DOI: http://doi.org/10.25728/cs.2024.1.3

COMBINED HIERARCHICAL CROSSOVER IN A GENETIC ALGORITHM FOR LAST-MILE DELIVERY: EFFICIENCY ANALYSIS

V. A. Sosedov

Trapeznikov Institute of Control Sciences, Russian Academy of Sciences, Moscow, Russia

✉ vladyslav.sosedov@gmail.com

Abstract. This paper considers routing for a group of unmanned aerial vehicles within a promising last-mile delivery system. The routing problem is reduced to the bi-criteria single-depot multiple traveling salesman problem and formalized using a directed graph. Being NP-hard, this problem cannot be efficiently solved by standard exact optimization methods. Therefore, heuristic algorithms should be applied to obtain good approximate solutions in a short time. The problem is solved using NSGA-II, the widespread elitist non-dominated sorting genetic algorithm that demonstrates good results in multicriteria optimization. Some chromosome representation and crossing and mutation operators are implemented in the algorithm. A simulation software tool is presented to investigate the influence of the crossing operators used on the convergence speed of the algorithm. Finally, several genetic crossing operators (Partially-Mapped Crossover, Order Crossover, Cycle Crossover, and Combined Hierarchical Crossover) are compared in terms of efficiency.

Keywords: last-mile delivery, multiple traveling salesman problem, multicriteria optimization, genetic algorithm, crossover.

INTRODUCTION

Fast and cost-efficient commercial delivery of small-sized goods ordered online is a difficult and complex logistics problem that many companies are currently trying to solve. One of the most promising technologies at the final stage of the supply chain, the so-called last-mile delivery, is the use of unmanned aerial vehicles (UAVs) within the system for delivering goods from distribution centers to customers [1]. The UAVs-based delivery of small-sized goods in an urban environment is more beneficial than classical courier delivery from an economic point of view: UAVs are not limited to a static set of roads and can flexibly move in three dimensions, which significantly reduces the time and, consequently, the cost of delivery [2]. In addition, automation of such a system significantly decreases the number of employees engaged in order transportation, also reducing its total cost.

Although last-mile delivery using UAVs suggests a more efficient alternative to courier delivery, its implementation within a real system is limited by the technical capabilities of currently available hardware and software tools. Note also that real delivery systems involve a large number of customers to be served simultaneously.

As was shown in the survey [3], for a group of UAVs, the simplest centralized routing problem is reduced to the NP-hard *Multiple Traveling Salesman Problem* (MTSP) provided that each UAV can serve one or several customers per flight. In this paper, to decrease the dimension of the optimization problem, we divide the delivery system into service zones with a single distribution center inside and separately solve the resulting *Single-Depot Multiple Traveling Salesman Problems* (SD-MTSPs). In Section 1, we show the necessity of simultaneous optimization of two conflicting objective functions (criteria) and present the formal problem statement.

Standard exact optimization methods need an exponentially growing time for computations as the number of optimized parameters increases; in a real system, heuristic algorithms should be therefore applied to find approximate solutions. Currently, the main approach to solving this class of problems is to

use various metaheuristic algorithms [4]. We choose the elitist *Non-dominated Sorting Genetic Algorithm II* (NSGA-II). See Section 2 for a detailed description of this algorithm, as well as the algorithms of genetic selection, crossing, and mutation.

The efficiency of a genetic algorithm mostly depends on the local search capabilities of its crossing operator. In Section 3, we compare the results produced by the algorithm with Combined Hierarchical Crossover and several standard crossing operators (Partially-Mapped Crossover, Order Crossover, and Cycle Crossover) in terms of efficiency.

1. PROBLEM STATEMENT

The single-depot multiple traveling salesman problem has the following general statement. Consider a set of $n > 1$ cities (points) and m identical traveling salesmen. Each salesman departs from the same starting point (point 0), makes a tour, and returns to this point. Each point, except the starting one, must be visited exactly once. The problem is solved by minimizing some global objective function F [5].

In the standard statement, it is the total length of routes of all salesmen. However, minimization of such an objective function without additional route constraints yields solutions with strongly imbalanced routes (Fig. 1).

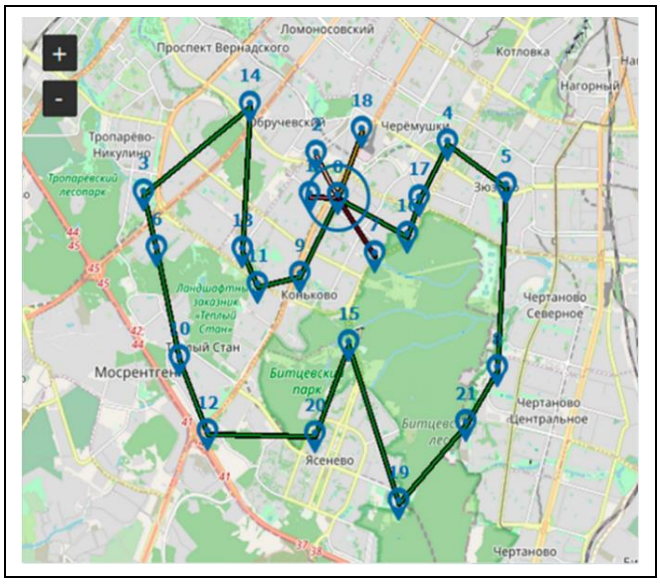

Fig. 1. A typical solution of the standard problem statement.

From a practical point of view, several salesmen are intended to reduce the time of service to all customers. Therefore, salesmen routes are often balanced by their length using different variations of the minimax problem statement, where the objective function can be, e.g., the length of the longest route or the so-called degree of imbalance (the difference between the lengths of the longest and shortest routes). In turn, this approach may result in the irrational routes of salesmen in the sense of minimum length (Fig. 2), and the time of serving all customers will also be nonoptimal.

