Formulation and algorithmization of the interleaved vehicle routing problem

Medvedev S N, Medvedeva O A, Zueva Y R and Chernyshova G D

Department of Applied Mathematics, Informatics and Mechanics, Voronezh State University, 1, University Square, Voronezh, 394018, Russia

E-mail: s_n_medvedev@mail.ru

Abstract. The problem of routing between two types of objects is considered. Two variants of the problem statement are considered: with a single point of collection and without it. An important aspect of the routes is the mandatory alternation of different types of objects. A mathematical model of the problem with boolean variables is proposed for each statement. A meaningful statement can be represented as the task of collecting haystacks in trucks afield. Two algorithms are developed to solve this problem. The first one is based on the "greedy" strategies, and the second one is based on the approach of Ant Colony Optimization (ACO) algorithms. To configure the parameters of the second algorithm it is carried out the computational experiment. The results of the experiment are provided. In conclusion, the main results of the work are presented.

1. Problem statement

Consider two statements of the Vehicle Routing Problem with a special requirement for routing.

1.1. The interleaved routing problem

There is a set of stationary objects of two types A and B, which are located in some given area, and the distances between all objects are known. It is necessary to visit all the stationary objects with the help of several moving objects of type C so that the total distance traveled is minimal (figure 1). Moreover, the objects of types A and B must alternate in the route of each object of type C. Every object of class A can only be visited once, and an object of type B can be visited any number of times.

Thus, it is necessary to construct a route for each mobile object of type C so that the total length of all routes is minimal.

This problem is closely related to such problems as the transport problem (the Monge-Kantorovich problem), the traveling salesman problem and the problem with several traveling salesmen, the planar and axial assignment problems and traveling salesman problem [1] and refers to the Vehicle Routing Problems [2]. To be exact, it refers to the Vehicle Routing Problem with Several Depots [3]. However, the problem under consideration has gone beyond the scope of the studied problems because of 1) the requirement of mandatory alternation of stationary objects of different types; 2) the absence of requirement to return to the same object of type B (in the same depot) from which the moving object came.
1.2. The interleaved routing problem with a single point of collection

We add the following requirement to the problem statement: let all moving objects of type C start and finish their routes at some fixed point of collection (figure 2).

This formulation is closer to the classical formulation of the single-depot Vehicle Routing Problem. However, in the classical formulation the stationary objects (customers) are not divided into two types, i.e., in each of them we can come, and from each we can leave only once. In the proposed formulation of the problem stationary objects are divided into two types A and B with a different admissible number of visits, and it is important that in they should be interchanged.

As the point of collection we can define some stationary object of new type D. But it is more convenient to introduce a fictitious additional object of one of the existing types of stationary objects A and B. We assume that the route must start and end in objects of type B. This requirement can be interpreted in different ways: considering and not considering the starting point of collection. If the point of collection meets a requirement, it must be a fictitious object of type B. Then the route starts and ends at the object of type B. But if we assume that the point of collection is not included in the route, it must be a fictitious object of type A. Then, when we leave the object of type A, we come to the object of type B, and the route starts. And after the end of the route in the object of type B, we
come to the point of collection, in the object of type A. Thus the requirement of alternation is not violated.

Note that in both cases the route becomes closed.

2. Mathematical model

For the proposed formulations of routing problems, we will make mathematical models in the form of mathematical discrete programming models, i.e. we will form an acceptable set of solutions based on constraints using Boolean variables.

2.1. A mathematical model of Interleaved routing problem

A sufficient number of works and books is devoted to modeling of various problems of discrete optimization [1-12]. It should be noted that modeling of each problem always combines an individual approach [2, 3, 6-12] and classical statements [1, 4, 5].

From problem statement we obtain the conditions for the mathematical model:

1) The total length of routes should be minimal;
2) Each object of type A must be visited by only one object of type C and only once;
3) Only one object of type C must leave each object of type A and only once;
4) If an object of type C comes to an object of type A, it also leaves this object of type A;
5) An object of type B can be either the beginning, an internal element or the end of a route;
6) The route of an object of type C must be connected.

Note that conditions 2, 3 are conditions similar to the conditions of the traveling salesman problem; conditions 4, 5, 6 are conditions of connectivity or continuity of the route.

We introduce the following notations:

- $m$ – a number of stationary objects of type A, index $j = 1, m$;
- $n$ – a number of stationary objects of type B, index $i = 1, n$;
- $K$ – a number of moving objects of type C, index $k = 1, K$;
- $C = (c_{ij}), i = 1, n, j = 1, m$ – the matrix of distances between objects of type A and type B.

We introduce two blocks of variables:

\[ x_{ij}^k = \begin{cases} 
1, & \text{if the } k \text{-th object } C \text{ comes from the } i \text{-th object } B \text{ to the } j \text{-th object } A, \\
0, & \text{otherwise,}
\end{cases} \]

\[ y_{ji}^k = \begin{cases} 
1, & \text{if the } k \text{-th object } C \text{ leaves from the } j \text{-th object } A \text{ to the } i \text{-th object } B, \\
0, & \text{otherwise,}
\end{cases} \]

where $i = 1, n$, $j = 1, m$, $k = 1, K$.

