Improving the Reference Route Using a Stack of Iterative Local Optimization Methods

Volodymyr Shevchenko[1,†], Yevhen Derevianko[2,†], Oleksii Korniienko[2,†] and Viktor Shevchenko[2,†,*]

[1] King's College London, Strand, London WC2R 2LS, United Kingdom
[2] Institute of Software Systems of the National Academy of Sciences of Ukraine, 40 Academician Glushkova Avenue, Building 5, Kyiv, Ukraine, 03187

Abstract

The work develops methods for constructing a suboptimal flight route for an unmanned aerial vehicle (UAV) that performs the task of eliminating the consequences of an emergency. The UAV must sequentially fly around a certain number of route points. Each point must be visited only once. The UAV returns to the same point from which it started. This task can be classified as a traveling salesman problem (TSP). At the beginning of the work, an analysis of existing research on solving the traveling salesman problem was performed. Methods of complete direct search, the method of ordered search based on a discrete dynamic programming procedure (Held-Karp method), integer linear programming methods, greedy algorithms, and local optimization methods were considered. A conclusion was made regarding the feasibility of solving the problem of constructing an optimal route using a combination of greedy algorithms and iterative methods for improving the solution. The resulting solution is not guaranteed to be optimal, but is close to optimal, that is, suboptimal. At the first stage of calculations, a reference trajectory is obtained using the greedy nearest neighbor algorithm. Then, local optimization methods are used to improve the reference trajectory. The methods of moving one point, exchanging two points, and deleting intersections of route sections are used as local optimization methods. During the search computational procedure, individual methods of improving the reference solution can enter local optimum, from which they cannot then escape. A combination of different methods allows you to leave local optimum, bypass ravine areas, and approach the global optimum. The use of iteration methods in the stack loop allows you to quickly find optimal UAV trajectories. The speed of the algorithms allows you to calculate the optimal route not only before the flight, but also to recalculate it during the flight, in case of a change in the task or the conditions for its execution. The performance of the proposed approach was tested by creating a model in the MatLab environment.

Keywords

emergency situation, drone, flight route, route points, optimization, reference solution, model, computational methods

1. Introduction

The problems of constructing optimal routes have appeared a long time ago: the problems of delivering goods by transport, delivering mail by postmen, delivering goods by traveling salesmen, etc. According to the last type of delivery, the problem was called Traveling Salesman Problem (TSP). These problems require routes that save time, money, etc. At the same time, the traveling salesman had to visit each point only once, and also finish at the point from which he started. The relevance of the problem has not decreased over the years. On the contrary, it has increased. Therefore, the relevance of methods for solving the problem is only growing. In this study, the TSP

Workshop "Software engineering and semantic technologies" SEST, co-located with 15th International Scientific and Practical Programming Conference UkrPROG'2025, May 13-14, 2025, Kyiv, Ukraine
* Corresponding author.
† These authors contributed equally.
vladimir_337@ukr.net (Volodymyr Shevchenko); evg.derevjanko@gmail.com (Y. Derevianko); oleksiikorniienko@gmail.com (O. Korniienko); gii2014@ukr.net (Viktor Shevchenko)
0000-0002-2152-6816 (Volodymyr Shevchenko); 0000-0003-2949-8896 (Y. Derevianko); 0009-0006-9289-7995 (O. Korniienko); 0000-0002-9457-7454 (Viktor Shevchenko)

is solved to form the optimal route of an unmanned aerial vehicle (UAV), which is used to eliminate the consequences of an emergency.

2. Analysis of existing research

Let us consider the main methods for solving the TSP for a route with n points to be visited.

The Brute Force method has a computational complexity of (n-1)! (for the symmetric case, the cost of a route section is (n-1)!/2) [1, 2, 3]. The method has a limitation on the maximum number of route points, which depends on the computational capabilities. In most cases, the method is used when the number of route points is up to 10. The ordered search method is close to the brute force method, but due to the rejection of unpromising route sections, it has a lower computational complexity of n^2 * 2^n. This is the discrete dynamic programming method (Held-Karp algorithm). This allows you to find optimal routes with up to 20-25 points. The disadvantage is the need for large memory to store intermediate chains of route sections. The advantage of the methods of brute-force selection is the guaranteed optimality of the found routes. The disadvantage is the impossibility of optimizing routes with the number of points more than 25.

