Optimizing Reference Routes through Waypoint Sequence Variation in Emergency Events of Natural and Technological Origin

Igor Sinitsyn[1], Yevhen Derevianko[1], Stanislav Denysyuk[1], and Volodymyr Shevchenko[2]

[1] *Institute of Software Systems of the National Academy of Sciences of Ukraine, 40 Academician Glushkova ave., Building 5, Kyiv, 03187, Ukraine*
[2] *Taras Shevchenko National University of Kyiv, 64/13, Volodymyrska str., Kyiv, 01601, Ukraine*

Abstract

The likelihood of rescuing people caught in emergencies of natural or technological origin depends on the speed with which they are provided the necessary information. In conditions where normal communication means (radio, television, Internet) are disrupted, unmanned aerial vehicles (drones) can serve as an alternative means of notification, providing information to the population through built-in audio broadcasting systems or by dropping instructions on paper, plastic, and other carriers. Similarly, essential cargo such as rescue equipment, food, water, etc., can be dropped. In the context of the rapid flow of dangerous processes and the limited number of drones available, optimizing the flight route for time becomes crucial. This work analyses existing methods of constructing flight routes through a given set of waypoints: exhaustive search methods, Bellman's discrete dynamic programming method, greedy algorithms (nearest neighbor method), and anytime algorithms. To save onboard computer resources, a method for improving the reference trajectory, which was initially found using the nearest neighbor method, was chosen. Improving the reference trajectory is done by varying the sequence of waypoints. The advantage of this approach is that at each step of the iterative improvement of the trajectory, new information that has become available at the current time can be taken into account. Changes to the set of necessary waypoints can occur due to changes in the emergency, the completion of tasks by other drones, or the failure of other drones. The workability of the algorithm with low computational time costs onboard computers was tested for routes that include from 8 to 200 points.

Keywords

Emergency, drone, flight route, route points, optimization, reference solution.

1. Introduction

The primary task in ensuring the safety of the population caught in an emergency zone is to provide notifications as quickly as possible under conditions of potential disruption to conventional means of communication—radio, television, Internet [1–3].

Unmanned aerial vehicles (drones) can serve as an alternative means of notification [4]. In the absence of the possibility for emergency service personnel to quickly enter the danger zone, drones can fly into the required area and provide the necessary information to the population through audio messages using built-in broadcasting equipment or by dropping instructions on paper, plastic, and other media. Similarly, essential supplies such as rescue equipment, food, water, etc., can be dropped.

The relevance of this work lies in the fact that the likelihood of rescuing people in danger significantly depends on the speed of providing

CPITS-2024: Cybersecurity Providing in Information and Telecommunication Systems, February 28, 2024, Kyiv, Ukraine
EMAIL: ips2014@ukr.net (I. Sinitsyn); evg.derevjanko@gmail.com (Y. Derevianko); sstdenysyuk@gmail.com (S. Denysyuk); vladimir_337@ukr.net (V. Shevchenko)
ORCID: 0000-0002-4120-0784 (I. Sinitsyn); 0000-0003-2949-8896 (Y. Derevianko); 0009-0006-9308-6959 (S. Denysyuk); 0000-0002-2152-6816 (V. Shevchenko)

them with the necessary information and material aid.

In conditions of the rapid flow of dangerous processes of natural and technological origin, and the limited resources (fleet) of unmanned vehicles, optimizing the flight route for time becomes a critical issue.

During an emergency, conditions can change very quickly, with new points being added or existing ones disappearing from the route. There is a need to develop methods that allow for finding optimal routes using the limited capabilities of the unmanned aerial vehicle's onboard computer, not only in conditions of communication with the base but also during autonomous flight. Under certain conditions, an unmanned aerial vehicle can identify the situation in real time through onboard sensors or can receive information from other drones nearby [5]. This necessitates the constant recalculation of the optimal route, taking into account new data received at the current moment. This is critical because it increases the chances of saving people's lives.

2. Literature Analysis

Let's analyze the state of existing research on creating optimal flight routes for unmanned aerial vehicles.

