} = \sum_{j=0}^{n} y_{j0}.$$

In other words, the number of arrivals at the dummy depot equals the number of departures from it.

According to the aforesaid, the variables $y_{j0} = 0$, $y_{0i} = 0$, $j = 1, \ldots, n$, $i = 1, \ldots, m$. Therefore, the two latter equalities can be represented as

$$\sum_{i=0}^{m} x_{i0} = y_{00},$$
$$\sum_{j=0}^{n} x_{0j} = y_{00}.$$

As a result, we write another constraint:

$$\sum_{i=0}^{m} x_{i0} = \sum_{j=0}^{n} x_{0j}.$$

Obviously, it is redundant but reflects an important property of the collection point: the number of arrivals at this point coincides with the number of departures from it.

Well, the resulting mathematical model has the following form:

$$L(X,Y)=\sum_{i=0}^{m}\sum_{j=0}^{n}(c_{ij}^{1}x_{ij}+c_{ij}^{2}y_{ij})\to\min, \tag{6}$$

$$\sum_{i=0}^{m}x_{ij}=1,\quad j=1,\dots,n, \tag{7}$$

$$\sum_{i=0}^{m}x_{i0}=y_{00}, \tag{8}$$

$$\sum_{i=0}^{m}y_{ji}=1,\quad j=1,\dots,n, \tag{9}$$

$$\sum_{j=0}^{n}x_{0j}=y_{00}, \tag{10}$$

$$\sum_{j=0}^{n}x_{ij}=\sum_{j=0}^{n}y_{ji},\quad i=0,\dots,m, \tag{11}$$

$$x_{00}=0, \tag{12}$$

$$y_{j0}=0,\quad j=1,\dots,n, \tag{13}$$

$$y_{0i}=0,\quad i=1,\dots,m, \tag{14}$$

$$x_{ij},y_{ij}=\{0,1\},\quad i=1,\dots,m,\ j=1,\dots,n, \tag{15}$$

$$x_{0j}=\{0,1\},\quad j=0,\dots,n, \tag{16}$$

$$x_{i0}\in\mathbb{N}\cup\{0\},\quad i=0,\dots,m, \tag{17}$$

$$y_{00}\in\mathbb{N}. \tag{18}$$

For brevity, let K denote the sum $\sum_{j=0}^{n}x_{0j}$ (the number of visits to the collection point, either arrivals or departures).

Note that within this model, the route of any mobile object may contain subcycles. Therefore, we consider the condition of no subcycles.

4. THE CONDITION OF NO SUBCYCLES

Within the current model, for some mobile object, there may exist an inseparable route not passing through the dummy pair (the collection point). The case in Fig. 4 is possible and does not contradict the mathematical model (6)–(18).

Such a solution contains subcycles; see the dotted line in the figure. Thus, it is necessary to introduce an additional condition of no subcycles into the mathematical model. It can be formulated in terms of the adjacency matrix.

Definition 1. Two depots l and h are said to be *adjacent* if there exists a target j such that $x_{lj}=1$ and $y_{jh}=1$.

We say that two such depots are connected by a route of length 1 provided that $l\neq h$.

If a depot is included in a route, it is supposed adjacent to itself.

Consider the process of constructing the adjacency matrix $A=(a_{lh})_{(m+1)\times(m+1)}$, $l,h=0,\dots,m$. The diagonal elements a_{ll} are 1 if there exists an index j such that $x_{lj}=1$, i.e., $\sum_{j=0}^{n}x_{lj}\geqslant 1$, and

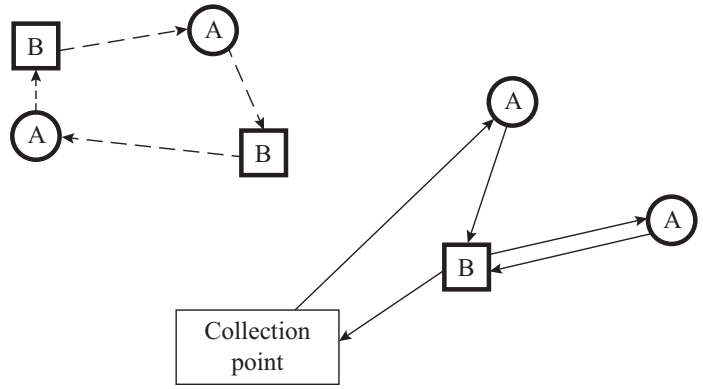

Fig. 4. An inseparable route not passing through the collection point.

are 0 otherwise. The other elements a_{lh}, $l \neq h$, are 1 if there exists an index j such that $x_{lj} = 1$ and $y_{jh} = 1$, i.e., $\sum_{j=0}^{n} x_{lj} y_{jh} \geqslant 1$, and are 0 otherwise.

Using the Iverson bracket, these conditions can be written as

$$a_{ll} = \left[\sum_{j=0}^{n} x_{lj} \geqslant 1\right], \quad a_{lh} = \left[\sum_{j=0}^{n} x_{lj} y_{jh} \geqslant 1\right], \quad l, h = 0, \ldots, m.$$

Let the adjacency matrix $A = (a_{lh})_{(m+1)\times(m+1)}$ of depots be constructed along the route by the following rule: if there exists an index j such that $x_{lj} = 1$ and $y_{jh} = 1$, then $a_{lh} = 1$. The elements of the matrix $\underbrace{A \cdot A \cdot \ldots \cdot A}_{k} = A^k = (a^k_{lh})_{(m+1)\times(m+1)}$ determine the number of routes of length k between the corresponding depots [12]. If there are $(m+1)$ depots in total, the longest route between them will have length m. Thus, if the matrix A^m contains no zero elements, then it is possible to move between any pair of depots: they are all connected with each other, and there are no subcycles in the route.

Consider the matrix $A_M = (a_{lh})_{M\times M}$ obtained from the matrix A by removing the zero rows and columns, $M \leqslant m$. Consequently, the matrix A_M is an adjacency matrix with the depots included in the route only. If the matrix $(A_M)^k$ with any number $k \geqslant M$ contains no zero elements, then all the depots included in the route are interconnected.

Thus, the condition of no subcycles can be formulated as follows:

$$a^{(m)}_{lh} \geqslant 1, l, \quad h = 0, \ldots, m, \tag{19}$$

where $a^{(m)}_{lh}$ are the elements of the matrix $(A_M)^m$ constructed for the variables x_{ij} and y_{ji}.

The constraints (7)–(19) form the admissible set of solutions Ω.

Thus, the objective function (6) and the constraints (7)–(19) describe the mathematical model of the multi-depot VRP with object alternation and a single collection point.

5. GREEDY HEURISTIC ALGORITHMS

Let us consider a variant of the greedy algorithm for solving the problem. The general idea is to accept locally optimal solutions at each stage, assuming that the final solution will also turn out to be optimal. The algorithm consists in constructing a connected route with the nearest object of the desired type chosen at each stage [13].

In the following, the set of targets will be denoted by J.

Algorithm 1. The direct greedy algorithm.

1. Specify $L = 0,\ J = \{0, \ldots, n\},\ K = 0$, and $i = 0$.

2. If $i = 0$, then find $j' = arg\min_{j \in J \setminus \{0\}}\{c^1_{ij}\}$;
otherwise, find $j' = arg\min_{j \in J}\{c^1_{ij}\}$.

3. Let $x_{ij'} = 1$, $L = L + c^1_{ij'}$.

4. If $j' = 0$, then $i' = 0$, let $y_{00} = y_{00} + 1$ and $K = K + 1$;
otherwise, find $i' = arg\min_{i=1,\ldots,m}\{c^2_{ji}\}$ and let $y_{j'i'} = 1$ and $L = L + c^2_{j'i'}$.

5. If $j' \neq 0$, then update J by $J = J \setminus (j')$.

6. Check: $J = \{0\}$?
If "no," then let $i = i'$ and return to Step 2;
If "yes," then let $x_{i'0} = x_{i'0} + 1,\ L = L + c^1_{i'0},\ y_{00} = y_{00} + 1$, and $K = K + 1$. The answer is obtained.