Linear programming methods can be considered classical. They have a clear mathematical formalization of the problem statement in the form of an integer linear programming problem. Unfortunately, they also require a lot of computational resources and can lose even to direct brute-force methods. These methods can be effective in the case of a certain set of constraints of a special type, which can lead to a decrease in the order of the problem (branch-and-bound method) [4].

Let us clarify the features of the statement of the applied TSP:

1. The solution does not necessarily have to be perfectly accurate. It can be close to accurate, that is, not optimal, but suboptimal.
2. But the solution must be timely, even if you have to pay for it with accuracy.

Conclusion: you need to use methods or a combination of methods that provide an acceptable solution in an acceptable time. To do this, at the first stage it is necessary to quickly find a reference solution, which can then be refined using other methods.

Greedy algorithms are fast, but inaccurate [4, 5, 6]. They allow finding a reference solution that may be far from optimal. After obtaining the reference solution, it is refined using local optimization methods such as 2-opt, 3-opt (permutation of segments). Genetic algorithms, ant algorithms, annealing algorithms are also used [4, 7, 8]. Refining the reference solution is a good approach that has good convergence. However, existing studies have not paid enough attention to the combination of such methods in a single set of nested iterative procedures.

In [9], the construction of a reference trajectory using a greedy algorithm (Nearest Point Method) was already considered. The shortcomings of such routes were noticeable even during visual control of their quality. Therefore, just like other researchers, the authors defined the found route as a reference and then improved it in an iterative procedure by moving the route points to new places in the sequence of route points. For each route point, all possible new positions were checked. For each new position of the point, a new route length was calculated. If the movement of the point improved the current result, then such a movement was fixed. If it did not improve, then it was ignored. In general, such a procedure led to a noticeable improvement in the result, but it often also looked imperfect.

Analysis of existing studies allowed us to draw the following conclusions:

1. For a large number of route points (more than 20-30), methods of direct enumeration of all options and even methods of ordered enumeration of options such as the discrete Bellman procedure consume too much computational resources.
2. Greedy algorithms (Nearest Point Method) are the most economical in terms of machine resources, but inefficient in terms of solution quality.

3. It seems advisable to quickly build a reference solution using greedy algorithms with further improvement of the reference solution using special iterative methods.
4. The method of improving the reference trajectory by sequentially moving individual route points to new places in the sequence of route points cannot be considered sufficient. Therefore, it would be advisable to supplement (combine) it with other methods of improving the reference solution.

The aim of the article is to improve the quality of route construction by quickly building a reference solution and its further improvement by cyclic application of a stack of special iterative methods. The task is to develop or adapt existing methods, as well as to determine the scheme of their joint (sequential) use and to determine the criteria for stopping the iterative process.

3. Building a reference route

A reference route is one that is constructed using simplified and, accordingly, fast methods. For example, greedy algorithms (NPM, Nearest Point Method) are such. In the future, the reference solution must be refined using other methods that do not require a large amount of computing resources. To find the distance between two points, we use the Euclidean measure

$$D_e(i,j) = \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2}, \quad (1)$$

where i, j - point numbers; x, y - coordinates of points.

In general, instead of distance, the cost of passing a route section should be used

$$Cost(i,j) = D_e(i,j). \quad (2)$$

The cost of passing a route section may include: distance, energy consumption, UAV resource use, probability of damage or destruction of the UAV, probability of other incidents, probability of failure to complete the task, etc. The cost can also be found as a certain aggregation of a certain set of factors (listed or others). For this, scalar convolutions can be used, for example, additive, multiplicative, others or a combination thereof [10, 11].