We will build a mathematical model of the problem on the basis of the above conditions and the entered designations.

1) The total length of routes should be minimal. The objective function of this problem has a form:

\[ \sum_{k=1}^{K} \sum_{j=1}^{m} \sum_{i=1}^{n} c_{ij}(x_{ij}^k + y_{ji}^k) \rightarrow \min. \]

The constraints of the problem have a form:

2) Each object of type A must be visited by only one object of type C and only once:

\[ \sum_{k=1}^{K} \sum_{i=1}^{n} x_{ij}^k = 1, \forall j = 1, m. \]

3) Only one object of type C must leave each object of type A and only once

\[ \sum_{k=1}^{K} \sum_{j=1}^{n} y_{ji}^k = 1, \forall j = 1, m. \]

4) If the $k$-th object of type C comes to the $j$-th object of type A, it also leaves this object of type A:

\[ \sum_{i=1}^{n} x_{ij}^k = \sum_{j=1}^{n} y_{ji}^k, \forall j = 1, m, k = 1, K. \]
5) Each \(i\)-th object of type B can be either the beginning, an internal element or the end of a route:

\[
|\sum_{j=1}^{m} y_{ij}^{k} - \sum_{j=1}^{m} x_{ij}^{k}| \leq 1, \forall i = 1, n, \forall k = 1, K.
\]

This restriction means that some moving object can leave the object of type B without coming to it (the beginning of the route):

\[
\sum_{j=1}^{m} y_{ij}^{k} - \sum_{j=1}^{m} x_{ij}^{k} = -1,
\]

can come and leave it one or more times:

\[
\sum_{j=1}^{m} y_{ij}^{k} = \sum_{j=1}^{m} x_{ij}^{k},
\]

can not come and not leave from it:

\[
\sum_{j=1}^{m} y_{ij}^{k} = \sum_{j=1}^{m} x_{ij}^{k} = 0,
\]

and can come to it, not leaving from it (the end of the route):

\[
\sum_{j=1}^{m} y_{ij}^{k} - \sum_{j=1}^{m} x_{ij}^{k} = 1,
\]

where \(i = 1, n, k = 1, K\).

6) The route of an object of type C must be connected.

Let \(J^{k} = \{j: x_{ij}^{k} = 1\}\) is the set of indices of objects of type A, which were visited by an object of type C; \(N^{k} = \sum_{j=1}^{m} \sum_{i=1}^{n} x_{ij}^{k}\) is the number of objects of type A, which were visited by an object of type C, i.e., \(|J^{k}| = N^{k}, \forall k = 1, K\). Note that \(N^{k}\) depends on the variables \(x_{ij}^{k}\). Let \(J^{k}\) is the set obtained from the set \(J^{k}\) by removing any index. Then the necessary restriction will have the form:

\[
u_{i} - u_{i} + N^{k}(\sum_{j=1}^{m} x_{ij}^{k} y_{ij}^{k}) = N^{k} - 1, \forall i, l \in J^{k}, i \neq l, \forall k = 1, K,
\]

where \(u_{i}\) are variables that take arbitrary real values. It is possible to show [1] that such values can be considered as non-negative integers. For example, it is convenient to take the sequence number of an object of type A in the route as the values of these variables.

As a result, a mathematical model is obtained for the problem, which has the following form:

\[
\sum_{k=1}^{K} \sum_{j=1}^{m} \sum_{i=1}^{n} c_{ij}(x_{ij}^{k} + y_{ij}^{k}) \rightarrow \min
\]

(1)

\[
\sum_{k=1}^{K} \sum_{j=1}^{m} x_{ij}^{k} = 1, \forall j = 1, m,
\]

(2)

\[
\sum_{k=1}^{K} \sum_{i=1}^{n} y_{ij}^{k} = 1, \forall j = 1, m,
\]

(3)

\[
\sum_{i=1}^{n} x_{ij}^{k} = \sum_{j=1}^{n} y_{ij}^{k}, \forall j = 1, m, k = 1, K,
\]

(4)

\[
|\sum_{j=1}^{m} y_{ij}^{k} - \sum_{j=1}^{m} x_{ij}^{k}| \leq 1, \forall i = 1, n, \forall k = 1, K,
\]

(5)

\[
u_{i} - u_{i} + N^{k}(\sum_{j=1}^{m} x_{ij}^{k} y_{ij}^{k}) = N^{k} - 1, \forall i, l \in J^{k}, i \neq l, \forall k = 1, K,
\]

(6)

\[
x_{ij}^{k} \in (0, 1), \forall i = 1, n, j = 1, m, k = 1, K,
\]

(7)

\[
y_{ij}^{k} \in (0, 1), \forall i = 1, n, j = 1, m, k = 1, K,
\]

(8)

\[
u_{i} \in \mathbb{Z}, \forall i = 1, n.
\]

(9)

Note that the constraints (2)-(3) belong to the type of constraints of the assignment problem or the traveling salesman problem, and the constraints (4)-(5) are balance equations. The restriction (6) is a similarity to the restriction of the absence of sub-cycles for the Traveling Salesman Problem [1].