At Step 5 of the algorithm, the situation with $J = \{0\}$ and $j' = 0$ is impossible: for $j' = 0$, the set J is not updated; if it contained the zero element, the algorithm would terminate at the previous step with the previous number $j' \neq 0$. Thus, the "yes" branch of Step 6 will not duplicate the increase of y_{00} at Step 4 and will not yield the value $x_{00} = 1$. Also, the constraint (11) is satisfied for $i = 0$ at Step 6.

The answer is the matrix-form route with length L and K visits to the collection point.

Note that this algorithm provides at least one departure from and one arrival at the collection point. This is ensured by Steps 1 and 6 on the "yes" branch. Also, the route is closed due to construction. Thus, the route constraints hold.

Algorithm 1 is an intuitive direct solution of the problem. Each of its steps can be a command to a mobile object initially located at the collection point.

Another version of the greedy algorithm does not require starting the route at the collection point. However, it ensures at least one arrival at and one departure from the collection point, i.e., meets the constraints. This algorithm starts at an arbitrary depot. Generally speaking, greedy algorithms do not necessarily start at the first object (the first row of the cost matrix, the first column, the first city, etc.). The problem under consideration has two types of objects. For each of them, we develop the corresponding schemes; see Algorithm 2 and Algorithm 4 below.

Algorithm 2. The greedy algorithm for depots.

1. Specify $L = 0,\ J = \{0, \ldots, n\},\ K = 0$, and an arbitrary index $i_{\text{ini}} = \{0, \ldots, m\}$.

2. If $i_{\text{ini}} = 0$, then execute Algorithm 1 and terminate Algorithm 2; otherwise, proceed to Step 3.

3. Let $x_{i_{\text{ini}}0} = 1,\ L = L + c^1_{i_{\text{ini}}0},\ y_{00} = 1,\ K = K + 1$, and $i = 0$.

4. If $i = 0$, then find $j' = arg\min_{j \in J \setminus \{0\}}\{c^1_{ij}\}$;
otherwise, find $j' = arg\min_{j \in J}\{c^1_{ij}\}$.

5. Let $x_{ij'} = 1$ and $L = L + c^1_{ij'}$.

6. If $j' \neq 0$, then update J: $J = J \setminus (j')$.

7. Check: $J = \{0\}$?
If "yes," then go to Step 10;
if "no," then proceed to Step 8.

8. If $j' = 0$, then let $i' = 0$, $y_{00} = y_{00} + 1$, and $K = K + 1$;
otherwise, let $i' = arg\min_{i=1,\ldots,m}\{c^2_{ji}\}$, $y_{j'i'} = 1$, and $L = L + c^2_{j'i'}$.

9. Let $i = i'$ and return to Step 4.

10. Let $y_{j'i_{\text{ini}}} = 1$ and $L = L + c^2_{j'i_{\text{ini}}}$. The answer is obtained.

We emphasize a mandatory pass through the collection site at Step 3 (the beginning of the algorithm). This step is necessary; otherwise, the algorithm does not ensure visits to dummy objects. At Step 10, the route becomes closed, i.e., the constraint (11) is satisfied for i_{ini}.

This algorithm has an iterative modification: on each iteration, the route construction procedure starts at a new depot. Here we adopt the classical iterative modification scheme of greedy algorithms based on enumerating all possible starting points. We propose two iterative greedy algorithms (Algorithm 3 and Algorithm 5) for the two types of objects.

Algorithm 3. The iterative greedy algorithm for depots.

1. Specify $i_{\text{cur}} = 0$ and $L^* = \infty$.

2. Execute Algorithm 2 with $i_{\text{ini}} = i_{\text{cur}}$. The resulting solution is $L^{i_{\text{cur}}}$, $(x_{ij})^{i_{\text{cur}}}$, $(y_{ji})^{i_{\text{cur}}}$, $K^{i_{\text{cur}}}$.

3. Check: $L^{i_{\text{cur}}} < L^*$?

If "yes," then update L^* by $L^* = L^{i_{\text{cur}}}$, let $(x_{ij})^* = (x_{ij})^{i_{\text{cur}}}$, $(y_{ji})^* = (y_{ji})^{i_{\text{cur}}}$, and $K^* = K^{i_{\text{cur}}}$, and proceed to Step 4;

otherwise, proceed to Step 4 directly.

4. Check: $i_{\text{cur}} = m$?

If "yes," then go to Step 6.

If "no," then proceed to Step 5.

5. Let $i_{\text{cur}} = i_{\text{cur}} + 1$ and return to Step 2.

6. The answer L^*, $(x_{ij})^*$, $(y_{ji})^*$, and K^* is obtained.

Other variants of the greedy algorithm and its iterative modification consist in starting the route at some target. In this case, the mandatory passage through the collection point occurs at the end of the algorithm.

Algorithm 4. The greedy algorithm for targets.

1. Specify $L = 0,\ J = \{0, \ldots, n\},\ K = 0$ and an arbitrary index $j_{\text{ini}} = \{0, \ldots, n\}$.

2. If $j_{\text{ini}} = 0$, then let $i = 0$ and execute Algorithm 1 and terminate Algorithm 4;

otherwise, proceed to Step 3.

3. Let $j = j_{\text{ini}}$.

4. If $j = 0$, then let $i' = 0$, $y_{00} = y_{00} + 1$, and $K = K + 1$;

otherwise, find $i' = arg\min_{i=1,\ldots,m}\{c^2_{ji}\}$ and let $y_{ji'} = 1$ and $L = L + c^2_{ji'}$.

5. If $j \neq 0$, then update J by $J = J\backslash(j)$.

6. Check: $J = \{0\}$?

If "yes," then go to Step 9;

If "no," then proceed to Step 7.

7. If $i' = 0$, then find $j' = arg\min_{j\in J\backslash\{0\}}\{c^1_{i'j}\}$;

otherwise, find $j' = arg\min_{j\in J}\{c^1_{i'j}\}$.

8. Let $x_{i'j'} = 1$, $L = L + c^1_{i'j'}$, and $j = j'$ and return to Step 4.

9. Let $x_{i'0} = x_{i'0} + 1,\ L = L + c^1_{i'0},\ y_{00} = y_{00} + 1,\ K = K + 1,\ x_{0j_{\text{ini}}} = 1,$ and $L = L + c^1_{0j_{\text{ini}}}$. The answer is obtained.

In this algorithm, the route closure at j_{ini} and the mandatory visits to dummy objects (the fulfillment of the constraints (7) and (9)) occur at Step 9.

Let us also consider an iterative modification of this algorithm.

Algorithm 5. The iterative greedy algorithm for targets.

1. Specify $j_{\text{cur}} = 0$ and $L^* = \infty$.

2. Execute Algorithm 4 with $j_{\text{ini}} = j_{\text{cur}}$. The resulting solution is $L^{i_{\text{cur}}}$, $(x_{ij})^{i_{\text{cur}}}$, $(y_{ji})^{i_{\text{cur}}}$, $K^{i_{\text{cur}}}$.

3. Check: $L^{i_{\text{cur}}} < L^*$?

If "yes," then update L^* by $L^* = L^{i_{\text{cur}}}$, let $(x_{ij})^* = (x_{ij})^{i_{\text{cur}}}$, $(y_{ji})^* = (y_{ji})^{i_{\text{cur}}}$, and $K^* = K^{i_{\text{cur}}}$, and proceed to Step 4;

otherwise, proceed to Step 4 directly.

4. Check: $j_{\text{cur}} = n$?

If "yes," then go to Step 6.

If "no," then proceed to Step 5.