In the tasks of constructing optimal routes, Dijkstra's algorithm [6] is very common, but it is mostly used for building the shortest route between two points. In this case, passing through all points is not mandatory.

The discrete dynamic programming method of Bellman [7] allows finding optimal routes, including solving the problem in reverse time. Among optimization specialists, there is a widespread saying attributed to Bellman: "Any optimization problem can be solved provided there are sufficient computational resources" (the authors do not have information on whether this quote is accurate, or whether the information about the authorship of the quote is reliable). However, many researchers, including [8, 9], note that the cost of calculations can become a limitation that, in the conditions of a specific problem setting, does not allow the use of good iterative methods with good convergence, such as the Bellman method.

Therefore, researchers' attention remains on fast, albeit imperfect methods like greedy algorithms [10–12], which provide decent results if the solution quality requirements are not high. The drawback of the greedy algorithm is that it finds the best solution at the current iteration step without considering the consequences this current decision might have in the future. The low quality of the solution is the price paid for the speed of obtaining the decision.

In other situations, the greedy algorithm is used to find a reference solution that will then be refined by more precise methods. Sometimes these algorithms are called anytime algorithms, to note the fact that the iterative procedure for improving the reference solution can be interrupted at any moment, and the solution obtained at that moment will be at least no worse than the previous ones [13, 14]. In our view, a more accurate name would be the "improving the reference solution" method [15], as it reflects the essence of the actions taken about existing solutions. However, this does not change the essence of the optimization process.

Details of the processes occurring when working with anytime algorithms for finding fault routes in cloud systems are considered in [16]. In [17], improving the reference solution is called continuous optimization and is used for optimizing control in the well-known game Pinball, where flippers are used to control metal balls on a physical playfield covered with glass. In [8], the focus is on, perhaps, the most significant advantage of anytime algorithms, namely, that the algorithm allows for considering new information that arrives in real-time at each iteration of improving the reference solution. The disadvantages of the considered anytime algorithms are that they are adapted to the specifics of particular tasks and do not take into account the peculiarities of searching for an optimal route with the performance of certain actions at each point of the route.

The anytime algorithm can be particularly effective in situations where the decision-making time is commensurate with the time it takes for the initial conditions to find a solution to change. Indeed, in an emergency, waypoints may appear or disappear during the drone's flight and while searching for the optimal route, especially if this search takes a considerable amount of time. This can occur both due to

changes at the waypoints themselves and because several drones, which distribute the waypoints among themselves and duplicate each other's actions in case of failure, may be allocated for task execution. In such a scenario, some waypoints may disappear from the plan due to the actions of other drones or, conversely, be added to the task plan due to the failure of other drones that were scheduled for that task. In any case, a recalculation of the optimal route will be necessary.

The most closely related to the problem statement in this work are studies [18, 19], which are dedicated to constructing the optimal flight route of a UAV (drone) based on criteria of minimum energy, minimum cost, and maximization of territory coverage by onboard sensors. However, the task of passing through specific waypoints with the execution of certain tasks at them was not considered. Instead of working at waypoints, the tasks were aimed at operating in certain areas.

Thus, part of the unresolved problem is the methods of constructing and further improving the UAV flight trajectory during an emergency event by the criterion of minimum time. The trajectory is built by determining the order of passing through the waypoints, the set of which is clearly defined. A waypoint is considered a point where the drone must stop and perform a task: make an audio announcement, deliver cargo (material aid or information), etc.

The goal of the work is to develop algorithms for improving the reference trajectory based on varying the sequence of passing through the waypoints.

3. Drone Route Optimization Strategies

Let's accept the main assumptions of the study.

Assumption 1. The operational time of the drone at each waypoint is the same, as the execution of an identical set of actions is assumed at every point.

Assumption 2. The drone flies at a constant speed across all segments of the route, which is cruise speed, as it allows for energy savings, thereby increasing the available time and flight range.

Assumption 3. The drone starts from the base (the initial point of the route) and returns to the base after flying over all waypoints.