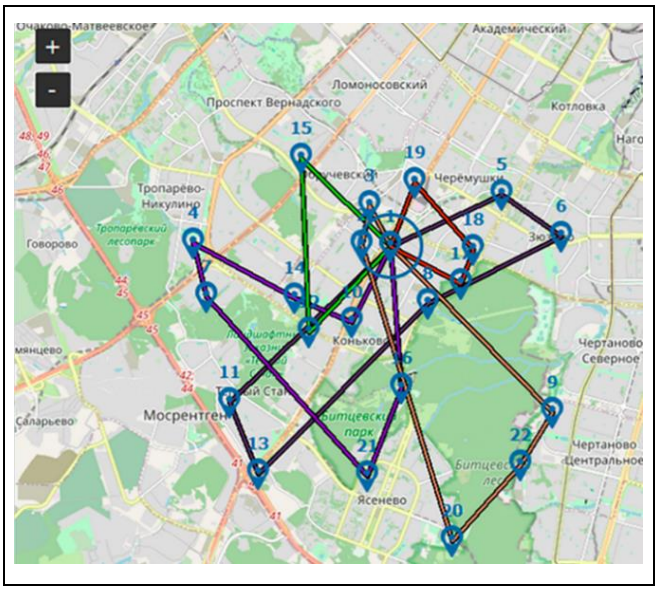

Fig. 2. A typical solution of the minimax problem statement.

Thus, minimizing the total length and balancing the routes are two conflicting optimization problems that need to be considered together. The multicriteria approach to solving the multiple traveling salesman problem has proven to be efficient [6–8] since it smoothens the disadvantages of conflicting objective functions without introducing additional solution constraints. Moreover, the presence of several objective functions in the problem naturally leads to a set of Pareto-optimal solutions instead of a single optimal solution, and the best solution (in some sense) can be flexibly selected under additional information available about the problem.

The bi-criteria single-depot multiple traveling salesman problem can be formalized [6] using a directed graph $G = (V, A)$, where V and A denote the sets of vertices and arcs, respectively. A symmetric weight matrix (distance matrix) $C = (c_{ij})$, $(i, j) \in A$, is associated with this graph; it can be defined considering the constraints imposed by the environment of the real delivery system. In this paper, we define the values c_{ij} in the simplest way, i.e., as the Euclidean distances between points. Let x_{ijk} be a binary variable taking value 1 if salesman k passes through the arc (i, j) and value 0 otherwise. Also, let u_i be the number of points visited by the salesman on the route from the starting point to point i.

With the notations introduced above, the formal problem statement is as follows:

$$F = (f_1,\ f_2) \to \min, \tag{1}$$

$$f_1 = \sum_{k=1}^{m}\sum_{i=0}^{n}\sum_{j=0}^{n} c_{ij} x_{ijk}, \tag{2}$$

$$f_2 = \max_{1\le k\le m}\sum_{i=0}^{n}\sum_{j=0}^{n} c_{ij} x_{ijk} - \min_{1\le k\le m}\sum_{i=0}^{n}\sum_{j=0}^{n} c_{ij} x_{ijk}, \tag{3}$$

where $x_{ijk} \in \{0, 1\}\ \ \forall (i, j) \in A,\ \ k = 1, \ldots, m,$

$$\sum_{j=1}^{n} x_{0jk} = 1,\ \ k = 1, \ldots, m, \tag{4}$$

$$\sum_{i=1}^{n} x_{i0k} = 1,\ \ k = 1, \ldots, m, \tag{5}$$

$$\sum_{k=1}^{m}\sum_{i=1}^{n} x_{ijk} = 1,\ \ j = 1 \ldots n,\ \ i \ne j, \tag{6}$$

$$\sum_{k=1}^{m}\sum_{j=1}^{n} x_{ijk} = 1,\ \ i = 1 \ldots n,\ \ i \ne j, \tag{7}$$

$$\sum_{i=1}^{n} x_{ijk} = \sum_{i=1}^{n} x_{jik},\ \ j = 1, \ldots, n,\ k = 1, \ldots, m,\ \ i \ne j, \tag{8}$$

$$u_i - u_j + (n - m)\cdot \sum_{k=1}^{m} x_{ijk} \le n - m - 1, \quad 2 \le i \ne j \le n. \tag{9}$$

The expressions (2) and (3) define two objective functions to be minimized jointly (1), i.e., the total route lengths of all salesmen and the difference between the lengths of the longest and shortest routes, respectively. Conditions (4) and (5) ensure that exactly *m* salesmen will leave the starting point and return to it. Due to conditions (6)–(8), each point (except the starting one) will be visited exactly once. The constraint (9) is a classical form of the *subtour elimination constraint* (SEC), which ensures that the solution will not contain any degenerate tours.

2. ALGORITHM DESCRIPTION

2.1. Chromosome Representation

The key to obtaining good solutions of an optimization problem using a genetic algorithm is a correct representation of an individual or chromosome that should accurately determine the solution of the problem and allow genetic operators to efficiently generate the fittest solutions as the iterative evolutionary process continues. The paper [9] proposed a two-part chromosome representation method that reduces the redundancy of the solution space (compared to the one-string and two-string chromosome representation methods) and, consequently, improves the efficiency of the algorithm. This method was used for solving the multiple traveling salesman problem.

The chromosome consists of two parts (Fig. 3): the first part contains (n – 1) numbers characterizing the order of visiting points in the routes of salesmen, and the second part is composed of (m – 1) break points that divide the first part into m groups. Each group describes the tour of one salesman (Fig. 4).

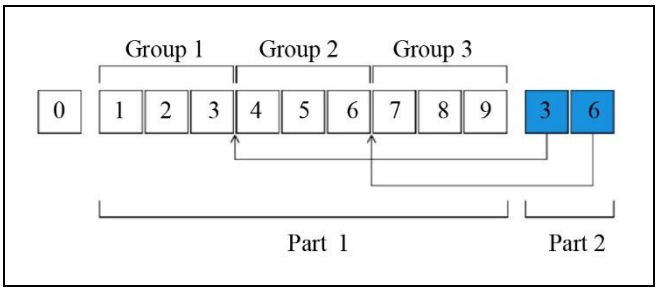

Fig. 3. The chromosome representation.

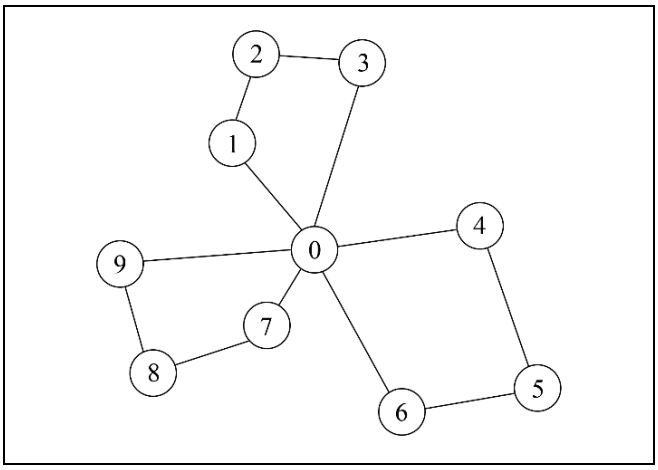

Fig. 4. The solution visualized.