Let us consider the restriction (4). Strictly speaking, the condition "if-then" should be written in the form

\[
(\sum_{i=1}^{n} x_{ij}^{k})(1 - \sum_{i=1}^{n} y_{ij}^{k}) = 0, \forall j = 1, m, k = 1, K.
\]
However, condition 4 can be reformulated as follows: if \(j\)-th object of type A leaves \(k\)-th object of type C, it must first come to this object of type A.

\[
\left(\sum_{l=1}^{n} y_{jl}^k\right)\left(1 - \sum_{l=1}^{n} x_{ij}^k\right) = 0, \forall j = \overline{1,m}, k = \overline{1,K}.
\]

Solving the system of two equations, we obtain the final restriction (4)

\[
\sum_{i=1}^{n} x_{ij}^k = \sum_{l=1}^{n} y_{jl}^k, \forall j = \overline{1,m}, k = \overline{1,K}.
\]

Thus, the proposed problem is a problem of discrete optimization.

2.2. A mathematical model of Interleaved Routing Problem with a single point of collection

First, consider the case where the point of collection is an object of type B.

We introduce a fictitious \(0\)-th object of type B, \(\bar{B}_0\). Then in the objective function (1) the summation over \(i\) starts with 0. In restrictions (5), (7)-(9) the \(i\)-indexing starts with 0. Condition 5 will take the form:

5*) If an object of type C comes to the object of type B, it leaves the object of type B.

Then the restriction (5) holds as the sum equality

\[
\sum_{j=1}^{m} y_{ji}^k = \sum_{j=1}^{m} x_{ij}^k, \forall i = \overline{0,n}, \forall k = \overline{1,K}.
\]

Also, we add the constraints describing the \(0\)-th object. Namely, all moving objects must leave the \(0\)-th object:

\[
\sum_{j=1}^{m} x_{0j}^k = 1, \forall k = \overline{1,K}
\]
or

\[
\sum_{k=1}^{K} \sum_{j=1}^{m} x_{0j}^k = K.
\]

All moving objects must come back to the \(0\)-th object:

\[
\sum_{j=1}^{m} y_{j0}^k = 1, \forall k = \overline{1,K}.
\]

However, the latter restriction is redundant.

Now consider the case when the point of collection is an object of type A.

We introduce a fictitious \((m+1)\)-th object of type A, \(\bar{A}_{m+1}\). Then in the objective function (1) and restriction (6) the summation over \(j\) ends with a number \((m+1)\). In restrictions (4), (7)-(8) the \(j\)-indexing ends with a number \((m+1)\). Then the restriction (5) holds as the sum equality

\[
\sum_{j=1}^{m+1} y_{ji}^k = \sum_{j=1}^{m+1} x_{ij}^k, \forall i = \overline{1,n}, \forall k = \overline{1,K}.
\]

Restrictions describing \((m+1)\)-th object will also be added. Namely, all moving objects must leave the \((m+1)\)-th object:

\[
\sum_{i=1}^{n} x_{i(m+1)}^k = 1, \forall k = \overline{1,K}
\]
or

\[
\sum_{k=1}^{K} \sum_{i=1}^{n} x_{i(m+1)}^k = K.
\]

All moving objects must come back to the \((m+1)\)-th object:

\[
\sum_{i=1}^{n} y_{i(m+1)}^k = 1, \forall k = \overline{1,K}.
\]

However, the latter restriction is redundant due to (4).
3. **Meaningful problem statement**

A meaningful statement can be represented as the task of collecting haystacks in trucks afield. Then the haystacks and trucks are stationary objects (types A and B respectively), and the tractors are moving objects of type C. Indeed, the tractor drives up to each haystack and picks it up only once; the tractor drives up to each truck any number of times (subject to trucks capacity). We will assume that the tractor moves only one haystack at a time and the total loading capacity of the trucks allows to collect all the haystacks from the field. It is necessary to collect all the haystacks in trucks so that the total distance traveled by tractors was minimal.

Note that all tractors start from some collection point and return to this point at the end of the work. Thus, we obtain an Interleaved routing problem with a single point of collection. However, consider the difference that arises in connection with the logic of collecting haystacks in trucks. At the beginning of work, the tractor goes to some haystack, not to the truck. At the end of work, the tractor collects the last haystack, moves it to the truck and then goes to the collection point. Thus, the route begins in an object of type A, and ends in an object of type B. Therefore, it is necessary to introduce two fictitious objects: the \(0\)-th truck and the \((m+1)\)-th haystack, the distance between these objects is equal to zero.

Hence, we must add the constraints that describe these new objects. Namely, all tractors leave from \(0\)-th truck, but no one needs to come back to the \(0\)-th truck:

\[
\sum_{j=1}^{\text{m}} x_{0j}^k = 1, \forall k = \overline{1,K};
\]

\[
\sum_{j=1}^{\text{m}} y_{0j}^k = 0, \forall k = \overline{1,K}.
\]

All tractors arrive to the \((m+1)\)-th haystack, but no one needs to go away from the \((m+1)\)-th haystack:

\[
\sum_{i=1}^{\text{n}} x_{i(m+1)}^k = 1, \forall k = \overline{1,K};
\]

\[
\sum_{i=1}^{\text{n}} y_{i(m+1)}^k = 0, \forall k = \overline{1,K}.
\]