5. Let $j_{\text{cur}} = j_{\text{cur}} + 1$ and return to Step 2.

6. The answer L^*, $(x_{ij})^*$, $(y_{ji})^*$, and K^* is obtained.

6. THE ADAPTIVE ALGORITHM

Adaptive algorithms [14] are another possible modification of greedy algorithms. They are based on the transition to a probabilistic problem statement.

An original problem

$$L(X, Y) \to \min_{X,Y \in \Omega}$$

is replaced by a probabilistic one of the form

$$M(L(X, Y)) \to \min_{\{\mathrm{X}\}: X \in \Omega, \{\mathrm{Y}\}: Y \in \Omega},$$

where $\{\mathrm{X}\}$ and $\{\mathrm{Y}\}$ denote the sets of random variables X and Y with realizations X and Y, respectively, from a set Ω.

They have the following structures: $\mathrm{X} = [\mathrm{X}^1, \ldots, \mathrm{X}^n]$ is the matrix of dimensions $(m+1)\times(n+1)$ composed of the rows $\mathrm{X}^i = (\mathrm{X}_{i0}, \mathrm{X}_{i1}, \ldots, \mathrm{X}_{in})$, where X_{ij} is a random variable, $i = 0, \ldots, m$, $j = 0, \ldots, n$; $\mathrm{Y} = [\mathrm{Y}^1, \ldots, \mathrm{Y}^n]$ is the matrix of dimensions $(n+1) \times (m+1)$ composed of the rows $\mathrm{Y}^j = (\mathrm{Y}_{j0}, \mathrm{Y}_{j1}, \ldots, \mathrm{Y}_{jm})$, where Y_{ji} is a random variable, $i = 0, \ldots, m$, $j = 0, \ldots, n$.

Remark 1. To construct the adaptive algorithm, we need only the factual departure of a mobile object (or its absence) from a depot to the dummy target and the factual departure from the dummy target to the dummy depot: the number of such departures does not matter. Thus, the random variables X_{i0}, $i = 0, \ldots, m$, and Y_{00} are supposed to take only two possible values: 0 and 1.

Well, the random variables have the distribution series presented in Table 1.

Table 1. Distribution series

X_{ij}			Y_{ji}		
Values	1	0	Values	1	0
Probabilities	p^1_{ij}	$q^1_{ij} = 1 - p^1_{ij}$	Probabilities	p^2_{ij}	$q^2_{ij} = 1 - p^2_{ij}$

For each $j = 0, \ldots, n$, we introduce the exhaustive events A_{ij} and B_{ij} as follows. The event A_{ij} is that x_{ij} takes value 1 for some $i = 0, \ldots, m$ and the event B_{ji} is that y_{ji} takes value 1 for some $i = 0, \ldots, m$:

$$\sum_{i=0}^{m} p^1_{ij} = 1, \quad j = 0, \ldots, n, \tag{20}$$

$$\sum_{i=0}^{m} p^2_{ji} = 1, \quad j = 0, \ldots, n. \tag{21}$$

There are no initial assumptions about the possible route, and a mobile object may arrive at the target from any depot with equal probability. Therefore, considering the set Ω and conditions (20) and (21), we can define the distributions, e.g., by Table 2.

Table 2. Distribution series

X_{00}			Y_{00}		
Values	1	0	Values	1	0
Probabilities	$p^1_{00}=0$	$q^1_{00}=1$	Probabilities	$p^2_{00}=1$	$q^2_{00}=0$
$X_{i0}, i=1,\dots,m$			$Y_{j0}, j=1,\dots,n, Y_{0i}, i=1,\dots,m$		
Values	1	0	Values	1	0
Probabilities	$p^1_{i0}=\dfrac{1}{m}$	$q^1_{i0}=\dfrac{m-1}{m}$	Probabilities	$p^2_{j0}=0,$ $p^2_{0i}=0$	$q^2_{j0}=1,$ $q^2_{0i}=1$
$X_{ij}, i=0,\dots,m, j=1,\dots,n$			$Y_{ji}, j=1,\dots,n, i=1,\dots,m$		
Values	1	0	Values	1	0
Probabilities	$p^1_{ij}=\dfrac{1}{m+1}$	$q^1_{ij}=\dfrac{m}{m+1}$	Probabilities	$p^2_{ji}=\dfrac{1}{m}$	$q^2_{ji}=\dfrac{m-1}{m}$

In other words, the probability matrices P^1 and P^2 for possible positive values have the form

$$P^1=(p^1_{ij})_{(m+1)\times(n+1)}=\begin{pmatrix} 0 & \dfrac{1}{m+1} & \cdots & \dfrac{1}{m+1} \\ \dfrac{1}{m} & \dfrac{1}{m+1} & \cdots & \dfrac{1}{m+1} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{1}{m} & \dfrac{1}{m+1} & \cdots & \dfrac{1}{m+1} \end{pmatrix},$$

$$P^2=(p^2_{ij})_{(n+1)\times(m+1)}=\begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & \dfrac{1}{m} & \dots & \dfrac{1}{m} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dfrac{1}{m} & \dots & \dfrac{1}{m} \end{pmatrix}.$$

The adaptive algorithm rests on Stages I–III [13]:

I. Specify a motion in the set of random variables X_{ij} and Y_{ji}.

II. Resolve the local improvement condition (LIC), i.e., the inequality

$$M\Big[L\big((X_{ij})^{N+1},(Y_{ji})^{N+1}\big)-L\big((X_{ij})^{N},(Y_{ji})^{N}\big)\Big]\leqslant 0.$$

III. Recalculate the probabilities p^1_{ij} and p^2_{ji}, $i=0,\dots,m$, $j=0,\dots,n$, according to the LIC results.

This algorithm is iterative, and the values $(X_{ij})^{N+1}$ and $(Y_{ji})^{N+1}$ on the $(N+1)$th iteration must be chosen to improve the value of the objective function compared to the previous iteration.

The theoretical grounds of the algorithm were presented in [13]. We just recall the necessary considerations and formulas below.

At Stage I, the motion in the set of random variables is defined by

$$(\mathrm{X})^{N+1} = \overline{u}(\mathrm{X})^N + u(\chi)^{N+1},$$
$$(\mathrm{Y})^{N+1} = \overline{v}(\mathrm{Y})^N + v(\gamma)^{N+1}.$$

Here, $(\chi)^{N+1}$ and $(\gamma)^{N+1}$ are the unknown random variables specifying the direction of motion on the $(N+1)$ iteration. Note that $(\chi)^{N+1} \in \mathrm{X}$ and $(\gamma)^{N+1} \in \mathrm{Y}$, and the random variables χ_{ij} and γ_{ji} have the distribution series given by Table 3.

Table 3. Distribution series

$(\chi)_{ij}$			$(\gamma)_{ji}$		
Values	1	0	Values	1	0
Probabilities	π_{ij}^1	$1-\pi_{ij}^1$	Probabilities	π_{ji}^2	$1-\pi_{ji}^2$

For the random variables u and v, the distribution series are given by Table 4.

Table 4. Distribution series

u			v		
Values	1	0	Values	1	0
Probabilities	p_u	$q_u = 1-p_u$	Probabilities	p_v	$q_v = 1-p_v$

At Stage III, the probabilities are recalculated considering the motion by the formulas

$$(p_{ij}^1)^{N+1} = q_u(p_{ij}^1)^N + p_u(\pi_{ij}^1)^{N+1}, \quad i = 0,\dots,m,\ j = 0,\dots,n,$$
$$(p_{ji}^2)^{N+1} = q_v(p_{ji}^2)^N + p_v(\pi_{ji}^2)^{N+1}, \quad i = 0,\dots,m,\ j = 0,\dots,n.$$

The probabilities π_{ij}^1 and π_{ji}^2 are found from the LIC (Stage II).