2.2. Crossing Operators

As is known, the crossing operator (crossover) used in a genetic algorithm has the greatest influence on the local search capabilities and efficiency of the algorithm. Such crossing operators as *Partially-Mapped Crossover* (PMX), *Order Crossover* (OX), and *Cycle Crossover* (CX) are most widespread in combinatorial optimization problems. The principles of their operation were described in many works; for example, see the books [10, 11]. The algorithms of the standard crossing operators used below are adapted to conform to the two-part chromosome representation method. The modified algorithms of the crossing operators are described in detail below.

The modified PMX algorithm includes several steps as follows (Fig. 5).

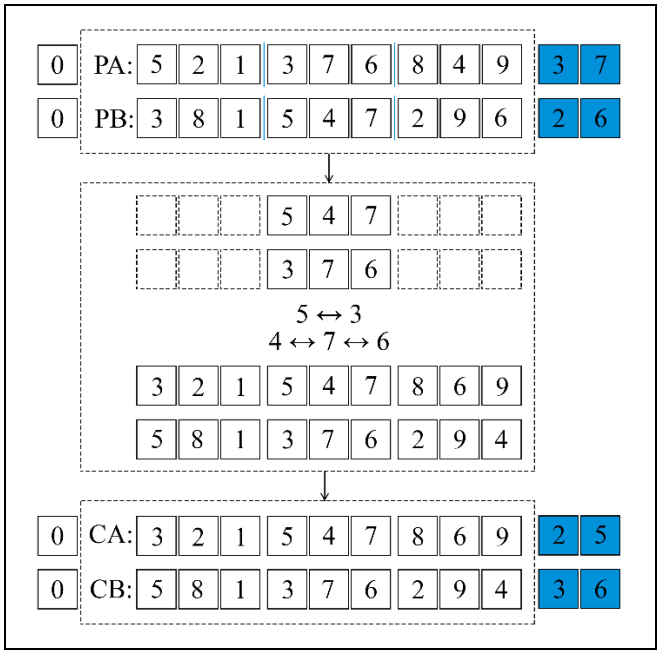

Fig. 5. The modified PMX algorithm.

Step 1. A pair of parents is supplied to the input of the algorithm. Two cutting points are randomly and uni

formly selected from the first parts of the parent chromosomes *PA* and *PB*. The gene sequences between the two cutting points are called map sections.

Step 2. The map sections are swapped and copied in accordance with the gene positions into the first parts of the child chromosomes *CA* and *CB*. The other genes are considered undefined.

Step 3. The gene relationship between the two map sections is determined.

Step 4. The genes undefined in *CA* and *CB* are filled by copying the genes of the corresponding parent. If a gene present in a child is replaced according to its map.

Step 5. The second parts of the child chromosomes are generated randomly.

The modified OX algorithm includes several steps as follows (Fig. 6).

Step 1. A pair of parents is supplied to the input of the algorithm. Two cutting points are randomly selected from the first parts of the parent *PA* and *PB*.

Step 2. The gene sequences are swapped and copied in accordance with the gene positions into the first parts of the child chromosomes *CA* and *CB*. The other genes are considered undefined.

Step 3. The left-to-right gene orders are determined in *PA* and *PB*.

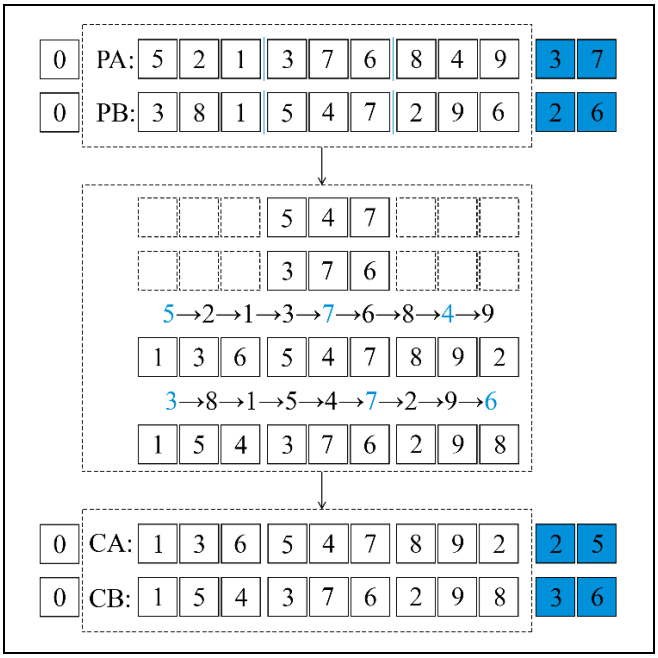

Fig. 6. The modified OX algorithm.

Step 4. The genes undefined in *CA* and *CB* are filled in accordance with the gene order of the corresponding parents, starting from the second cutting point. A gene present in a child is skipped.

Step 5. The second parts of the child chromosomes are generated randomly.

The modified CX algorithm includes several steps as follows (Fig. 7).

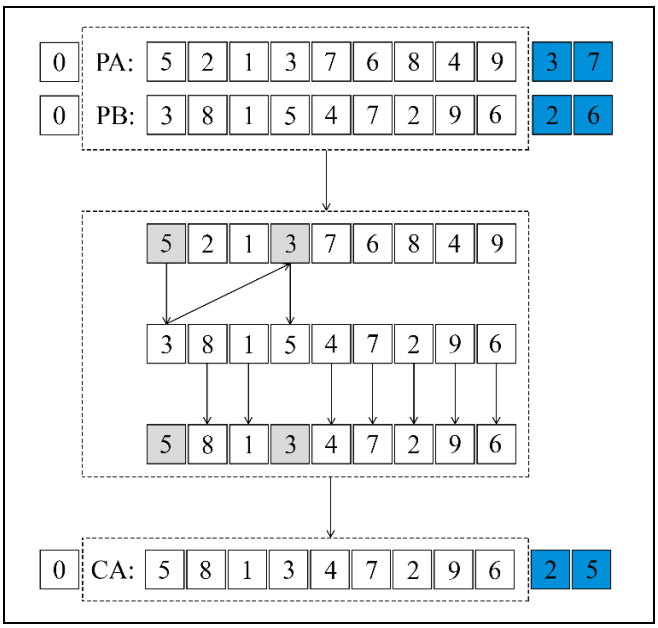

Fig. 7. The modified CX algorithm.