In the study, we distinguish the following sets of route points:

LocationSet. An unordered set of route point numbers that are assigned to certain locations in a list sequentially or in random order

$$LocationSet = \{1, 2, 3, \ldots . . N\}. \quad (3)$$

RouteSet. An ordered set of route points (tuple) that consists of the same points, but now arranged in an order that corresponds to a specific route.

$$RouteSet = (3, 7, 1, \ldots . . 4, N, 10, \ldots). \quad (4)$$

The cost of such a route is equal to the sum of the costs of passing all sections between neighboring points.

$$CostRoute = \sum_{RouteSet, i=2,N} Cost(i, i-1). \quad (5)$$

In this study, we assume that the cost of a route is equal to its length, as the sum of the distances between neighboring sections.

$$CostRoute = \sum_{RouteSet, i=2,N} D_e(i, i-1). \quad (6)$$

The procedure for finding the route length is the same for the reference and improved routes.

4. Improving the reference route

Improvement of the reference route can be performed using various methods. Let's analyze those used in this study.

4.1. 1-Point Moving Method (1PM)

We have already considered the 1PM method above when analyzing the publication [9]. The disadvantage of the method is that it does not go through all possible route change options. Therefore, the procedure of complete analysis of possible movements of all points should be repeated several times until the improvement of the quality indicator (route length) stops.

But previous shuffling can make certain successful solutions unavailable for the procedure of subsequent shuffling. By analogy with gradient search procedures, we can say that the search can enter a local optimum, which is separated from the global optimum by a ravine. The way out of this situation is to shake (shuffle) the route tuple using other methods that allow you to bypass or jump over the ravine.

4.2. 2-Point Exchange Method (2PE)

This is exactly what the 2PE method may be in many cases, analyzing the feasibility of exchanging two points in a tuple describing a route. This procedure analyzes the exchange of points for all possible pairs of points. The procedure can be repeated several times. The disadvantage of the method is that it can create new intersections of route sections (segments), which immediately indicates its non-optimality. Such a newly formed intersection can appear with the participation of points that have already been analyzed and therefore do not fall into repeated checks. It would be desirable to have a separate method that eliminates only intersections of route sections and does not touch other points that are not related to the intersection.

4.3. Delete Crossing Method (DC)

The DC method is aimed at eliminating intersections of route sections (Figure 1, 2). The method can be considered a special case of the 2PE method. The advantage of the method is that it purposefully eliminates only intersections. There remains a small probability that the elimination of one intersection will form a new intersection. But this probability is significantly less than in the 2PE method.

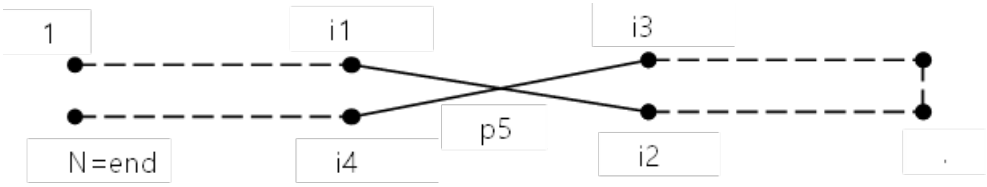

Figure 1: Route segments i1-i2 and i3-i4 intersecting (before intersection elimination)

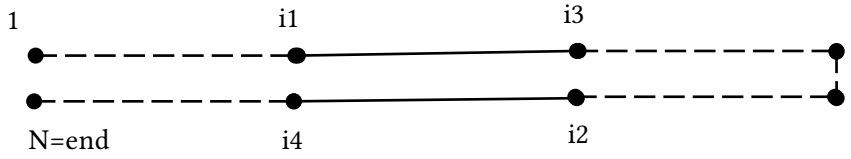

Figure 2: Route sections i1-i3 and i2-i4 that do NOT intersect (after eliminating the intersection)

Even based on visual analysis, it can be concluded that the exchange of points i2, i3 should lead to a reduction in the length of the route (Figure 1, 2). Since the elimination of one intersection can form another intersection, the result of each replacement is checked for a reduction in the length of

the route. Let us consider the calculated dependencies. The equations of the lines passing through the corresponding pairs of points before the elimination of the intersection have the form

$$\begin{cases} y = a_{12}x + b_{12} \\ y = a_{34}x + b_{34} \end{cases}. \quad (7)$$