The objective function (6) is representable as the sum of two functions of the variables x and y. Therefore, the LIC (Stage II) take the form

$$M\Big[L_{\mathrm{X}}\big((\mathrm{X}_{ij})^{N+1}\big) - L_{\mathrm{X}}\big((\mathrm{X}_{ij})^N\big)\Big] + M\Big[L_{\mathrm{Y}}\big((\mathrm{Y}_{ji})^{N+1}\big) - L_{\mathrm{Y}}\big((\mathrm{Y}_{ji})^N\big)\Big] \leqslant 0.$$

According to [13], the LIC can be written as

$$M\Big[L_{\mathrm{X}}\big((\chi_{ij})^{N+1}\big) - L_{\mathrm{X}}\big((\mathrm{X}_{ij})^N\big)\Big] + M\Big[L_{\mathrm{Y}}\big((\gamma_{ji})^{N+1}\big) - L_{\mathrm{Y}}\big((\mathrm{Y}_{ji})^N\big)\Big]$$
$$= M\Big[L_{\mathrm{X}}\big((\chi_{ij})^{N+1}\big)\Big] - M\Big[L_{\mathrm{X}}\big((\mathrm{X}_{ij})^N\big)\Big] + M\Big[L_{\mathrm{Y}}\big((\gamma_{ji})^{N+1}\big)\Big] - M\Big[L_{\mathrm{Y}}\big((\mathrm{Y}_{ji})^N\big)\Big] \leqslant 0.$$

Let Algorithm 4 be basic for the probabilistic modification. In other words, the algorithm finds the variable $y_{ji'} = 1$ prior to the variable $x_{i'j'} = 1$.

The LIC inequality has an infimum of solutions. We apply the coordinate descent algorithm.

Assume that the lth row of the matrices (c_{ji}^2) and γ (i.e., the unknown random variable γ^l) is considered on the current $(N+1)$th iteration of the algorithm. Suppose that $(\pi_{ji}^2)^{N+1} = (p_{ji}^2)^N$, i.e., the random variables $(\gamma_{ji})^{N+1}$ and $(\mathrm{Y}_{ji})^N$ have the same distribution, $i = 0,\dots,m,\ j \in J$,

$j \neq l$. We resolve the inequality for the unknown variable γ^l:

$$M\Big[L_{\mathrm{X}}\big((\chi_{ij})^{N+1}\big)\Big] - M\Big[L_{\mathrm{X}}\big((\mathrm{X}_{ij})^{N}\big)\Big] + M\Big[L_{\mathrm{Y}}\big((\gamma_{ji})^{N+1}\big)\Big] - M\Big[L_{\mathrm{Y}}\big((\mathrm{Y}_{ji})^{N}\big)\Big]$$
$$= M\Big[L_{\mathrm{X}}\big((\chi_{ij})^{N+1}\big)\Big] - M\Big[L_{\mathrm{X}}\big((\mathrm{X}_{ij})^{N}\big)\Big]$$
$$+ M\left[\sum_{i=0}^{m} c_{li}^2(\gamma_{li})^{N+1} + \sum_{i=0}^{m}\sum_{j\in J, j\neq l} c_{ji}^2(\gamma_{ji})^{N+1}\right]$$
$$- M\left[\sum_{i=0}^{m} c_{li}^2(\mathrm{Y}_{li})^{N} + \sum_{i=0}^{m}\sum_{j\in J, j\neq l} c_{ji}^2(\mathrm{Y}_{ji})^{N}\right] \leqslant 0.$$

In view of $\sum\limits_{i=0}^{m} p_{li}^2 = 1$, we fix a realization of the random variable Y^l. Let $Y^l = e_{i_1}$, where e_{i_1} is a unit vector of dimension $(n+1)$ with 1 at the i_1th position: $y_{li_1} = 1$, $y_{ij} = 0$, $i \neq i_1$, $j \neq l$. Then the expression $M\Big[\sum\limits_{i=0}^{m} c_{li}^2(\mathrm{Y}_{li})^{N} + \sum\limits_{i=0}^{m}\sum\limits_{j\in J, j\neq l} c_{ji}^2(\mathrm{Y}_{ji})^{N}\Big]$ with $\mathrm{Y} : Y \in \Omega$ takes the form

$$c_{li_1}^2 + M_{\mathrm{Y}|Y^l = e_{i_1}}\left[\sum_{i=0}^{m}\sum_{j\in J, j\neq l} c_{ji}^2(\mathrm{Y}_{ji})^{N}\right] = c_{li_1}^2 + \sum_{i=0}^{m}\sum_{j\in J, j\neq l} c_{ji}^2(p_{ji}^2)^{N}.$$

By analogy, for some realization e_{i_2} of the random variable γ^l, due to $\gamma \in \mathrm{Y}$: $Y \in \Omega$ and $(\pi_{ji}^2)^{N+1} = (p_{ji}^2)^{N}$, $i = 0, \ldots, m$, $j \in J$, $j \neq l$, the expression $M\Big[\sum\limits_{i=0}^{m} c_{li}^2(\gamma_{li})^{N+1} + \sum\limits_{i=0}^{m}\sum\limits_{j\in J, j\neq l} c_{ji}^2(\gamma_{ji})^{N+1}\Big]$ takes the form

$$c_{li_2}^2 + \sum_{i=0}^{m}\sum_{j\in J, j\neq l} c_{ji}^2(p_{ji}^2)^{N}.$$

As a result, the expression $M\Big[L_{\mathrm{Y}}\big((\gamma_{ji})^{N+1}\big)\Big] - M\Big[L_{\mathrm{Y}}\big((\mathrm{Y}_{ji})^{N}\big)\Big]$ in the LIC inequality becomes

$$\left(c_{li_2}^2 + \sum_{i=0}^{m}\sum_{j\in J, j\neq l} c_{ji}^2(p_{ji}^2)^{N}\right) - \left(c_{li_1}^2 + \sum_{i=0}^{m}\sum_{j\in J, j\neq l} c_{ji}^2(p_{ji}^2)^{N}\right) = c_{li_2}^2 - c_{li_1}^2. \tag{22}$$

Now we study the other part of the LIC inequality.

Consider the i_2th row of the matrices (c_{ij}^1) and χ, i.e., the unknown random variable χ^{i_2}. Assume that $(\pi_{ij}^1)^{N+1} = (p_{ij}^1)^{N}$, i.e., the random variables $(\chi_{ij})^{N+1}$ and $(\mathrm{X}_{ij})^{N}$ have the same distribution, $i = 0, \ldots, m$, $i \neq i_2$, $j \in J$. We resolve the inequality for the unknown variable χ^{i_2}.

Let the realization of the random row X^{i_2} be fixed, e.g., on the previous iteration of the algorithm, i.e., $\mathrm{X}^{i_2} = e_{l_1,\ldots,l_L}$, where $e_{l_1,\ldots,l_L}$ is the vector of zeros of dimension $(n+1)$ with 1 at the positions $l_1, \ldots, l_L$. In this case, $x_{i_2 l_1} = 1, \ldots, x_{i_2 l_L} = 1$. In view of $\mathrm{X} : X \in \Omega$ (the constraint (7)), the math-

ematical expectation $M\left[L_{\mathrm{X}}((\mathrm{X}_{ij})^N)\right] = M\left[\sum\limits_{j\in J} c^1_{i_2j}(\mathrm{X}_{i_2j})^N + \sum\limits_{\substack{i=0\\ i\neq i_2}}^{m}\sum\limits_{j\in J} c^1_{ij}(\mathrm{X}_{ij})^N\right]$ takes the form

$$c^1_{i_2l_1} + \ldots + c^1_{i_2l_L} + M_{\mathrm{X}|X^{i_2}=e_{l_1,\ldots,l_L}}\left[\sum_{\substack{i=0\\ i\neq i_2}}^{m}\sum_{j\in J, j\neq l_1,\ldots,j\neq l_L} c^1_{ij}(\mathrm{X}_{ij})^N\right]$$

$$= c^1_{i_2l_1} + \ldots + c^1_{i_2l_L} + \sum_{\substack{i=0\\ i\neq i_2}}^{m}\sum_{j\in J, j\neq l_1,\ldots,j\neq l_L} c^1_{ij}(p^1_{ij})^N$$

$$= c^1_{i_2l_1} + \ldots + c^1_{i_2l_L} + \sum_{\substack{i=0\\ i\neq i_2}}^{m}\sum_{j\in J} c^1_{ij}(p^1_{ij})^N$$

$$- \left(\sum_{\substack{i=0\\ i\neq i_2}}^{m} c^1_{il_1}(p^1_{il_1})^N + \ldots + c^1_{il_L}(p^1_{il_L})^N\right).$$

Some index $l_k = 0$ (if any) will not affect the sum in the expectation formula.