As a result, the mathematical model of the problem will have the following form:

\[
\sum_{k=1}^{K} \sum_{j=1}^{m+1} \sum_{i=0}^{n} c_{ij} (x_{ij}^k + y_{ij}^k) \to \text{min}
\]

\[
\sum_{k=1}^{K} \sum_{i=1}^{n} x_{ij}^k = 1, \forall j = \overline{1,m},
\]

\[
\sum_{k=1}^{K} \sum_{i=1}^{n} y_{ij}^k = 1, \forall j = \overline{1,m},
\]

\[
\sum_{j=0}^{m} x_{ij}^k = \sum_{i=1}^{n} y_{ij}^k, \forall j = \overline{1,m}, k = \overline{1,K},
\]

\[
\sum_{j=1}^{m} y_{ij}^k = \sum_{j=1}^{m+1} x_{ij}^k, \forall i = \overline{1,n}, \forall k = \overline{1,K},
\]

\[
u_i - u_l + NK (\sum_{j=1}^{m+1} x_{ij}^k y_{ij}^k) = N^k - 1, \forall i, l \in \{k, i \neq l, \forall k = \overline{1,K},
\]

\[
\sum_{i=1}^{n} x_{0j}^k = 1, \forall k = \overline{1,K};
\]

\[
\sum_{j=1}^{m} y_{0j}^k = 0, \forall k = \overline{1,K};
\]

\[
\sum_{i=1}^{n} x_{i(m+1)}^k = 1, \forall k = \overline{1,K};
\]

\[
\sum_{i=1}^{n} y_{i(m+1)}^k = 0, \forall k = \overline{1,K};
\]

\[
 x_{ij}^k \in \{0,1\}, \forall i = \overline{1,n}, j = \overline{1,m+1}, k = \overline{1,K},
\]

\[
y_{ij}^k \in \{0,1\}, \forall i = \overline{0,n}, j = \overline{1,m+1}, k = \overline{1,K},
\]

\[
u_i \in \mathbb{Z}, \forall i = \overline{1,n}.
\]
Denote by $\overline{W}_i$ the loading capacity of the $i$-th truck $i = \overline{1,n}$. Consider a condition that the quantity of the haystacks shipped in each truck has to be no more than its loading capacity:

$$\sum_{k=1}^{K} \sum_{j=1}^{n_i} y_{ji}^k \leq \overline{W}_i, \forall i = \overline{1,n}. \quad (23)$$

Note also that the formulation of the problem implies the compatibility condition: the total number of haystacks doesn’t exceed the total loading capacity of all trucks:

$$\sum_{i=1}^{n} \overline{W}_i \geq m.$$

4. "Greedy" algorithm

Consider the developed "greedy" algorithm for solving the problem (10)-(22) [1, 5]. The main idea of the algorithm is as follows: each tractor passes a mini-route "truck-haystack-truck" one after the other, finding the nearest unharvested haystack or unloaded truck, respectively.

We introduce the following notations:

- $SUM$ is the total distance traveled by the tractors during transportation all the haystacks from the field in trucks;
- $\overline{W}_i$ is the loading distance traveled by the tractors during transportation all the haystacks from the field in the $i$-th truck, $i = \overline{1,n}$;
- $W_i$ is the number of haystacks loaded into the $i$-th truck, $i = \overline{1,n}$;
- $I = \{i: W_i < \overline{W}_i, i = \overline{1,n}\}$ is the set of indices of unloaded trucks;
- $J = \{1,2,...,m\}$ is the set of indices of unharvested haystalks;
- $ind_{car}k$ is the index of the truck which the $k$-th tractor arrived to, $k = \overline{1,K}$.

Algorithm 1. «Greedy» algorithm

0. Preparatory step:
   - let $W_i = 0$ – the current number of haystacks in trucks, $i = \overline{1,n}$;
   - set $I = \{1,2,...,n\}$ is the set of indices of unloaded trucks;
   - let $ind_{car}k = 0$ – the index of the truck which the $k$-th tractor arrived to, $k = \overline{1,K}$;
   - set $J = \{1,2,...,m\}$ is the set of indices of unharvested haystalks;
   - let $SUM = 0$ – the total distance traveled by the tractors during transportation all the haystacks from the field in the trucks.
1. Put $k = 1$ – the number of the current tractor.
2. Put $i = ind_{car}k$ – the index of the truck which the $k$-th tractor arrived to.
3. Calculate $c_{ij}^r = \min_{j\in J} c_{ij}$ – the distance to the nearest haystack on the field;
   - put $x_{ij}^r = 1$ – fill the array of tractor routes;
   - put $SUM = SUM + c_{ij}^r$ – increase the total distance traveled by tractor.
4. Calculate $c_{i^*j^*} = \min_{i\in I} c_{i^*j^*}$ – the distance to the nearest tractor on the field;
   - put $y_{i^*j^*} = 1$ – fill the array of tractor routes;
   - put $SUM = SUM + c_{i^*j^*}$ – increase the total distance traveled by tractor.
5. Put $W_{i^*} = W_{i^*} + 1$ – increase the current number of haystacks in the truck;
   - if $W_{i^*} = \overline{W}_{i^*}$, the truck is fully loaded, put $I = I\setminus\{i^*\}$ – exclude the truck index from the set of indices of unloaded trucks.
6. Put $ind_{car}k = i^*$ – remember the index of the truck which the $k$-th tractor arrived to.
7. Put $J = J\setminus\{j^*\}$ – exclude the haystack index from the set of indices of unharvested haystalks;
   - if $J = \emptyset$, there are no haystacks on the field, proceed to step 9.
8. If $k < K$, put $k = k + 1$ – go to the next tractor, otherwise, put $k = 1$ – go to the first tractor; proceed to step 2.
9. Put $k = 1$ – the number of the current tractor.
10. Put $i = ind_{car}k$ – the index of the truck which the $k$-th tractor arrived to.
11. Put $x_{i(m+1)}^r = 1$ – fill the array of tractor routes.
12. Put $SUM = SUM + c_{i(m+1)}$ – increase the total distance traveled by tractor.
13. If \( k < K \), put \( k = k + 1 \) – go to the next tractor and proceed to step 10, otherwise, the end of the algorithm.