As a result of studying the characteristics of existing drones, it was found that, with a certain margin of error, the total energy a drone can use for a flight task can be related to the flight time through the following equation

$$E_{max} = k_t\, t + k_s(V)\, V\, t, \qquad (1)$$

where E_{max} is the total energy of the drone's battery, t is flight time, V is flight speed, k_t is battery energy expenditure coefficient depending on time, $k_s(V)$ is battery energy expenditure coefficient depending on the flight range (distance covered).

In turn, the coefficient $k_s(V)$ also depends on the flight speed. Up to a certain flight speed, the coefficient remains unchanged. However, after reaching a certain value, it starts to increase, leading to accelerated battery energy consumption. In simplified form, this dependency can be represented as

$$k_s(V) = \begin{cases} k_{s0} \text{ when } V \le V_{\text{cruise}} \\ k_{s0}(V/V_{\text{cruise}}) \text{ when } V > V_{\text{cruise}} \end{cases}, \qquad (2)$$

where k_{s0} is the battery energy expenditure coefficient depending on the flight range (distance covered) at a speed not exceeding cruise speed, V_{cruise} is cruise flight speed, at which the battery energy is used most efficiently.

In this study, we will assume that the flight speed does not exceed the cruise speed.

Considering the possible change over time t of the speed V and, accordingly, the coefficient $k_s(V)$, the expression for the drone's energy expenditure can be represented in the form

$$E(t) = \int_0^t (k_t + k_s(V(t))\, V(t)\, dt \quad , \qquad (3)$$

3.1. Exhaustive Search of Options

The simplest way to construct the optimal route is to enumerate all possible sequences of passing through the waypoints. However, in this case, for a route that includes N points, it will be necessary to check several options equal to (Fig. 1).

$$N! \approx 10^{Order},$$

$$Order = \sum_{i=1}^{N} \lg N.$$

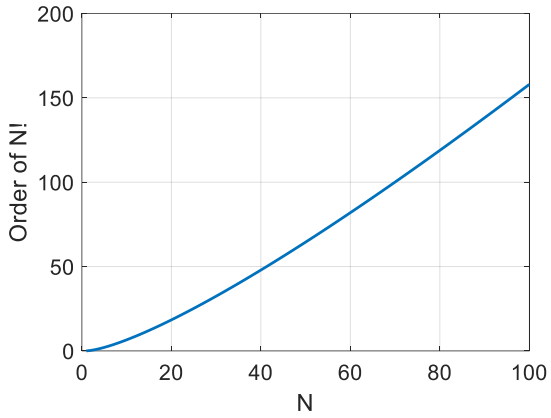

Figure 1: The order of magnitude equal to the number of options that need to be iterated through brute force for optimizing a route consisting of N points

In our case, brute force iteration of route options makes sense when the number of route points is no more than 10–15, depending on the power of the onboard computer. For a larger number of route points, more computationally efficient methods are required. However in doing so, one might have to pay for the speed of finding the route with the loss of some of its quality.

3.2. The Nearest Point Method

The simplest way to find a route that approximates the optimal is to find the nearest point that the drone has not yet visited. To do this, for the current point (x_0, y_0), we build an array of Euclidean distances to all other points (x_i, y_i) where the drone has not been yet:

$$D_e(0,i) = \sqrt{(x_0 - x_i)^2 + (y_0 - y_i)^2}.$$

We find the smallest:

$$D_{e\,min}(0) = \min_i D_e(0,i).$$

We remember the found index and remove the corresponding route point from further consideration. The procedure continues until all route points have been considered, i.e., an array of indices corresponding to the order of visiting the route points is constructed:

$$iRoute_2$$

The found route has a route length indicator D_2 much better than the length of the initial route D_1, which corresponded to the sequential numbers of the points visited, but in most cases, D_2 is still not optimal.