Note that several unit variables x_{ij}, $j\in J$ may be selected on one iteration. However, there may exist only one such variable at each step of the algorithm. Thus, one value $x_{i_2q} = 1$ is selected at the current step of the algorithm, i.e., the intermediate realization e_q of χ^{i_2} is fixed. Due to $\chi\in\mathrm{X} : X\in\Omega$ and $(\pi^1_{ij})^{N+1} = (p^1_{ij})^N$, $i = 0,\ldots,m$, $i\neq i_2$, $j\in J$, the expression $M\left[L_{\mathrm{X}}((\chi_{ij})^{N+1})\right] = M\left[\sum\limits_{j\in J} c^1_{i_2j}(\chi_{hj})^{N+1} + \sum\limits_{\substack{i=0\\ i\neq i_2}}^{m}\sum\limits_{j\in J} c^1_{ij}(\chi_{ij})^{N+1}\right]$ takes the form

$$c^1_{i_2q} + \sum_{\substack{i=0\\ i\neq i_2}}^{m}\sum_{j\in J, j\neq q} c^1_{ij}(p^1_{ij})^N = c^1_{i_2q} + \sum_{\substack{i=0\\ i\neq i_2}}^{m}\sum_{j\in J} c^1_{ij}(p^1_{ij})^N - \left(\sum_{\substack{i=0\\ i\neq i_2}}^{m} c^1_{iq}(p^1_{iq})^N\right).$$

Then the expression $M\Big[L_{\mathrm{X}}((\chi_{ij})^{N+1})\Big] - M\Big[L_{\mathrm{X}}((\mathrm{X}_{ij})^N)\Big]$ in the LIC inequality becomes

$$\left[c^1_{i_2q} + \sum_{\substack{i=0\\ i\neq i_2}}^{m}\sum_{j\in J} c^1_{ij}(p^1_{ij})^N - \left(\sum_{\substack{i=0\\ i\neq i_2}}^{m} c^1_{iq}(p^1_{iq})^N\right)\right]$$

$$- \left[c^1_{i_2l_1} + \ldots + c^1_{i_2l_L} + \sum_{\substack{i=0\\ i\neq i_2}}^{m}\sum_{j\in J} c^1_{ij}(p^1_{ij})^N - \left(\sum_{\substack{i=0\\ i\neq i_2}}^{m} c^1_{il_1}(p^1_{il_1})^N + \ldots + c^1_{il_L}(p^1_{il_L})^N\right)\right]$$

$$= \left[c^1_{i_2q} - \left(\sum_{\substack{i=0\\ i\neq i_2}}^{m} c^1_{iq}(p^1_{iq})^N\right)\right] - \left[c^1_{i_2l_1} + \ldots + c^1_{i_2l_L} - \left(\sum_{\substack{i=0\\ i\neq i_2}}^{m} c^1_{il_1}(p^1_{il_1})^N + \ldots + c^1_{il_L}(p^1_{il_L})^N\right)\right] \leqslant 0.$$

Considering (22), the LIC is finally reduced to

$$c^2_{li_2} - c^2_{li_1} + \left[c^1_{i_2q} - \sum_{\substack{i=0\\ i\neq i_2}}^{m} c^1_{iq}(p^1_{iq})^N \right]$$
$$- \left[c^1_{i_2l_1} + \ldots + c^1_{i_2l_L} + \left(\sum_{\substack{i=0\\ i\neq i_2}}^{m} c^1_{il_1}(p^1_{il_1})^N + \ldots + \sum_{\substack{i=0\\ i\neq i_2}}^{m} c^1_{il_L}(p^1_{il_L})^N \right) \right] \leqslant 0 \tag{23}$$

or, equivalently,

$$\left[c^2_{li_2} + c^1_{i_2q} - \sum_{\substack{i=0\\ i\neq i_2}}^{m} c^1_{iq}(p^1_{iq})^N \right]$$
$$- \left[c^2_{li_1} + c^1_{i_2l_1} + \ldots + c^1_{i_2l_L} + \left(\sum_{\substack{i=0\\ i\neq i_2}}^{m} c^1_{il_1}(p^1_{il_1})^N + \ldots + \sum_{\substack{i=0\\ i\neq i_2}}^{m} c^1_{il_L}(p^1_{il_L})^N \right) \right] \leqslant 0. \tag{24}$$

Well, under the fixed realization $Y^l = e_{i_1}$ of the random variable Y^l on the previous iteration, the LIC inequalities (23) will hold if the indices i_2 and q are chosen by minimizing the values $c^2_{li_2}$ and $\left[c^1_{i_2q} - \sum\limits_{\substack{i=0\\ i\neq i_2}}^{m} c^1_{iq}(p^1_{iq})^N \right]$. (Recall that the random variable X^{i_2} is fixed on the previous iteration: $X^{i_2} = e_{l_1,\ldots,l_L}$.)

As a result, the unknown values $(\gamma)^{N+1}$ are given by

$$(\gamma_{l\mu})^{N+1} : \pi^2_{l\mu} = 1 \ \text{ if } \ \min_{i=0,\ldots,m}\{c^2_{li}\} = c^2_{l\mu},$$
$$(\gamma_{li})^{N+1} : \pi^2_{li} = 0, \ \text{ for } \ i = 0,\ldots,m, \ \ i \neq \mu,$$
$$(\gamma^j)^{N+1} = (\mathrm{Y}^j)^{N+1}.$$

At Stage III, the probabilities are recalculated as follows:

$$(p^2_{l\mu})^{N+1} = q_\nu (p^2_{l\mu})^N + p_\nu, \tag{25}$$
$$(p^2_{li})^{N+1} = q_\nu (p^2_{li})^N \text{ for } i = 0,\ldots,m, \ i \neq \mu, \tag{26}$$
$$(p^2_{ji})^{N+1} = (p^2_{ji})^N \text{ for } i = 0,\ldots,m, \ i \in J, \ j \neq l. \tag{27}$$

The value $(\chi)^{N+1}$ is given by

$$(\chi_{\mu\nu})^{N+1} : \pi^1_{\mu\nu} = 1 \text{ if } \min_{j\in J}\left\{ c^1_{\mu j} - \left(\sum_{\substack{i=0\\ i\neq \mu}}^{m} c^1_{ij}(p^1_{ij})^N \right) \right\} = c^1_{\mu\nu} - \left(\sum_{\substack{i=0\\ i\neq \mu}}^{m} c^1_{i\nu}(p^1_{i\nu})^N \right),$$
$$(\chi_{i\nu})^{N+1} : \pi^1_{i\nu} = 0 \ \text{ for } \ i = 0,\ldots,m, \ i \neq \mu,$$
$$(\chi_{ij})^{N+1} = (\mathrm{X}^{ij})^{N+1} \ \text{ for } \ i = 0,\ldots,m, \ j \in J, \ j \neq \nu.$$

The probabilities are recalculated as follows:

$$(p^1_{\mu\nu})^{N+1} = q_u(p^1_{\mu\nu})^N + p_u, \tag{28}$$

$$(p^1_{i\nu})^{N+1} = q_u(p^1_{i\nu})^N \text{ for } i = 0,\dots,m,\ i \neq \mu, \tag{29}$$

$$(p^1_{ij})^{N+1} = (p^1_{ij})^N \text{ for } i = 0,\dots,m,\ i \in J,\ j \neq \nu. \tag{30}$$

The probabilities are not involved in choosing the minimum element of the matrix (c^2_{ij}). Hence, the algorithm will correctly work without recalculating the probability matrix (p^2_{ij}) by formulas (25)–(27).