Note that it is possible to separate finding minimum distances "truck-haystack" and "haystack-truck" in this algorithm. I.e. after step 3 we can go to the next tractor, and start step 4 after the execution of this action with all the tractors.

Note that finding the nearest haystack is a sequential process for tractors, and the problem of its paralleling is an isolated issue. But finding the nearest truck can be implemented concurrently for all tractors because generally speaking, we can come to each truck as many times as we want. Thus, step 3 is a sequential process and step 4 is a concurrent process.

5. Ant Colony Optimization algorithm

Let us consider the developed Ant Colony Optimization algorithm [13-15], which is based on the idea of "greedy" algorithm. As before, the basic idea of the algorithm is to find a mini-route "truck-haystack-truck" for each tractor. But now the process of finding the next haystack and truck will be implemented taking into account the probabilities of choosing a particular path.

An important step in the Ant Colony Optimization algorithm is the probabilistic choice (or transition): it is necessary to move to one of several points on the basis of the available transition probabilities (figure 3).

\[ \text{Figure 3. Possible variants of probabilistic transition.} \]

The idea of the program implementation of probabilistic transition is based on the fact that all possible transitions are exhaustive events. Thus, the sum of all transition probabilities is equal to one. Suppose that each probability value specifies the length of the segment, and all probability values are located on the unit segment, completely filling it (figure 4). To make a probabilistic choice, it is necessary to generate a random real number from 0 to 1, determine which segment it appears in, and make the appropriate transition (figure 5).

\[ \text{Figure 4. The principle of probabilistic choice.} \]
Algorithm 2. Probability transition algorithm

0. Preparatory step:
   - set \( M \) and \( p_l \) – the set of objects and corresponding probabilities, \( \sum_i p_i = 1 \);
   - generate \( \text{rand} \in [0, 1] \) – the threshold of the probabilistic choice;
   - set \( \text{sum}_p = 0 \) – current sum of probabilities.
1. Put \( i = 1 \) – the sequence number of the object in the set \( M \).
2. Put \( \text{sum}_p = \text{sum}_p + p_l \) – increase the total probability.
3. If \( \text{sum}_p < \text{rand} \), put \( i = i + 1 \) and proceed to step 2. Otherwise, the end of the algorithm.

The result of the algorithm is the index \( i \) – the sequence number of the object in the set \( M \).

In the developed Ant Colony Optimization algorithm, we use not one large colony of ants, but several small colonies corresponding to each tractor. In each group, we number the agents-ants in order and consider the ants under the same number from all the colonies. Each of these ants passes the mini-route "truck-haystack-truck" alternately, making the choice of the nearest object with the help of probabilistic choice.

Note that it is necessary to use two pheromone matrices \( \tau_{x_{ij}}, i = 0, n, j = 1, m + 1 \) and \( \tau_{y_{ij}}, i = 1, n, j = 1, m \), since the pheromone deposited on the path "\( i \)-th truck \( \rightarrow \) \( j \)-th haystack" doesn’t affect the path "\( j \)-th haystack \( \rightarrow \) \( i \)-th truck".

The amount of pheromone is updated at the end of each iteration, i.e. after constructing the route by the ant-agents with the same number from each colony. Denote by \( \text{amount}_\text{ant} \) the amount of ants in the colonies, then only one iteration gives \( K \times \text{amount}_\text{ant} \) routes or \( \text{amount}_\text{ant} \) sets of routes.