Step 1. A pair of parents is supplied to the input of the algorithm. The starting position of the cycle is ran-

domly selected from the first part of the parent chromosome *PA*.

Step 2. The gene occupying the starting position in *PA* is written into the first part of the child chromosome in accordance with its position. The gene occupying the same position in the first part of the parent chromosome *PB* cannot be written into the child chromosome at the same position, so it is written into the child chromosome in accordance with its position in *PA*. The cycle continues until the gene occupying the starting position in *PA* is found in *PB*.

Step 3. After the cycle is completed, the remaining undefined genes are copied from *PB* in accordance with their positions.

Step 4. The second part of the child chromosome is generated randomly.

Step 5. To form the second child, the parents are swapped, and Steps 1–5 of the algorithm are repeated.

The crossing operators described above are universal for combinatorial optimization problems and generate the fittest solutions during the iterative evolutionary process of the genetic algorithm. In applications, however, their local search capabilities turn out to be relatively small.

The authors [12] introduced *Combined Hierarchical Crossover* (Combined HGA, cHGA) to solve the multiple traveling salesman problem. This crossing operator provides high local search efficiency by considering the distances between points in the main child-formation process and good population diversity due to different approaches used to form two children.

The basic cHGA algorithm includes several steps as follows.

Step 1. A pair of chromosomes *PA* and *PB* is supplied to the input to the algorithm. The gene PA_k is randomly selected from *PA* and written into the starting position of the child chromosome.

Step 2. Depending on the given search direction, two subsequent genes (PA_{k+1} and PB_{k+1}) or two previous genes (PA_{k-1} and PB_{k-1}) are selected from the parent chromosomes.

Step 3. The gene PA_k is eliminated from the parent chromosomes.

Step 4. The two distances c_{ij} between the genes selected at Step 2 and the gene PA_k are compared with each other. The gene with the smaller distance is written into the child and becomes the new gene PA_k . The algorithm continues from Step 2 while the length of the parent chromosome *PA* exceeds 1.

According to [12], the children produced by this algorithm have different values of the two objective functions depending on the representation of the input data *PA* and *PB*. Two input data generation methods were proposed to obtain a well-balanced set of solutions.

The first child is formed (Fig. 8) by selecting the first parts of the parent chromosomes as the input data for the main algorithm. The algorithm produces the first part of the child chromosome, and its second part is randomly selected from the second parts of the parent chromosomes. Most of the resulting children will have fairly balanced salesman routes but will not contribute to reducing the total length of all routes.

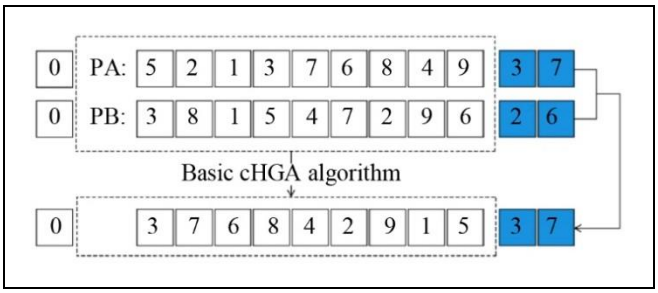

Fig. 8. The formation algorithm of the first child.

The second child (Fig. 9) is formed using the preliminary decoding of parent chromosomes into a single-string representation as follows: the genes of the starting point 0 are added in the beginning of the first part of the chromosome and in the positions defined by the break points; the second part of the chromosome is eliminated. The decoded parent chromosomes are supplied to the input of the main algorithm.

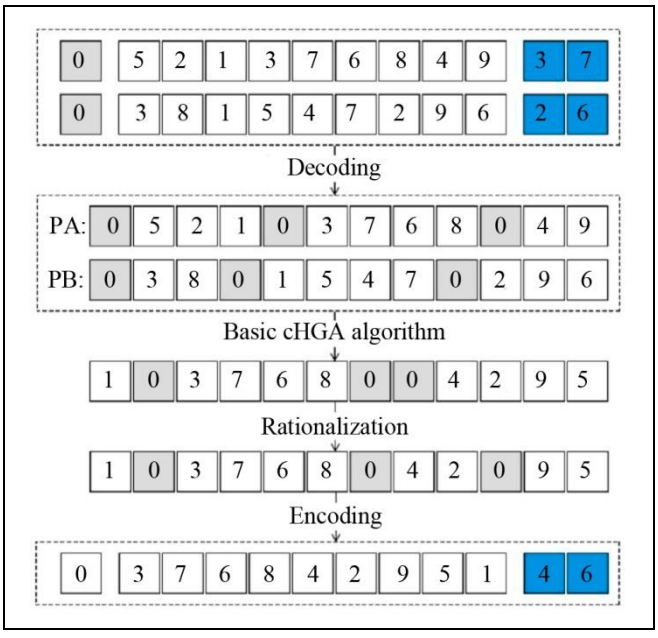

Fig. 9. The formation algorithm of the second child.

The algorithm result is subjected to a rationalization procedure: the “null” routes are eliminated (i.e., the right adjacent genes of the starting point 0 are moved to a randomly selected position to the right

along the chromosome). Finally, the result is encoded into the original two-part chromosome representation. Most children produced in this way will contribute to reducing the total length of all routes but will not contribute to decreasing the degree of route imbalance.

2.3. Selection and Mutation

The selection operator chooses individuals with the highest value of the fitness function to form the set of individuals for crossover. The proposed algorithm uses binary tournament selection, the standard operator for the non-dominant sorting genetic algorithm.

The search area of the optimal solution may be narrowed when hitting a local minimum. Mutation operators prevent this situation by editing child genes. The proposed algorithm involves four different mutation operators [10, 13]: *Insertion Mutation*, *Exchange Mutation*, *Inversion Mutation*, and *Scramble Mutation*. All mutation operators have an equal probability of being called.