Substitute the coordinates of the node points

$$\begin{cases} y_1 = a_{12}x_1 + b_{12} \\ y_2 = a_{12}x_2 + b_{12} \end{cases}; \quad \begin{cases} y_3 = a_{34}x_3 + b_{34} \\ y_4 = a_{34}x_4 + b_{34} \end{cases}. \quad (8)$$

We find the coefficients

$$a_{12} = \frac{y_2 - y_1}{x_2 - x_1}; \quad b_{12} = y_1 - a_{12}x_1; \quad a_{34} = \frac{y_4 - y_3}{x_4 - x_3}; \quad b_{34} = y_3 - a_{34}x_3. \quad (9)$$

We write the equation for the intersection point p5.

$$\begin{cases} y_5 = a_{12}x_5 + b_{12} \\ y_5 = a_{34}x_5 + b_{34} \end{cases}. \quad (10)$$

Subtract the upper equation from the lower one.

$$0 = x_5(a_{34} - a_{12}) + (b_{34} - b_{12}). \quad (11)$$

Convert. Find the coordinates of the intersection point

$$x_5 = -\frac{b_{34} - b_{12}}{a_{34} - a_{12}}; \quad y_5 = a_{12}x_5 + b_{12}. \quad (12)$$

Next, we need to make sure that the intersection point really lies on the segments, and not on their imaginary extension. To do this, we find the minimum and maximum abscissas and ordinates for each pair of points (for each segment)

$$\begin{aligned} x_{12min} &= min(x_1, x_2); & x_{34min} &= min(x_3, x_4); \\ y_{12min} &= min(y_1, y_2); & y_{34min} &= min(y_3, y_4); \\ x_{12max} &= max(x_1, x_2); & x_{34max} &= max(x_3, x_4); \\ y_{12max} &= max(y_1, y_2); & y_{34max} &= max(y_3, y_4). \end{aligned} \quad (13)$$

A sign that the intersection point lies on the segments is the simultaneous fulfillment of the following conditions

$$\begin{aligned} &x_{12min} \le x_5; \quad x_{34min} \le x_5; \quad x_{12max} \ge x_5; \quad x_{34max} \ge x_5; \\ &y_{12min} \le y_5; \quad y_{34min} \le y_5; \quad y_{12max} \ge y_5; \quad y_{34max} \ge y_5. \end{aligned} \quad (14)$$

Analysis of Figure 1 shows that it is sufficient to perform the check on only one coordinate. This will significantly reduce the computational load.

$$x_{12min} \le x_5; \quad x_{34min} \le x_5; \quad x_{12max} \ge x_5; \quad x_{34max} \ge x_5. \quad (15)$$

Reducing the computational load can also be achieved by replacing conditions (13), (14) with checking the distances between points along one coordinate.

$$\begin{aligned} |x_1 - x_2| \ge |x_1 - x_5|; \quad |x_1 - x_2| \ge |x_2 - x_5|; \\ |x_3 - x_4| \ge |x_3 - x_5|; \quad |x_3 - x_4| \ge |x_4 - x_5|. \end{aligned} \quad (16)$$

If conditions (13), (14) or (13), (15) or (16) are met, then points i2 and i3 are swapped. It is clear that such a modification of the route may not be final. The method can be applied repeatedly until the condition for stopping the iterative procedure is met.

4.4. Nesting of iterative procedures

Let us pay attention to the fact that the proposed approach to improving the reference solution provides for several levels of iterative procedures:

Level 1. Iterations at the level of route points (within the method). All possible route change options are reviewed with an analysis of possible changes for all points or all pairs of route points.

Level 2. Iterations at the level of individual methods. One iteration includes a single application of a separate method for all points or all pairs of route points (one of the methods: 1PM, 2PE, DC). And such iterations at the level of one method can be repeated.