Algorithm 3. Ant Colony Optimization algorithm

0. Preparatory step:
   - set \( \text{amount}_\text{it} \) – the number of iterations of the algorithm;
   - set \( \text{amount}_\text{ant} \) – the number of ants in colonies;
   - set \( \tau_{x_{ij}} = 1 \) – the amount of pheromone on the path between the \( i \)-th truck and the \( j \)-th haystack, \( i = 0, n, j = 1, m + 1 \);
   - set \( \tau_{y_{ij}} = 1 \) – the amount of pheromone on the path between the \( j \)-th haystack and the \( i \)-th truck, \( i = 1, n, j = 1, m \);
   - set \( d_{ij} = 1/c_{ij} \) – the desirability of path between \( i \)-th truck and \( j \)-th haystack, \( i = 0, n, j = 1, m \);
   - set \( \alpha \) – the coefficient of pheromone influence on the probability of path selection;
   - set \( \beta \) – the coefficient of the desirability influence on the probability of path selection;
   - set \( \rho \) – the pheromone evaporation rate;
set $SUM' = \infty$ – the “record”, the resulting total distance traveled by the tractors during transportation all the haystacks from the field in the trucks.
1. Put it = 1 – the number of the current iteration of the algorithm.
2. Put ant = 1 – the sequence number of ants in each colony.
3. Specify:
   - $W_i = 0$ – the current number of haystacks in trucks, $i \in 1, n$;
   - $I = \{i: W_i < \overline{W}_i, i = 1, n\}$ – the set of indices of unloaded trucks;
   - $\text{ind\_car}_k = 0$ – the index of the truck which the $k$-th tractor arrived to, $k = 1, K$;
   - $J = \{1, 2, ..., m\}$ – the set of indices of unharvested haystacks;
   - $SUM = 0$ – the total distance traveled by the tractors during transportation all the haystacks from the field in the trucks.
4. Put $k = 1$ – the number of the current tractor.
5. Put $i = \text{ind\_car}_k$ – the index of the truck which the $k$-th tractor arrived to.
6. Find $p_{ij} = \frac{\tau_{x_i}^{a_d} x_i^{d_i}}{\sum_{i,j=1}^{m} \tau_{x_j}^{a_d} x_j^{d_i}}$ – the probability of path selection between the $i$-th truck and $j$-th haystack for all $j \in J$.
7. Apply algorithm 2, where set $M = J$ (the set of indices of unharvested haystacks), get the index $j^*$;
   - put $x_{jk}^* = 1$ – fill the array of tractor routes;
   - put $SUM = SUM + c_{ij^*}$ – increase the total distance traveled by tractors.
8. Find $p_{i^*j} = \frac{\tau_{y_i^*}^{a_d} y_i^*^{d_i}}{\sum_{i,j=1}^{m} \tau_{y_j}^{a_d} y_j^{d_i}}$ the probability of path selection between the $i$-th truck and $j$-th haystack for all $i \in I$.
9. Apply algorithm 2, where set $M = I$ (the set of indices of unloaded trucks), get the index $i^*$;
   - put $y_{i^*k} = 1$ – fill the array of tractor routes;
   - put $SUM = SUM + c_{i^*j}$ – increase the total distance traveled by tractors.
10. Put $W_i^* = W_i^* + 1$ – increase the current number of haystacks in the truck; if $W_i^* = \overline{W}_i$, the truck is fully loaded, put $I = I \{i^*\}$ – exclude the truck index from the set of indices of unloaded trucks.
11. Put $\text{ind\_car}_k = i^*$ – remember the index of the truck which the $k$-th tractor arrived to.
12. Put $J = J \{j^*\}$ – exclude the haystack index from the set of indices of unharvested haystacks; if $J = \emptyset$, there are no haystacks on the field, proceed to step 14.
13. If $k < K$, put $k = k + 1$ – go to the next tractor, otherwise, put $k = 1$ – go to the first tractor; proceed to step 5.
14. Put $k = 1$ – the number of the current tractor.
15. Put $i = \text{ind\_car}_k$ – the index of the truck which the $k$-th tractor arrived to.
16. Put $x_{i(m+1)}^k = 1$ – fill the array of tractor routes.
17. Put $SUM = SUM + c_{i(m+1)}$ – increase the total distance traveled by tractors.
18. If $k < K$, put $k = k + 1$ – go to the next tractor and proceed to step 15.
19. If $SUM < SUM^*$, put $SUM' = SUM$ – remember the “record”;
   - put $x_{ij}^k = x_{ij}^k$, $y_{ij}^k = y_{ij}^k$ – remember tractor routes, $i = 0, m, j = 1, m + 1, k = 1, K$.
20. Put $k = 1$ – the number of the current tractor.
21. If $x_{ij}^k = 1$, $i = 0, m, j = 1, m + 1$ – the route of the $k$-th tractor passed between the $i$-th truck and the $j$-th haystack, then put
   $$\tau_{x_{ij}} = \tau_{x_{ij}} * (1 - \rho) + \frac{1}{\epsilon_{ij}},$$
   otherwise,
   $$\tau_{x_{ij}} = \tau_{x_{ij}} * (1 - \rho).$$
22. If \( y_{ji}^k = 1, \ i = 1, n, j = 1, m \) – the route of the \( k \)-th tractor passed between the \( j \)-th haystack and the \( i \)-th truck, then put
\[
\tau_{-yi} = \tau_{-yi} * (1 - \rho) + \frac{1}{cij},
\]
otherwise,
\[
\tau_{-yi} = \tau_{-yi} * (1 - \rho).
\]
23. If \( k < K \), put \( k = k + 1 \) – go to the next tractor; proceed to step 21.
24. If \( ant < amount \_ant \), put \( ant = ant + 1 \) – increase the sequence number of ants in each colony, proceed to step 3.
25. If \( it < amount \_it \), put \( it = it + 1 \) – increase the number of the current iteration of the algorithm, proceed to step 2, otherwise, the end of the algorithm.