2.4. NSGA-II

We solve the problem under consideration using NSGA-II, the elitist Non-Dominated Sorting Genetic Algorithm II originally proposed in [14]; see Fig. 10. This algorithm demonstrates good results in multicriteria optimization. It involves two procedures: the *fast non-dominated sort* (FNDS) of the set of solutions into fronts, which ensures a high convergence rate of the algorithm, and *crowding-distance sort* in the space of functionals, which ensures a good diversity of the population.

Fast non-dominant sort consists in determining the rank r for each individual and combining individuals with the same rank into subsets $\mathcal{F}_r$ (the fronts of rank r).

Definition. A solution p is said to be dominant over a solution q if:

1. p is not worse than q in terms of all functionals.
2. p is strictly better than q in terms of at least one functional. ♦

For the bi-criteria optimization problem under study, we therefore have the following dominance condition of individual p over individual q:

$$\begin{aligned}&\big(f_1(p)\le f_1(q)\wedge f_2(p)\le f_2(q)\big)\\ &\wedge\big(f_1(p)< f_1(q)\vee f_2(p)< f_2(q)\big).\end{aligned}\qquad(10)$$

For the sake of convenience, condition (10) will be written as the relation $p \prec q$.

The FNDS algorithm includes several steps as follows.

Step 1. For each individual p of the population:

- The list $S_p = \emptyset$ is initiated to include the population individuals dominated by individual p.
- The counter $n_p = 0$ is initiated for the population individuals dominating individual p.

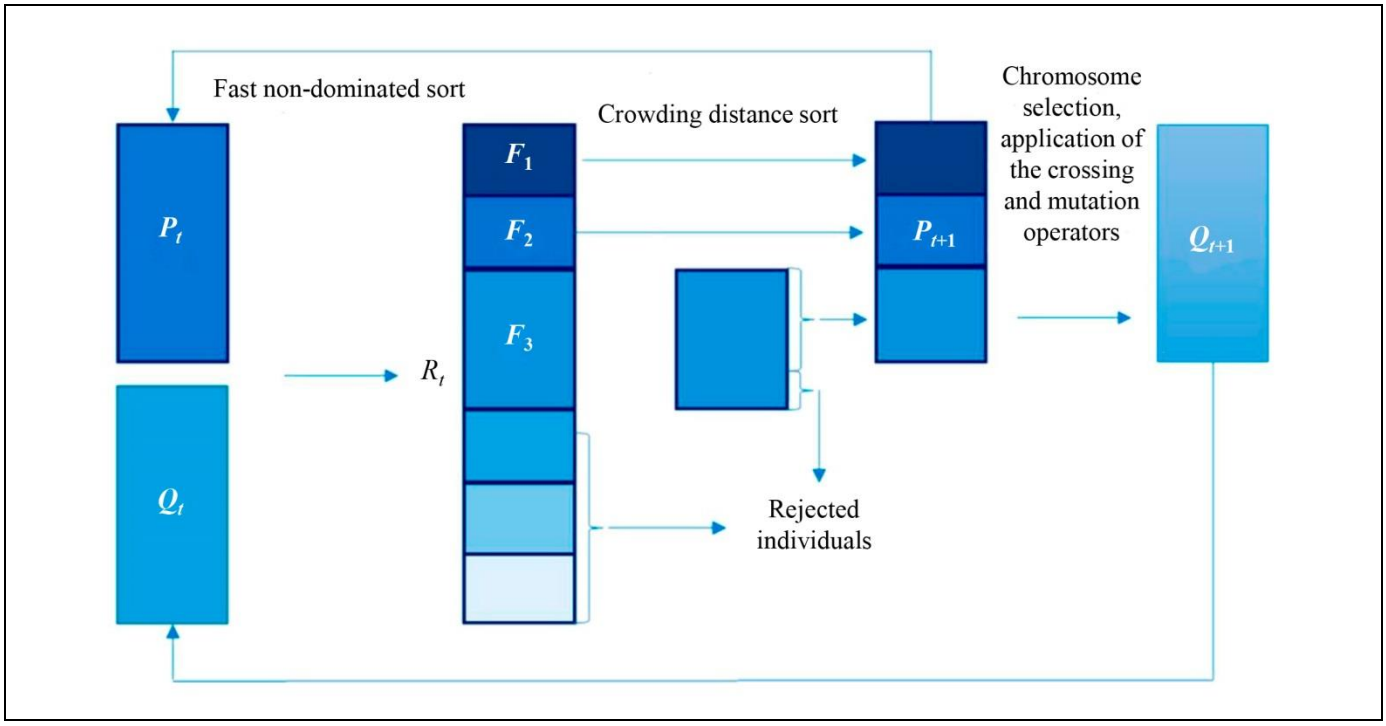

Fig. 10. The population optimization process in NSGA-II.

• The dominance condition (10) is verified for each individual q of the population:

○ If $p \prec q$, then individual q is added in the list S_p, i.e., $S_p = S_p \cup \{q\}$.

○ If $q \prec p$, then the counter n_p is incremented by 1, i.e., $n_p = n_p + 1$.

• If $n_p = 0$, then individual p is written in the first non-dominant front, i.e., $\mathcal{F}_1 \cup \{p\}$.

Step 2. The front counter $i = 1$ is initiated.

Step 3. While $\mathcal{F}_i \neq \emptyset$:

• The list $Q = \emptyset$ is initiated to include the individuals of the $(i + 1)$th front.

• The list S_p is considered for each individual $p \in \mathcal{F}_i$: if $n_q - 1 = 0$, then individual $q \in S_p$ is written into the list Q, i.e., $Q = Q \cup \{q\}$.

• The front counter is incremented by 1, i.e., $i = i + 1$.

• The ith front $\mathcal{F}_i = Q$ is formed.

The procedure continues until all fronts $\mathcal{F}_r$, $r = 1,..., r_{\max}$, are identified.

Crowding-distance sort in the space of functionals is necessary to select solutions within the same front. For each individual $\mathcal{F}_r(i) \in \mathcal{F}_r$, we calculate the value

$$\operatorname{dist}\left(\mathcal{F}_r(i)\right) = \sum_{m=1}^{2} \operatorname{dist}_m\left(\mathcal{F}_r(i)\right),$$

where $\operatorname{dist}_m\left(\mathcal{F}_r(i)\right)$ is the distance to the mth functional obtained as follows.