3.3. Refinement of the Reference Solution

Since the solution $iRoute_2$ is not optimal but approximates the optimal, it is subsequently taken as a reference solution that needs to be refined. The refinement of the reference route is carried out by varying the order of the route points in $iRoute_2$. For each point of the route $iRoute_2$, the following *algorithm for improving the reference route* is implemented:

1. The first route point is selected.
2. The route point in the order of visiting in $iRoute_2$ is moved to a new position.
3. The value of the route length D_3 is calculated.
4. If the condition $D_3 < D_2$ is met, then the variation is considered successful and fixed. If not, the result of the variation is ignored.
5. The next variation is selected—a new position for the transfer of the chosen reference route point (while all possible options are sequentially considered).
6. Steps 3–5 are repeated.
7. If all possible options for the new placement of the reference route point have been considered, a new reference route point is selected to search for a better place in the sequence $iRoute_2$.
8. Steps 2–7 are repeated.
9. The found value is considered the best.

Steps 1–9 can be repeated several times. This allows for increasing the depth of variations and further improving the quality of the optimal solution.

4. Algorithm Testing

In the graphs showing the route change during the improvement of the reference solution, the following notations with route length are adopted:

- D_1 is the route length for the initial order of route points (non-optimal)—indicated by a light blue line in the diagram.
- D_2 is the route length optimized by the nearest neighbor method—indicated by a thick blue line in the diagram.
- D_3 is the route length optimized by the method of varying the reference solution—indicated by a medium red line in the diagram.

As seen in Figs. 2–3, with a small number of route points ($N = 8$), the nearest neighbor method does not require further improvement. The solution is already optimal.

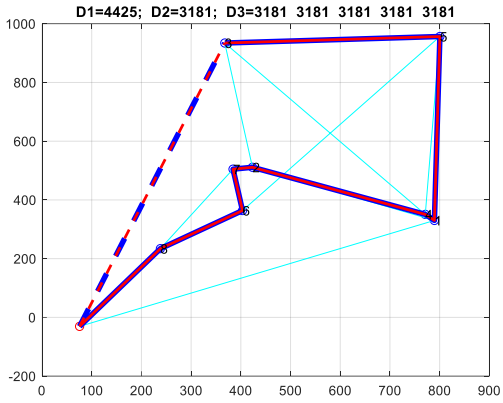

Figure 2: Route for $N = 8$

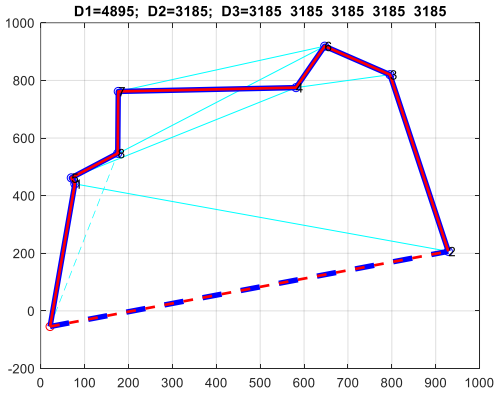

Figure 3: Route for $N = 8$

For 30 route points, varying the reference solution results in improvements, but only to a depth of no more than 2–3 complete cycles of variations (depending on the placement of the route points). As the dynamics of the change in D_3 (caption above Figs. 4–6) show, that increasing the depth of variations beyond 2–3 does not lead to an improvement in the D_3 indicator.