Level 3. Iterations at the level of a set of different methods. One iteration includes the sequential execution of components of a certain set of methods, represented by a tuple (2PE, 1PM, DC). If the sequence of methods can be repeated R times, then the scenario for creating and refining the reference solution can be represented by the following chain of methods

$$NPM + (2PE + 1PM + DC) * R. \quad (17)$$

The order of the 1PM, 2PE, DC methods may vary depending on the specifics of the route and the experience of the researcher. It can also be changed in different iterations at the level of the set of methods.

4.5. Conditions for stopping iterative procedures of reference route refinement methods

The stopping conditions of the reference route refinement procedure are used only for Iterations at the level of individual methods and for iterations at the level of a set of methods.

The condition for stopping the iterative procedure at level 2 (of a separate method) is the equality of the results of two consecutive iterations of this method.

$$CostRoute(end) = CostRoute(end - 1). \quad (18)$$

After that, the transition to the next method in the tuple occurs (2PE, 1PM, DC).

The condition for stopping the iterative procedure at level 3 (set of methods) is the equality of the results of all methods within the current iteration at level 3

$$CostRoute2PE(end) = CostRoute1PM(end) = CostRouteDC(end). \quad (19)$$

5. Method validation

The proposed approach was tested in the MatLab software environment. Let us consider the results of the simulation by iterations at level 2 (separate methods) for a route including 50 points (Figure 3-6). The red circle indicates the location of the base from which the UAV takes off and returns to. The blue line indicates the reference route that was built using the greedy algorithm, the red line is the route obtained as a result of improving the reference route at this iteration at level 2.

Figure 7 shows the results of the numerical experiment. To the left of the equal sign is the abbreviated name of the method used. To the right of the equal sign are the route lengths that were achieved using this method at each iteration of level 2. The protocol of using the reference route improvement methods (Figure 7) shows that the 2PE method reached its potential already at the second iteration of level 2, the 1PM method implemented 4 successful iterations, and the DC method successfully worked only once. In the next iteration cycle of level 3 (multiple methods), the 2PE method was unable to add any improvements to the solution in the iterations of level 2. But after that, the 1PM method successfully implemented two iterations of the reference route improvement at level 2. Further repetitions of using the methods did not bring any improvements.

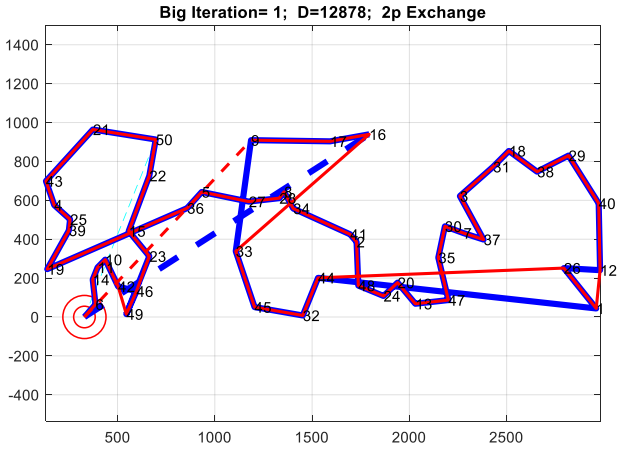

Figure 3: Result of improving the reference route using the 2PE method.

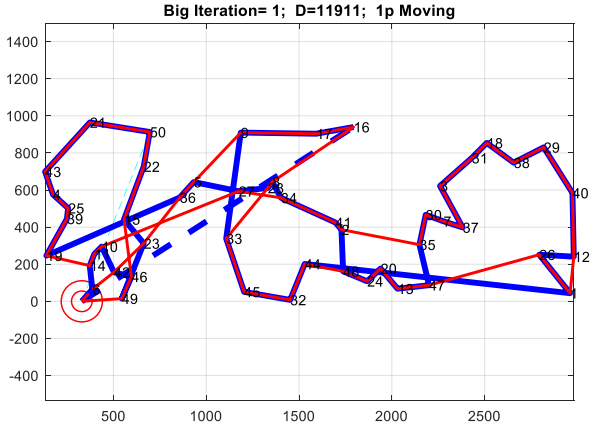

Figure 4: Result of improving the reference route using the 1PM method.