Step 1. The solutions within the front $\mathcal{F}_r$ of length l are sorted in ascending order of the values of the mth functional. Then the boundary solutions of the front satisfy the relations

$$\mathcal{F}_r(1) = f_m^{\min}, \quad \mathcal{F}_r(l) = f_m^{\max}.$$

Step 2. The desired distances for the boundary solutions are assigned the maximum value $\operatorname{dist}_m\left(\mathcal{F}_r(1)\right) = \operatorname{dist}_m\left(\mathcal{F}_r(l)\right) = 10^9$, and the distances of all intermediate solutions are determined by

$$\operatorname{dist}_m\left(\mathcal{F}_r(i)\right) = \frac{f_m^{\mathcal{F}_r(i+1)} - f_m^{\mathcal{F}_r(i-1)}}{f_m^{\max} - f_m^{\min}},$$
$$i = 2,\ldots, l-1.$$

NSGA-II (see Fig. 10) includes five steps as follows.

Step 1. The initial parent population P_0 of size N is formed randomly. Based on this population, an initial child population Q_0 of size N is generated using the genetic operators of selection, crossing, and mutation. Elitism is introduced by comparing the current population with the best solutions found previously. Therefore, the process of forming subsequent populations will be different.

Step 2. The parent and child populations are combined into the set $R_t = P_t \cup Q_t$, which is divided into fronts F_r, $r = 1,..., r_{\max}$, using fast non-dominated sort.

Step 3. The N fittest individuals are selected from the set R_t to form the new parent population P_{t+1}. The fittest individuals are considered to be those with the smallest rank r. If the next front cannot be written entirely in P_{t+1}, it undergoes crowding-distance sort in the space of functionals; the fittest individuals are those with the largest value of the crowding distance.

Step 4. Based on the population P_{t+1}, the new child population Q_{t+1} is generated using the genetic operators of selection, crossing, and mutation.

Step 5. The algorithm continues from Step 2 until reaching the terminal criterion (a given number of generations or a sufficient degree of population homogeneity).

3. COMPUTATIONAL EXPERIMENTS AND THEIR RESULTS

A simulation software tool was developed to organize computational experiments and investigate the influence of the crossing operator on the efficiency of the algorithm. This tool is used to:

• specify the system state (the set $\{n\}$ of delivery points and the number m of salesmen);

• specify the algorithm parameters (the population size N, the number N_{Gen} of algorithm iterations, the number N_{Rep} of algorithm repetitions, the probability $0 \le p_x \le 1$ of calling the crossing operator, the probability $0 \le p_m \le 1$ of calling the mutation operator, and the seed of the pseudorandom number generator);

• choose the genetic operators of crossing and mutation used in the algorithm;

• collect statistics across generations (in automatic mode) and visualize the final set of Pareto-optimal solutions;

• visualize the final solution on an interactive map if the delivery points in the set are have geographical references;

• create new datasets on the interactive map.

The software tool was implemented using Python 3.9.6 as the programming language. All computations were performed on a PC with AMD Ryzen 7 5800X3D 4.70 GHz CPU and 64 GB RAM.

The following parameter values were selected for all computational experiments: N = 100, N_{Rep} = 1, p_x = 1, and p_m = 0.05. The benchmarks berlin52, eli76, and rat99 [15–17] were used as datasets to compare the results of the algorithm with different crossing operators. Note that the first point was taken as the starting point (see all conditions in the table).

The conditions of computational experiments

Dataset	The number of points, n	The number of salesmen, m	The number of iterations, N_{Gen}
berlin52	52	5	1400
eli76	76	7	1800
rat99	99	7	2200

The graphs in Fig. 11 show the final non-dominated Pareto fronts for the corresponding datasets. They represent the sets of possible solutions of the multiple traveling salesman problem. The abscissa axis corresponds to the total route length whereas the ordinate axis to the difference between the lengths of the longest and shortest routes. (Both values are dimensionless.)

For all benchmark datasets, the Pareto fronts yielded by the algorithm with cHGA have better results in terms of the joint minimization of the two objective functions compared to standard crossing operators.

The evolution graphs of the minimum values of the two objective functions corresponding to the boundary individuals of the current non-dominated Pareto front were plotted to estimate the convergence rate of the algorithm with cHGA. The following pattern is evident for all benchmark datasets: at the first generations, the values of the objective functions decrease rapidly; then, the rate of change drops dramatically (Fig. 12). In a real last-mile delivery system, it is therefore possible to use the solution sets obtained in a small number of iterations to save computational (time) resources.