The results of the algorithm are \( SUM^*, x^*, y^* \).

6. Computational experiment

An application to test and analyze the algorithms proposed in this paper has been developed in the Microsoft Visual Studio 2017 using C#.

The resources of the application allow setting the dimension of the field, the number of tractors, trucks and haystacks. In accordance with the entered parameters, the program randomly generates the position of haystacks on the field, as well as trucks and their loading capacity. If it is necessary to specify the fixed location of objects on the field, the program allows setting the coordinates of all objects manually. After completion of work, the detailed information about the distances traveled by each tractor at each iteration, the resulting routes of tractors, as well as the total distance traveled by all tractors displays on the form. The application provides the ability of graphically displaying the final route of each tractor separately.

Both algorithms – “greedy” algorithm and Ant Colony Optimization algorithm are realized in the application.

On the basis of this application, a computational experiment was conducted.

The aims of the computational experiment:
1. To compare the results of the “greedy” algorithm and Ant Colony Optimization algorithm;
2. To determine the optimal amount of ants in the colonies;
3. To determine the optimal number of iterations of the algorithm.

The input data of the experiment: the number of haystacks; the number of trucks; the number of tractors; the matrix of distances filled with random numbers from the interval \([1, 100]\).

Fixed parameters of the algorithm in the experiment: \( \alpha = 1, \beta = 1, \rho = 0.2 \).

For each variant of the initial data 500 cubical matrices were set and the arithmetical mean of the values of the objective functions was calculated.

6.1. Comparison of algorithms

Table 1 shows the results of the “greedy” algorithm and Ant Colony Optimization algorithm for problems of different dimensions with a fixed number of ants in the colonies equal to 20, as well as a fixed number of tractors equal to 3.
Table 1. The results of the "greedy" algorithm and Ant Colony Optimization algorithm with 20 ants and 3 tractors.

| Haystacks × trucks | "Greedy" algorithm | ACO algorithm (20 iterations) | ACO algorithm (30 iterations) | ACO algorithm (40 iterations) | ACO algorithm (50 iterations) |
|--------------------|--------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|
| 25 × 5             | 1714               | 1680                          | 1589                          | 1559                          | 1546                          |
| 25 × 7             | 1562               | 1572                          | 1481                          | 1438                          | 1405                          |
| 25 × 10            | 1428               | 1505                          | 1388                          | 1338                          | 1320                          |
| 50 × 5             | 3227               | 2965                          | 2862                          | 2762                          | 2721                          |
| 50 × 7             | 2849               | 2819                          | 2585                          | 2506                          | 2448                          |
| 50 × 10            | 2507               | 2672                          | 2357                          | 2220                          | 2179                          |
| 100 × 5            | 6305               | 5566                          | 5275                          | 5244                          | 5210                          |
| 100 × 7            | 5575               | 5281                          | 4972                          | 4778                          | 4555                          |
| 100 × 10           | 4888               | 5028                          | 4500                          | 4227                          | 3958                          |

Table 2 shows the similar results of the "greedy" algorithm and Ant Colony Optimization algorithm with a fixed number of tractors equal to 5.

Table 2. The results of the "greedy" algorithm and Ant Colony Optimization algorithm with 20 ants and 5 tractors.

| Haystacks × trucks | "Greedy" algorithm | ACO algorithm (20 iterations) | ACO algorithm (30 iterations) | ACO algorithm (40 iterations) | ACO algorithm (50 iterations) |
|--------------------|--------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|
| 25 × 5             | 1993               | 1922                          | 1822                          | 1807                          | 1765                          |
| 25 × 7             | 1874               | 1829                          | 1715                          | 1675                          | 1652                          |
| 25 × 10            | 1754               | 1742                          | 1632                          | 1596                          | 1578                          |
| 50 × 5             | 3419               | 3186                          | 3055                          | 2951                          | 2949                          |
| 50 × 7             | 3140               | 3031                          | 2812                          | 2733                          | 2677                          |
| 50 × 10            | 2791               | 2875                          | 2595                          | 2476                          | 2415                          |
| 100 × 5            | 6390               | 5791                          | 5510                          | 5381                          | 5295                          |
| 100 × 7            | 5761               | 5441                          | 5128                          | 4876                          | 4822                          |
| 100 × 10           | 5048               | 5184                          | 4679                          | 4397                          | 4222                          |

Summary:
1. The results of the "greedy" algorithm and Ant Colony Optimization algorithm approximately equally with a small of the number of iterations (less than 20).
2. ACO algorithm leads to improving the result with increase of the number of iterations. The results of the ACO algorithm are 0.8-0.85 times lower than the results of the "greedy" algorithm (40-50 iterations). The best results of the ACO algorithm with a number of iterations equal to 50.
3. Increase of the number of trucks on the field leads to decreasing the total distance traveled by all tractors.

6.2. Determining the optimal amount of ants in colonies

Table 3 shows the results of the ACO algorithm with different amount of ants in colonies. The number of iterations is fixed: amount_it = 50. The number of tractors equal to 3.