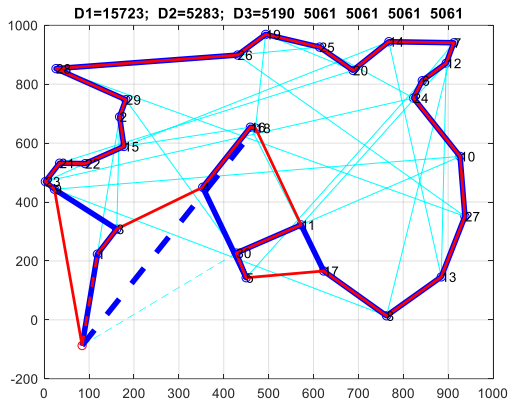

Figure 4: Route for $N = 30$

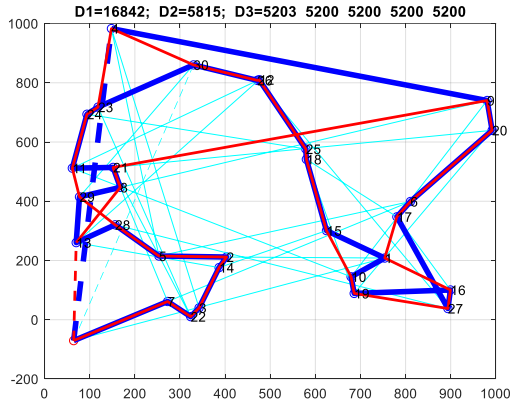

Figure 5: Route for $N = 30$

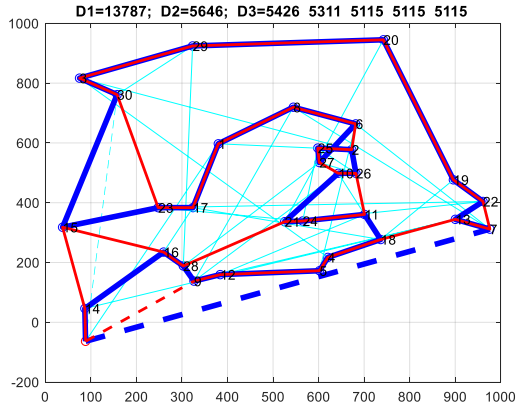

Figure 6: Route for $N = 30$

For 50 route points, varying the reference solution can lead to improvements at a variation depth of 3 to 5, also depending on the placement of the route points (Figs. 7–9).

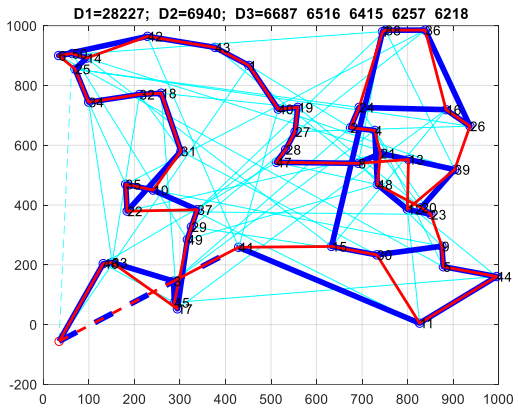

Figure 7: Route for $N = 50$

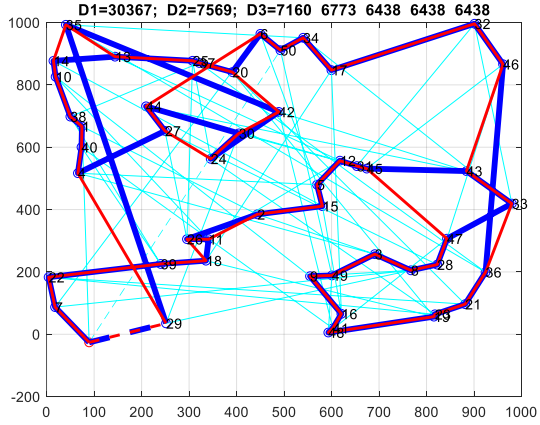

Figure 8: Route for $N = 50$

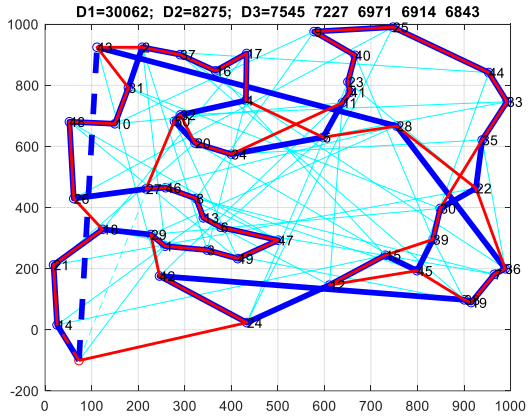

Figure 9: Route for $N = 50$

For 100 route points, varying the reference solution can lead to improvements at a variation depth of 3 to 5, just as for 50 points, depending on the placement of the route points (Figs. 10–12).