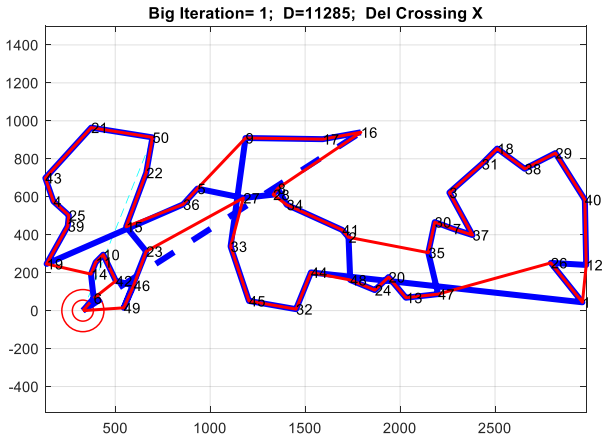

Figure 5: Result of improving the reference route using the DC method.

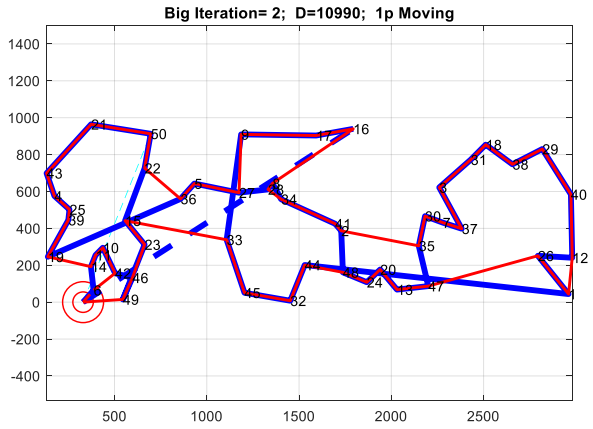

Figure 6: Result of improving the reference route using the 2PE method.

```
Nearest   = 13271.8873
2p Exchange    = 12995.0344      12877.7633      12877.7633
1p Moving      = 12494.982       12301.2062      11952.5211      11910.7467      11910.7467
Del Crossing X = 11284.9747      11284.9747
2p Exchange    = 11284.9747      11284.9747
1p Moving      = 10994.8736      10989.5421      10989.5421
Del Crossing X = 10989.5421      10989.5421
2p Exchange    = 10989.5421      10989.5421
1p Moving      = 10989.5421      10989.5421
```

Figure 7: Reference route improvement procedure protocol.

The numerical experiment allows us to draw the following conclusions:

1. Changing methods can improve the result.
2. In some cases, the result can be improved by a method that seemed to have already done everything it could in the previous iteration (for example, 1PM. Otherwise, it may be another method).

6. Conclusions

The paper presents approaches to solving the TSP for constructing a UAV route involved in eliminating the consequences of emergency situations.

The resulting solution is not guaranteed to be optimal, but is close to optimal, i.e. suboptimal.

At the first stage of calculations, a reference trajectory is obtained using a greedy nearest neighbor algorithm.

Then, local optimization methods are used to improve the reference trajectory.

The methods of moving one point, exchanging two points, and eliminating intersections of route sections are used as local optimization methods.

During the computational procedure, individual methods of improving the reference solution may enter local optimum, from which they cannot then escape. A combination of different methods allows you to exit local optimum, bypass ravine areas, and approach the global optimum.

Using a stack of iteration methods in a loop allows you to quickly find suboptimal UAV trajectories.

The speed of the algorithms allows you to calculate the optimal route not only before the flight, but also to recalculate it during the flight, in case the task or the conditions for its execution change.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] Heping Jiang, The 1-1 algorithm for Travelling Salesman Problem, CoRR, v. abs/2104.13197 (2021) https://doi.org/10.48550/arXiv.2104.13197, https://arxiv.org/abs/2104.13197, https://dblp.org/rec/journals/corr/abs-2104-13197.bib.