Table 3. The results of the Ant Colony Optimization algorithm with 50 iterations and 3 tractors.

| Haystacks × trucks | 20 ants | 30 ants | 40 ants | 50 ants |
|--------------------|---------|---------|---------|---------|
| 25 × 5             | 1540    | 1500    | 1489    | 1492    |
| 25 × 7             | 1407    | 1381    | 1360    | 1352    |
| 25 × 10            | 1318    | 1283    | 1258    | 1243    |
| 50 × 5             | 2714    | 2681    | 2653    | 2598    |
| 50 × 7             | 2461    | 2393    | 2373    | 2344    |
| 50 × 10            | 2180    | 2147    | 2118    | 2102    |
| 100 × 5            | 5173    | 5122    | 5026    | 4963    |
| 100 × 7            | 4677    | 4658    | 4564    | 4552    |
| 100 × 10           | 4091    | 4049    | 4025    | 4024    |
Table 4 shows the similar results of the Ant Colony Optimization algorithm with a fixed number of tractors equal to 5.

| Haystacks × trucks | 20 ants | 30 ants | 40 ants | 50 ants |
|---------------------|---------|---------|---------|---------|
| 25 × 5              | 1761    | 1733    | 1722    | 1708    |
| 25 × 7              | 1648    | 1619    | 1601    | 1598    |
| 25 × 10             | 1563    | 1539    | 1528    | 1517    |
| 50 × 5              | 2936    | 2868    | 2862    | 2826    |
| 50 × 7              | 2665    | 2603    | 2599    | 2548    |
| 50 × 10             | 2432    | 2398    | 2388    | 2348    |
| 100 × 5             | 5327    | 5257    | 5160    | 5131    |
| 100 × 7             | 4806    | 4754    | 4701    | 4691    |
| 100 × 10            | 4238    | 4230    | 4205    | 4207    |

Summary:
1. An increase in the number of ants in the colonies leads to increasing the efficiency of the algorithm.
2. Beginning with a number of ants in the colonies equal to 50, an increase the number of ants does not significantly improve the values of the objective function. The best results of the ACO algorithm with a number of ants equal to 50.

7. Conclusion
The Vehicle Routing Problem has many different formulations. In this paper, we propose two statements of its formulation: Interleaved routing problem and the Interleaved routing problem with a single point of collection.

The main results:
1. Development of a mathematical model for two statements in the form of a boolean mathematical programming problem.
2. Development of a mathematical model for the meaningful problem statement.
3. Development of «greedy» algorithm for solving.
4. Development of ACO algorithm for solving.
5. Analysis of algorithms based on the computational experiment.
6. The optimal parameters of ACO algorithm for the Interleaved routing problem are obtained.

References
[1] Korbut A A and Finkel'shtein Yu Yu 1969 *Discrete programming* (Moscow: NAUKA) (in Russian)
[2] Assad A A and Golden B L 1995 Arc routing methods and applications *Network Routing. Handbooks in OR/MS* vol 8 ed M O Ball, T L Magnanti, C L Monma and G L Nemhauser (Amsterdam: Elsevier) pp 375-483
[3] Amberg A, Domschke W and Voss S 2000 Multiple center capacitated arc routing problems: A tabu search algorithm using capacitated trees *Europ. J. of Operat. Res. 124* pp 360-376
[4] Papadimitriou C H and Steiglitz K 1998 *Combinatorial Optimization. Algorithms and Complexity* (New York: Dover) p 528
[5] Minoux M 1990 *Mathematical Programming: Theory and Methods* (Moscow: Nauka) (in Russian) p 488
[6] Benavent E, Campos V, Corberan A and Mota E 1990 The CARP – a heuristic algorithm *Questiio 14* (1-3) pp 107-122
[7] Chapleau L, Ferland J A, Lapalme G and Rousseau J M 1984 A parallel insert method for the capacitated arc routing problem *Operat. Res. Letters 3* pp 95-99
[8] Medvedeva O A, Medvedev S N and Chernyshova G D 2013 Bulletin of Voronezh State University. Series: Systems analysis and information technologies 2 pp 38-41 (in Russian)
[9] Medvedeva O A 2013 Bulletin of St. Petersburg University: Applied Mathematics. Computer Science. Control Processes 10(1) pp 85-94 (in Russian)
[10] Medvedeva O A and Medvedev S N 2018 J. Phys.: Conf. Ser. 973 012039 URL: iopscience.iop.org/issue/1742-6596/973/1
[11] Pearn W L 1991 Augment-insert algorithms for the capacitated arc routing problem Comp. and Operat. Res. 18 pp 189-198
[12] Tregubov A G and Medvedev S N 2015 Bulletin of Voronezh State University. Series: Systems analysis and information technologies 4 pp 31-37 (in Russian)
[13] Dorigo M, Maniezzo V and Colorni A 1996 Ant System: Optimization by a Colony of Cooperating Agents IEEE Transactions on Systems (Man and Cybernetics Part B) 26 (1) pp 29-41
[14] Karpenko A P 2014 Modern search engine optimization algorithms. Nature-inspired optimization algorithms (Moscow: Publishing of Bauman Moscow State Technical University) (in Russ.)
[15] Shtovba S D and Ryduy O M 2004 Ant optimization algorithms Bulletin of Vinnitsa State Technical University 4 pp 62-69 (in Ukrainian)