The resulting probabilities (28)–(30) must be recalculated using the absolute probability formula assuming the uniform distribution of all the hypotheses involved:

$$(\overline{p})^{N+1} = \frac{N}{N+1}\Big((\overline{p})^N + \frac{1}{N}(p)^{N+1}\Big). \tag{31}$$

Here $(\overline{p})^{N+1}$ is the absolute probability and $(p)^{N+1}$ is the conditional probability obtained by formulas (28)–(30).

Thus, the probabilities are adjusted or adapted at each stage based on the previous iterations of the algorithm.

With the proposed solution of the LIC inequality, we modify Step 6 of Algorithm 4 as follows:

Given i', find

$$j' = arg\min_{j\in J}\left\{c^1_{i'j} - \left(\sum_{\substack{i=0\\ i\neq i'}}^{m} c^1_{ij}(p^1_{ij})^N\right)\right\}.$$

Therefore, the adaptive algorithm for solving the problem includes several steps.

Algorithm 6. The adaptive algorithm.

1. Specify the record value $L^* = \infty$, the maximum number $N_{\max}$ of iterations, the current iteration number $N = 1$, the initial probability distribution $(p^1_{ij})^N$ (Table 2), and p_u.

2. Specify $L = 0,\ J = \{0,\dots,n\},\ K = 0$, and $j = 0$.

3. If $j = 0$, then let $i' = 0$, $y_{00} = y_{00} + 1$, and $K = K + 1$;
otherwise, find $i' = arg\min_{i=1,\dots,m}\{c^2_{ji}\}$ and let $y_{ji'} = 1$ and $L = L + c^2_{ji'}$.

4. If $j \neq 0$, then update J by $J = J\backslash(j)$.

5. Check: $J = \{0\}$?
If "yes," then go to Step 10;
if "no," then move to Step 9.

6. If $i' = 0$, then find $j' = arg\min_{j\in J\backslash\{0\}}\left\{c^1_{i'j} - \left(\sum\limits_{\substack{i=0\\ i\neq i'}}^{m} c^1_{ij}(p^1_{ij})^N\right)\right\}$;
otherwise, find $j' = arg\min_{j\in J}\left\{c^1_{i'j} - \left(\sum\limits_{\substack{i=0\\ i\neq i'}}^{m} c^1_{ij}(p^1_{ij})^N\right)\right\}$.

7. Update the probabilities by formulas (28)–(30).

8. Update the absolute probabilities by formula (31).

9. Let $x_{i'j'} = 1$, $L = L + c^1_{i'j'}$, and $j = j'$ and return to Step 3.

10. Let $x_{i'0} = x_{i'0} + 1$, $L = L + c^1_{i'0}$, $y_{00} = y_{00} + 1$, $K = K + 1$, $x_{0j_{\text{ini}}} = 1$, and $L = L + c^1_{0j_{\text{ini}}}$. The resulting solution on the current iteration is L^N, $(x_{ij})^N$, $(y_{ji})^N$, and K^N.

11. Check: $L^N < L^*$?

If "yes," then update L^* by $L^* = L^N$, let $(x_{ij})^* = x^N_{ij}$, $(y_{ji})^* = y^N_{ji}$, and $K^* = K^N$ and proceed to Step 12;

otherwise, proceed to Step 12 directly.

12. Check: $N = N_{\max}$?

If "yes," then go to Step 14;

otherwise, proceed to Step 13.

13. Let $N = N + 1$ and return to Step 2.

14. The answer L^*, $(x_{ij})^*$, $(y_{ji})^*$, and K^* is obtained.

7. A COMPUTATIONAL EXPERIMENT

The proposed algorithms were implemented on the *C#* language in Visual Studio 2019. A computational experiment was carried out on a PC (CPU Intel Pentium N4200 1.1GHz, 4GB RAM, 64-bit operating system) to compare the performance of the algorithms.

For the experiment, we generated cost matrices of different dimensions, 500 matrices for each case. The distances between objects were taken as the cost. The problem dimensions were chosen based on practical considerations that the number of depots is often much smaller than that of targets. The tables below show the results, including the average values of the objective function and the number of visits to the collection site (K). Note that the dimensions in the tables are without the dummy objects of types A and B.

Algorithms 2 and 4 are executed as Algorithm 1 in the case of zero initial indices. Therefore, in the experiment, their initial indices were set equal to 1. Note also that the results for Algorithm 1 are omitted in the tables, as they are similar to those of Algorithm 2.

Table 5 provides the values of the objective function L^* and the obtained value K^*. For the adaptive Algorithm 6, we selected the following parameters: $N_{\max} = 100$ and $p_u = 0.1$.

In Table 5, the best (or two close) values of the objective function for each dimension are set in bold. Note that for most test data, the best results are obtained by the adaptive algorithm; also, good results are shown by Algorithm 5 (the iterative algorithm for targets). The value of the objective function grows with increasing the number of targets (the objects of type A). This result seems obvious: the number of visits for mobile objects raises accordingly. On the other hand, the value of the objective function declines with increasing the number of depots (the objects of type B), which is due to the higher variability of possible routes when mobile objects return from targets. Among the noniterative Algorithms 2 and 4, the best results are demonstrated by Algorithm 2. On the contrary, its iterative modification (Algorithm 3) has a worse performance than the iterative Algorithm 5. This is due to the higher variability of choice when enumerating targets than when enumerating depots.

As has been emphasized, the purpose of this paper is to compare the greedy algorithm with the adaptive algorithm and perform the corresponding analysis. However, based on the publication [10] with similar test data for the exact algorithm, we arrive at the following conclusion: the results of the adaptive algorithm are 2–15% worse than the optimal value, depending on the problem dimension.

According to [15], the optimal value of K is 1 in the case of distance-based cost. This outcome is confirmed by the experiment above. In Table 5, the smallest values of K correspond to the smallest values of the objective function. An additional regression analysis was performed to find a linear

Table 5. Objective function values and the number of mobile objects

Type B × Type A	Algorithm 2		Algorithm 3 (iter.)		Algorithm 4		Algorithm 5 (iter.)		Algorithm 6 (adapt.)	
	L^*	K^*	L^*	K^*	L^*	K^*	L^*	K^*	L^*	K^*
1×10	1106	1.932	1106	1.932	1155	1.87	**1071**	1.298	1086	1
1×30	3188	3.67	3188	3.67	3255	3.582	3159	2.716	**3052**	1.002
1×50	5261	5.478	5261	5.478	5334	5.398	5237	4.502	**4989**	1.006
1×100	10513	10.012	10513	10.012	10591	9.932	10496	9.038	**9891**	1.012
2×10	723	1.004	**711**	1.004	778	1.006	**709**	1.001	768	1.034
2×30	1960	1.005	**1956**	1.002	2043	1.004	**1954**	1.001	2033	1.116
2×50	3211	1.004	**3207**	1.002	3303	1.01	**3206**	1.001	3298	1.258
2×100	6356	1.007	**6354**	1.003	6460	1.02	**6353**	1.003	6461	1.526
5×10	677	2.092	652	2.092	701	1.884	**618**	1.444	**612**	1.078
5×30	1516	1.798	1488	1.798	1592	2.334	1476	1.484	**1447**	1.11
5×50	2802	3.88	2770	3.88	2854	3.8	2745	3.21	**2657**	1.37
5×100	5447	2.02	5439	2.02	5547	2.024	**5426**	1.384	**5428**	1.816
8×10	598	1.512	583	1.512	644	1.648	564	1.162	**553**	1.042
8×30	1554	3.508	1550	3.508	1643	3.522	1529	2.626	**1425**	1.202
8×50	2383	2.726	2368	2.726	2439	2.878	2336	2.146	**2281**	1.28
8×100	5097	5.268	5041	5.268	5117	5.236	4981	4.13	**4740**	1.636
10×10	564	1.644	545	1.644	598	1.678	**513**	1.17	**510**	1.04
10×30	1282	3.08	1277	3.08	1349	3.022	1233	2.08	**1168**	1.126
10×50	2287	2.498	2283	2.498	2362	2.62	2259	1.804	**2215**	1.214
10×100	3465	6.13	3384	6.13	3443	5.512	3331	4.898	**3255**	1.566