[2] Zakir Hussain Ahmed, Ibrahim Al-Dayel, An Exact Algorithm for the Single-Depot Multiple Travelling Salesman Problem, IJCSNS International Journal of Computer Science and Network Security, VOL.20 No.9 (2020) DOI: 10.22937/IJCSNS.2020.20.09.9. https://www.researchgate.net/publication/344463896_An_Exact_Algorithm_for_the_Single-Depot_Multiple_Travelling_Salesman_Problem

[3] Chandra Agung, Natalia Christine, Performance analysis of optimization methods for solving traveling salesman problem, Innovative Technologies and Scientific Solutions for Industries, No. 1 (15) (2021) 69–75. doi: https://doi.org/10.30837/ITSSI.2021.15.069, https://www.researchgate.net/publication/350515816_PERFORMANCE_ANALYSIS_OF_OPTIMIZATION_METHODS_FOR_SOLVING_TRAVELING_SALESMAN_PROBLEM.

[4] Nataliya Boyko, An Overview of Existing Methods for Solving the Travelling Salesman Problem in Order to Find the Optimal Solution for Economic Problems, European scientific journal of Economic and Financial innovation. №1(7) (2021). doi: http://doi.org/10.32750/2021-0108, https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://journal.eae.com.ua/index.php/journal/article/download/129/115/&ved=2ahUKEwjMnq-k0ZuNAxWyExAIHZqeG8QQFnoECB8QAQ&usg=AOvVaw0OzPk2F2kE3LoWlAP23rNG.

[5] V. Trotsko, I. Chernozubkin, Y. Doryshyn, Evaluation of the Practical Use of Search Algorithms for Solving Logistics Tasks Based on the Traveling Salesman's Task, Academic notes of the University "KROK", № 3(67) (2022) 69-73. doi https://doi.org/10.31732/2663-2209-2022-67-69-73, https://dspace.krok.edu.ua/handle/krok/2358.

[6] V.V. Trotsko, I.O. Chernozubkin, Combining the greedy algorithm with the Monte Carlo method for solving logistics problems based on the traveling salesman problem, Scientific notes of the KROK University, 2021, № 2(62). (2021) 125-131. doi: https://doi.org/10.31732/2663-2209-2021-62-125-131., http://snku.krok.edu.ua/vcheni-zapiski-universitetu-krok/article/view/414/441.

[7] Maha Ata Al-Furhud, Zakir Hussain Ahmed, Genetic Algorithms for the Multiple Travelling Salesman Problem, International Journal of Advanced Computer Science and Applications(IJACSA), 11(7) (2020) http://dx.doi.org/10.14569/IJACSA.2020.0110768.

[8] Dharm Raj Singh, Manoj Kumar Singh, Tarkeshwar Singh, Rajkishore Prasad, Genetic Algorithm for Solving Multiple Traveling Salesmen Problem using a New Crossover and Population Generation, Comp. y Sist. vol.22 no.2. Mexico City Apr./Jun. 2018 Epub Jan 21, (2021), https://doi.org/10.13053/cys-22-2-2956 , https://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S1405-55462018000200491.

[9] Igor Sinitsyn, Yevhen Derevianko, Stanislav Denysyuk, Volodymyr Shevchenko, Optimizing Reference Routes through Waypoint Sequence Variation in Emergency Events of Natural and Technological Origin, in: Proceedings of the Cybersecurity Providing in Information and Telecommunication Systems II (CPITS-II 2024), Kyiv, Ukraine, October 26, 2024 (online), CEUR Workshop Proceedings, ISSN 1613-0073, 2024, Vol-3826, pp. 505-512, https://ceur-ws.org/Vol-3826/short20.pdf.

[10] Viktor Shevchenko, Oleh Bakaiev, Ihor Syvachenko, Scalarization of the vector criterion of information system survivability based on information security indicators, in: Proceedings of the Cybersecurity Providing in Information and Telecommunication Systems II (CPITS-II 2024), Kyiv, Ukraine, October 26, 2024 (online), CEUR Workshop Proceedings, ISSN 1613-0073, 2024, Vol-3826, pp. 294-300, https://ceur-ws.org/Vol-3826/short20.pdf.

[11] Viktor L. Shevchenko, Optimization Modeling in Strategic Planning, TsVSD NUOU, 2011.