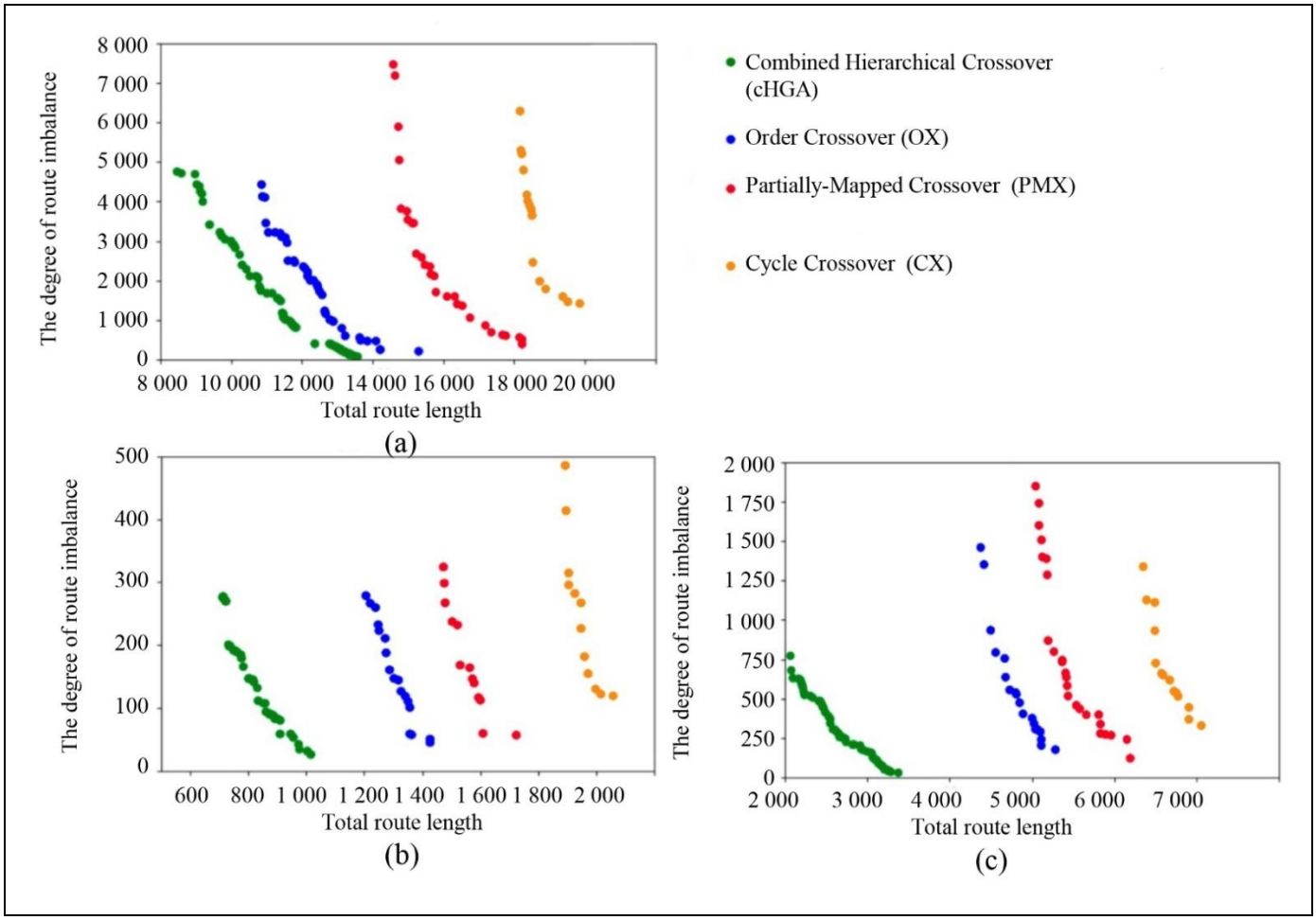

Fig. 11. A comparative analysis of the final non-dominated Pareto fronts for different datasets: (a) berlin52 with m = 5, (b) eli76 with m = 7, and (c) rat99 with m = 7.

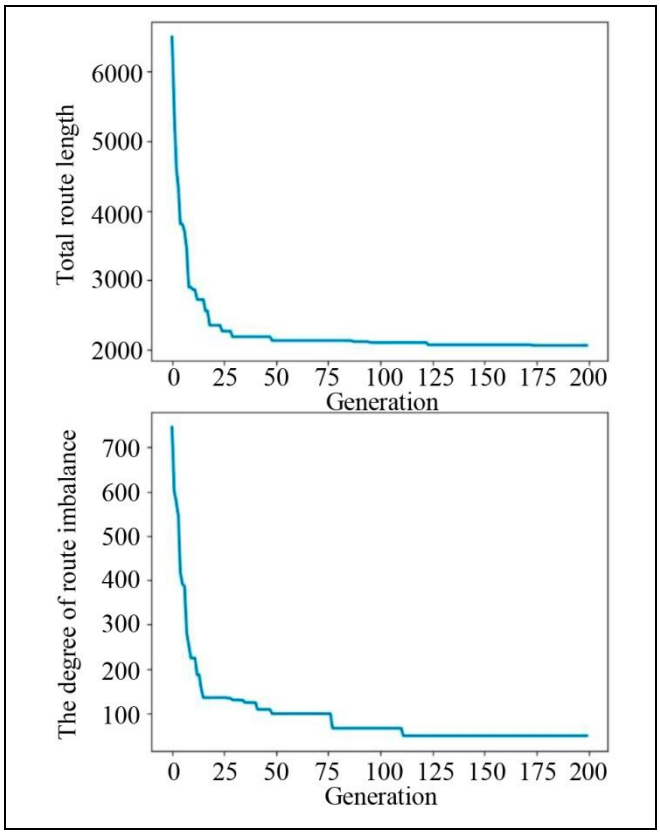

Fig. 12. The evolution of objective function values for the rat99 dataset with m = 7.

For example, we consider the Moscow–ICS dataset created using the interactive map of the software tool. For this dataset, the algorithm with n = 41, m = 5, and N_{Gen} = 200 yielded the solution demonstrated in Fig. 13. For visualization purposes, the Pareto-efficient solution occupying the middle position in the final front of Pareto-optimal solutions is presented in Fig. 14.

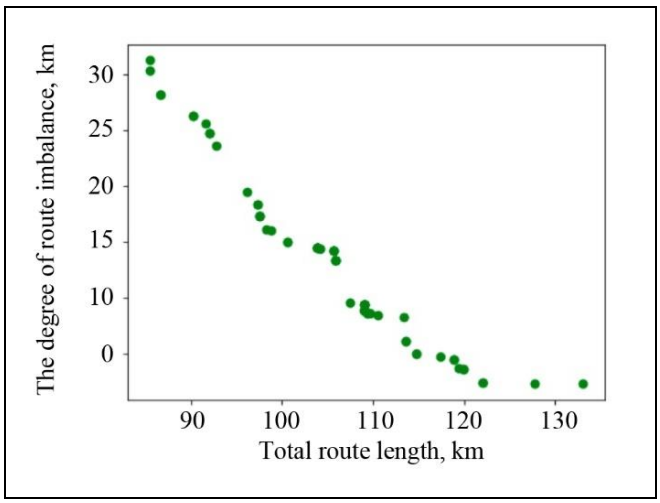

Fig. 13. The final non-dominated Pareto front for the Moscow-ICS dataset.