Table 6. The coefficients of determination

Type B × Type A	R-squared value of model 1	Type B × Type A	R-squared value of model 2	Type B × type A	R-squared value of model 3
1×10	**0.429364**	1×10		1×10	
1×30	0.840595	1×30	0.98947	2×10	
1×50	0.938962	1×50		5×10	0.561744
1×100	0.984294	1×100		8×10	
2×10	**0.359795**	2×10		10×10	
2×30	**0.311298**	2×30	0.995703	1×30	
2×50	**0.369441**	2×50		2×30	
2×100	0.997976	2×100		5×30	0.684219
5×10	**0.599609**	5×10		8×30	
5×30	0.91009	5×30	0.984412	10×30	
5×50	0.788263	5×50		1×50	
5×100	**0.219113**	5×100		2×50	
8×10	0.789282	8×10	5×50		0.666925
8×30	0.775139	8×30	0.994379	8×50	
8×50	0.854995	8×50		10×50	
8×100	0.964218	8×100		1×100	
10×10	0.759837	10×10		2×100	
10×30	0.797575	10×30	0.904579	5×100	0.653224
10×50	0.729778	10×50		8×100	
10×100	0.751988	10×100		10×100	

Table 7. Comparison of iterative Algorithms 3, 5, and 6

Type B × Type A	Algorithm 3 (iter.)	Algorithm 5 (iter.)	Algorithm 6 (adapt.)	Algorithm 6 (adapt.) 100 it.
10×10	545	**513**	523	**510**
30×30	936	892	**856**	**844**
50×50	1300	1223	**1159**	**1152**
100×100	1647	1547	**1501**	**1501**

relationship between K and the values of the objective function. The model was assessed using three data sets: the first model was constructed on the values of all algorithms for one dimension; the second model, for a fixed number of depots (the objects of type B) based on the results of the adaptive algorithm; the third model, for a fixed number of targets (the objects of type A) based on the results of the adaptive algorithm. Table 6 shows the coefficients of determination of these models.

For the first data set, we see a very pronounced linear dependence in most cases (the coefficient of determination is close to 1), but there are dimensions with unconfirmed linear dependence (set in bold in the table). The second data set validated the linear dependence of K on the values of the objective function. In the third case, the coefficient of determination varies from 0.56 to 0.66, which cannot confirm the linear dependence.

Based on this table, we can hypothesize the linear (or almost linear) dependence of K on the values of the objective function. However, this issue needs a detailed study, going beyond the scope of the paper.

In Table 5, the fixed value $N_{\max} = 100$ was chosen for the adaptive algorithm in all dimensions. However, in iterative greedy algorithms, the number of iterations varies depending on the problem dimension. Three iterative algorithms were compared under the same number of iterations. For this purpose, we generated additional cost matrices of dimensions 30×30, 50×50, and 100×100. The results are presented in Table 7, which has two columns for the adaptive Algorithm 6. In the first case, the number of iterations was equal to the problem dimension, coinciding with the number of iterations of Algorithms 3 and 5; in the second case, $N_{\max} = 100$.

Clearly, Algorithm 3 (the iterative algorithm for depots) has worse performance than Algorithm 5 (the iterative algorithm for targets). The adaptive Algorithm 6 gives a better or similar result under the same number of iterations as the other algorithms; moreover, this algorithm improves its own result under more iterations.

Table 8 presents a similar comparative analysis. Here, the adaptive algorithm is compared to Algorithm 5 under the same number of iterations. The data from Table 5 with $N_{\max} = 100$ are provided as well.

Algorithm 5 produces significantly better results than the adaptive Algorithm 6 only in the problems with 2 depots (the objects of type B).

Table 9 shows the running time of the algorithms in milliseconds.

Obviously, Algorithms 2 and 4 work very fast. The iterative Algorithm 3 for depots does so as well, and its running time is more dependent on the number of depots. The running time of the iterative Algorithm 5 depends on the number of targets, significantly exceeding that of Algorithm 3 since the number of targets is greater than the number of depots. The adaptive Algorithm 6 works much longer than the others. This result is due to a given number of iterations, recalculation of probabilities, and, most importantly, additional computation of the sums to find the minimum (Step 6). Additional modifications to the procedures for calculating and recalculating these sums would significantly improve its running time.

Table 8. Comparison of iterative Algorithms 5 and 6

Type B × Type A	Algorithm 5 (iter.)	Algorithm 6 (iter.)	Algorithm 6 (adapt.) 100 it.
1 × 10	**1071**	1086	1086
1 × 30	3159	**3052**	**3052**
1 × 50	5237	**4989**	**4989**
1 × 100	10496	**9891**	**9891**
2 × 10	**709**	768	768
2 × 30	**1954**	2033	2003
2 × 50	**3206**	3298	3298
2 × 100	**6353**	6461	6461
5 × 10	618	**616**	**612**
5 × 30	1476	**1451**	**1447**
5 × 50	2745	**2658**	**2657**
5 × 100	**5426**	**5428**	**5428**
8 × 10	564	564	**553**
8 × 30	1529	**1431**	**1425**
8 × 50	2336	**2283**	**2281**
8 × 100	4981	**4740**	**4740**
10 × 10	**513**	523	**510**
10 × 30	1233	**1175**	**1168**
10 × 50	2259	**2217**	**2215**
10 × 100	3331	**3255**	**3255**

Table 9. Running time of algorithms

Type B × Type A	Algorithm 2	Algorithm 3 (iter.)	Algorithm 4	Algorithm 5 (iter.)	Algorithm 6 (adapt.)
1 × 10	< 0.001	< 0.001	< 0.001	0.002	23.19
1 × 30	< 0.001	< 0.001	< 0.001	1.006	51.738
1 × 50	0.012	0.001	0.002	4.37	74.648
1 × 100	< 0.001	0.002	< 0.001	28.63	95.238
2 × 10	< 0.001	< 0.001	< 0.001	0.003	22.236
2 × 30	< 0.001	< 0.001	< 0.001	1.002	44.26
2 × 50	< 0.001	0.002	0.001	4.324	66.134
2 × 100	0,004	0.014	0.002	27.744	119.342
5 × 10	< 0.001	< 0.001	< 0.001	0.004	23.758
5 × 30	< 0.001	< 0.001	0.001	1.006	49.384
5 × 50	< 0.001	0.004	< 0.001	4.67	81.254
5 × 100	< 0.001	1	< 0.001	30.418	185.048
8 × 10	< 0.001	0.001	< 0.001	0.002	25.652
8 × 30	< 0.001	0.001	< 0.001	1.036	63.858
8 × 50	< 0.001	0.01	< 0.001	4.766	95.93
8 × 100	< 0.001	2.002	< 0.001	29.062	217.754
10 × 10	< 0.001	< 0.001	< 0.001	0.004	25.658
10 × 30	< 0.001	0.001	< 0.001	1.038	56.93
10 × 50	< 0.001	0.144	< 0.001	4.876	97.696
10 × 100	< 0.001	2.83	< 0.001	30.914	238.14

An important advantage of the adaptive Algorithm 6 compared to Algorithms 2–5 is the presence of tunable parameters: p_u and $N_{\max}$. Their competent choice can improve the search capabilities of the algorithm.