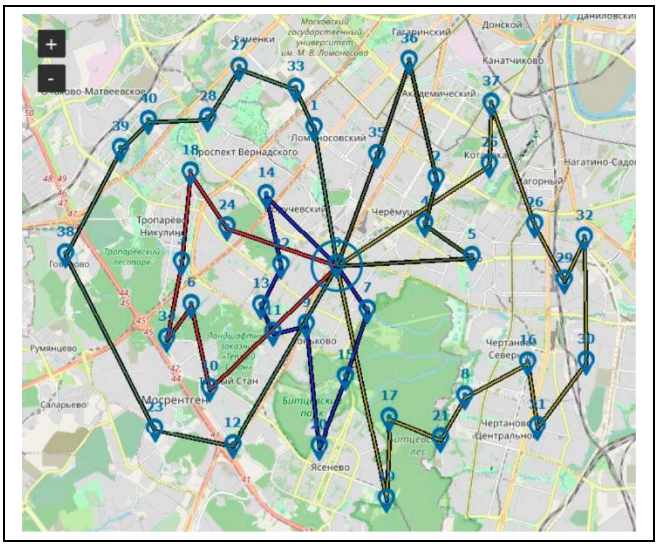

Fig. 14. The solution visualized for the Moscow–ICS dataset.

CONCLUSIONS

This paper has considered the routing problem for a group of UAVs within a promising last-mile delivery system, formalized as a bi-criteria single-depot multiple traveling salesman problem. As has been illustrated, the two conflicting objective functions should be optimized jointly. The genetic operators used have been described. A simulation software tool for this problem has been implemented based on NSGA-II, the elitist non-dominated sorting genetic algorithm.

Computational experiments have been carried out to compare the efficiency of different crossing operators of the genetic algorithm when solving the bi-criteria single-depot multiple traveling salesman problem. According to the experiments, combined hierarchical crossover demonstrates the best results in terms of the joint optimization of two objective functions. Also, the convergence rate of the algorithm with combined hierarchical crossover has been studied. As it has turned out, the proposed algorithm produces acceptable solutions in a short time.

REFERENCES

1. Baur, S., Cargo Drones: A Potential Gamechanger in the Logistics Industry, *Roland Berger*, 2022. URL: https://www.rolandberger.com/en/Insights/Publications/Cargo-drones-A-potential-gamechanger-in-the-logistics-industry.html. (Accessed September 23, 2023.)
2. Moadab, A., Farajzadeh, F., and Fatahi Valilai, O., Drone Routing Problem Model for Last-Mile Delivery Using the Public Transportation Capacity as Moving Charging Stations, *Scientific Reports*, 2022, vol. 12, no. 1, pp. 1–16.
3. Khoufi, I., Laouiti, A., and Adjih, C., A Survey of Recent Extended Variants of the Traveling Salesman and Vehicle

Routing Problems for Unmanned Aerial Vehicles, *Drones*, 2019, vol. 3, no. 3, art. no. 66.
4. Germanchuk, M.S., Lemtyuzhnikova, D.V., and Lukianenko, V.A., Metaheuristic Algorithms for Multi-Agent Routing Problems, *Control Sciences*, 2020, no. 6, pp. 3–13. (In Russian.)
5. Bektas, T., The Multiple Traveling Salesman Problem: an Overview of Formulations and Solution Procedures, *Omega*, 2006, vol. 34, no. 3, pp. 209–219.
6. Necula, R., Breaban, M., and Raschip, M., Tackling the Bi-criteria Facet of Multiple Traveling Salesman Problem with Ant Colony Systems, *Proceedings of the IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*, Vietri sul Mare, 2015, pp. 873–880.
7. Bolanos, R., Echeverry, M., and Escobar, J., A Multiobjective Non-dominated Sorting Genetic Algorithm (NSGA-II) for the Multiple Travelling Salesman Problem, *Decision Science Letters*, 2015, vol. 4, pp. 559–568.
8. Alves, R.M.F. and Lopes, C.R., Using Genetic Algorithms to Minimize the Distance and Balance the Routes for the Multiple Travelling Salesman Problem, *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, Sendai, 2015, pp. 3171–3178.
9. Carter, A.E. and Ragsdale, C., A New Approach to Solving the Multiple Traveling Salesperson Problem Using Genetic Algorithms, *European Journal of Operational Research*, 2005, vol. 175, no. 1, pp. 246–257.
10. Simon, D., *Evolutionary Optimization Algorithms*, New York: John Wiley & Sons, 2013.
11. Gladkov, L.A., Kureichik, V.V., and Kureichik, V.M., *Geneticheskie algoritmy* (Genetic Algorithms), 2nd ed., Moscow: FIZMATLIT, 2010. (In Russian.)
12. Shuaia, Y., Yunfengaand, S., and Kai, Z., An Effective Method for Solving Multiple Travelling Salesman Problem Based on NSGA-II, *Systems Science & Control Engineering*, 2019, vol. 7, no. 2, pp. 108–116.
13. Soni, N. and Kumar, T., Study of Various Mutation Operators in Genetic Algorithms, *International Journal of Computer Science and Information Technologies*, 2014, vol. 5, no. 3, pp. 4519–4521.
14. Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T., A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II, *IEEE Transactions on Evolutionary Computation*, 2002, vol. 6, no. 2, pp. 182–197.
15. *Benchmark Data for the Single-Depot Multiple Traveling Salesman Problem (Multiple-TSP)*, Iaşi: Alexandru Ioan Cuza University (UAIC). URL: https://profs.info.uaic.ro/~mtsplib/ (Accessed September 23, 2023.)
16. *TSPLIB. Symmetric Traveling Salesman Problem (TSP)*, Heidelberg: University of Heidelberg. URL: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/ (Accessed September 23, 2023.)
17. *TSPLIB. Capacitated Vehicle Routing Problem (CVRP)*, Heidelberg: University of Heidelberg. URL: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/vpr (Accessed September 23, 2023.)

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

Received May 15, 2023, and revised November 12, 2023. Accepted November 29, 2023.

Author information

Sosedov, Vladislav Aleksandrovich. Engineer, Trapeznikov Institute of Control Problems, Russian Academy of Sciences, Moscow, Russia
✉ vladyslav.sosedov@gmail.com
ORCID iD: https://orcid.org/0009-0001-0920-2579

Cite this paper

Sosedov, V.A., Combined Hierarchical Crossover in a Genetic Algorithm for Last-Mile Delivery: Efficiency Analysis. *Control Sciences* **1,** 18–27 (2024). http://doi.org/10.25728/cs.2024.1.3

Original Russian Text © Sosedov, V.A., 2024, published in *Problemy Upravleniya*, 2024, no. 1, pp. 23–34.

Translated into English by *Alexander Yu. Mazurov*,
Cand. Sci. (Phys.–Math.),
Trapeznikov Institute of Control Sciences,
Russian Academy of Sciences, Moscow, Russia
✉ alexander.mazurov08@gmail.com