Table 10. The effect of parameter p_u

p_u	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
5×30	**1447**	1457	1466	1473	1479	1485	1490	1494	1498	1503
p_u	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1
5×30	**1438**	1439	1439	1441	1441	1442	1444	1445	1446	1447
p_u	0.001	0.002	0.003	0.004	0.005	0.006	0.007	0.008	0.009	0.01
5×30	1485	1460	1450	1445	1443	1441	1439	**1438**	1439	**1438**

Table 11. The effect of iterations

$N_{\max}$	10	20	30	40	50	60	70	80	90	100
5×30	1495	1466	1454	1448	1445	1442	1440	1439	**1438**	**1438**
$N_{\max}$	100	200	300	400	500	600	700	800	900	1000
5×30	1438	1437	**1436**	**1436**	**1436**	**1436**	**1436**	**1436**	**1436**	**1436**

Tables 10 and 11 show the tuning results for these parameters. In Table 10, $N_{\max} = 100$.

During the first experiment, we varied the parameter p_u from 0.1 to 1. As was discovered, the results improve with decreasing the values of this parameter. Therefore, the values from 0.01 to 0.1 were taken for the next experiment; the results were better for smaller values. However, in the third experiment with the values from 0.001 to 0.01, this trend disappeared, and further decrease of the parameter p_u led to higher values of the objective function. Thus, the optimal value of the parameter p_u was found to be 0.01. However, this parameter value is optimal for the dimension 5×30.

The next experiment was conducted to tune the maximum number of iterations $N_{\max}$ under the given value $p_u = 0.01$.

Recall that the algorithm depends on the probabilities that are changed and adjusted between iterations. Therefore, it seems reasonable to expect better results from increasing the number of iterations. This hypothesis was confirmed in the two experiments performed. Note that the algorithm has a noticeable improvement in the results when increasing the maximum number of iterations from 10 to 100; however, the increase from 100 to 1000 changes the results by 2 units only. Moreover, increasing the maximum number of iterations further to 5000 even reduces the value of the objective function by 1 unit, to 1435. Thus, the number of iterations above 100 does not significantly improve the results of the algorithm.

Based on the results of the experiments, we can draw the following general conclusions:

1. Noniterative greedy algorithms are fast, but their results are worse compared to the other algorithms.

2. Iterative greedy algorithms improve the results of noniterative algorithms but take longer to execute. The iterative algorithm for targets gives the average value of the objective function smaller than the iterative algorithm for depots. This outcome equally applies to the case with more targets than depots (i.e., a different number of iterations in the algorithms) and the case with equal targets and depots (i.e., the same number of iterations in the algorithms).

3. The adaptive algorithm yields a smaller value of the objective function than the other algorithms, but its running time is much higher. This result is again confirmed for the different number of iterations in the adaptive algorithm compared to the iterative greedy ones.

4. The adaptive algorithm depends on the parameters and their proper tuning can improve the performance.

5. The value K is hypothesized to have a linear dependence on the values of the objective function.

8. CONCLUSIONS

This paper has presented the multi-depot vehicle routing problem with object alternation. Mathematical models with two blocks of variables have been constructed for the problem with and without a single collection point. Greedy, iterative greedy, and adaptive algorithms have been proposed for solving the problem with a single collection point. The algorithms have been implemented on the $C\#$ language in Visual Studio 2019, and a computational experiment has been conducted; see the main results in the corresponding section. In the future, a detailed study should address the dependence of the value K on the values of the objective function. Since the adaptive algorithm strongly depends on the parameters, further research should aim at finding their optimal values depending on the problem dimension. It is also necessary to consider the functional change of the parameter p_u depending on the current iteration number or the Hamming distance between two solutions.

REFERENCES

1. Benediktovich, V.I., Demidenko, V.M., Dymkov, M.P., and Brilevskii, A.O., Vehicle Routing Models and Their Application in Logistic Supply Chains of Commodity Distribution Networks, in *Ekonomika, modelirovanie, prognozirovanie* (Economics. Modeling. Forecasting), vol. 6, 2012, pp. 94–106.

2. Litvinchev, I.S., Cedillo, G., and Velarde, M., Integrating Territory Design and Routing Problems, *J. Comput. Syst. Sci. Int.*, 2017, vol. 56, no. 6, pp. 969–974. https://doi.org/10.1134/S1064230717060120

3. Kosonogova, L.G., Koroleva, A.A., and Dubasov, A.V., Analysis of the Optimal Traffic Flow Distribution When Routing the Number of Vehicles, *Vestnik: Nauchnyi Zhurnal*, 2021, no. 6(48), pp. 81–85.

4. Yusupova, N.I. and Valeev, R.S., A Routing Problem for the Delivery of a Homogeneous Product to Various Clients by Motor Vehicles, *Modern High Technologies*, 2020, no. 4, pp. 84–88.

5. Zhou, Y., Li, W., Wang, X., Qiu, Y., and Shen, W., Adaptive Gradient Descent Enabled Ant Colony Optimization for Routing Problems, *Swarm and Evolutionary Computation*, 2022, vol. 70(3), art. no. 101046. https://doi.org/10.1016/j.swevo.2022.101046

6. Ramalingam, A. and Vivekanandan, K., Genetic Algorithm Based Solution Model for Multi-depot Vehicle Routing Problem with Time Windows, *International Journal of Advanced Research in Computer and Communication Engineering*, 2014, vol. 3, no. 11, pp. 8433–8439.

7. Mazidi, A., Fakhrahmad, M., and Sadreddini, M., A Meta-heuristic Approach to CVRP Problem: Local Search Optimization Based on GA and Ant Colony, *Journal of Advances in Computer Research*, 2016, vol. 7, no. 1, pp. 1–22.

8. Medvedev, S.N., Medvedeva, O.A., Zueva, Y.R., and Chernyshova, G.D., Formulation and Algorithmization of the Interleaved Vehicle Routing Problem, *Journal of Physics: Conference Series*, 2019, vol. 1203, art. no. 012053. https://doi.org/10.1088/1742-6596/1203/1/012053

9. Medvedev, S., Sorokina, A., and Medvedeva, O., The Vehicle Routing Problem for Several Agents among the Objects of Two Types, *Proc. 2019 XXI International Conference "Complex Systems: Control and Modeling Problems" (CSCMP)*, Samara, 2019, pp. 535–540.

10. Medvedev, S.N., A Mathematical Model and an Algorithm for Solving a Multi-Depot Vehicle Routing Problem with a Single End Point, *Proceedings of Voronezh State University. Series: Systems Analysis and Information Technologies*, 2021, no. 1, pp. 21–32. https://doi.org/10.17308/sait.2021.1/3368

11. Kenzin, M.Y., Bychkov, I.V., and Maksimkin, N.N., Rich Vehicle Routing Problem for the Multi-robot Environmental Monitoring, *Izvestiya SFedU. Engineering Sciences*, 2019, no. 7, pp. 82–92.

12. Christofides, N., *Graph Theory. An Algorithmic Approach*, Academic Press, 1975.

13. Gol'shtein, E.G. and Yudin, D.B., *Zadachi lineinogo programmirovaniya transportnogo tipa* (Transportation Linear Programming Problems), Moscow: Nauka, 1969.

14. Medvedev, S.N. and Medvedeva, O.A., An Adaptive Algorithm for Solving the Axial Three-Index Assignment Problem, *Autom. Remote Control*, 2019, vol. 80, no. 4, pp. 718–732.

15. Medvedev, S.N., On the Optimal Solution of the Vehicle Routing Problem with Object Alternation and a Single Collection Point, *Trudy Vserossiiskoi nauchnoi konferentsii "Sovremennye metody prikladnoi matematiki, teorii upravleniya i komp'yuternykh tekhnologii" (PMTUKT-2021)* (Proc. All-Russian Sci. Conf. "Modern Methods of Applied Mathematics, Control Theory and Computer Technolgies" (AMCTCT-2021)), Voronezh, 2021, pp. 97–101.

This paper was recommended for publication by A.A. Lazarev, a member of the Editorial Board