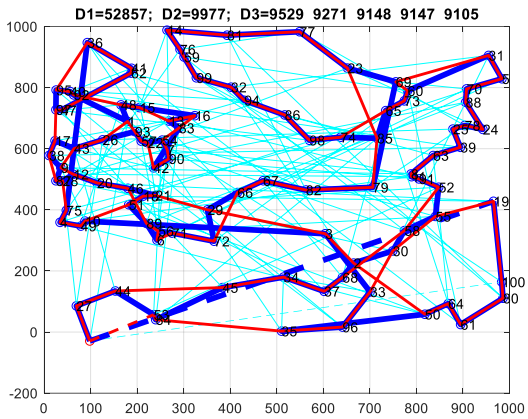

Figure 10: Route for $N = 100$

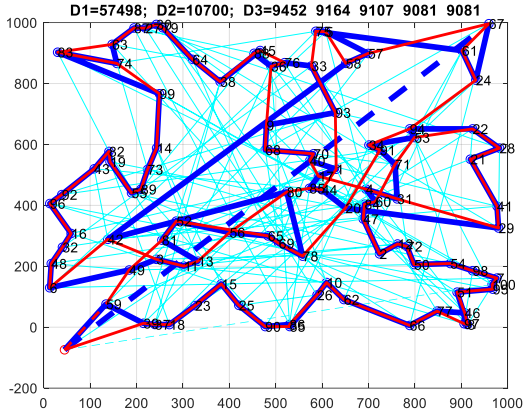

Figure 11: Route for $N = 100$

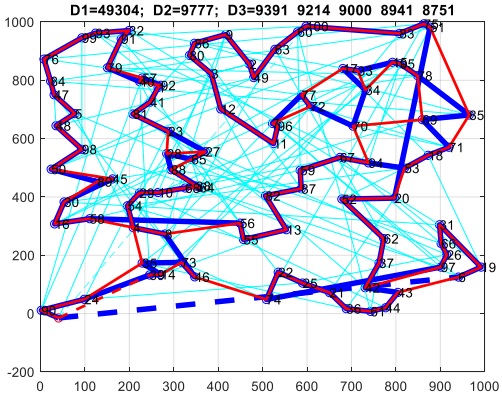

Figure 12: Route for $N = 100$

For 200 route points, the algorithm is also operational. However, the calculation time can increase to 10–20 seconds. Varying the reference solution can lead to improvements at a variation depth of 3 to 5, just as for 50 or 100 points, depending on the placement of the route points (Figs. 13–14).

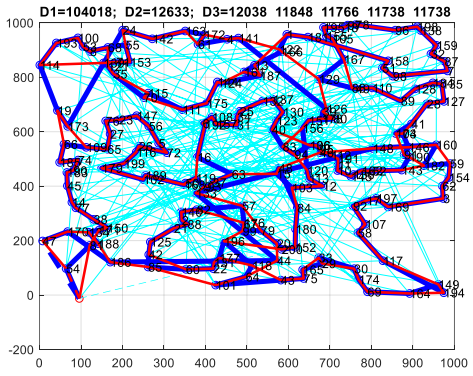

Figure 13: Route for $N = 200$

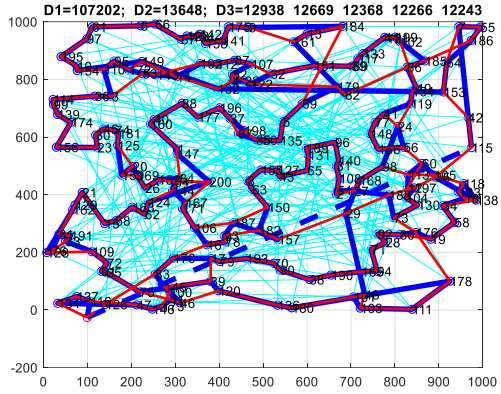

Figure 14: Route for $N = 200$

5. Conclusions

Thus, the work analyzed existing algorithms for constructing flight routes for drones that would have to fly through a given set of route points.

It was found that under conditions of possible changes to the set of route points during flight in the absence of communication with the control point, the use of methods for improving the reference trajectory (anytime algorithms) is most appropriate. For this, the construction of a reference trajectory using the nearest neighbor method and further refinement of the trajectory by varying the sequence of passing the route points were used.

The operability of the algorithm with low computational time costs on onboard computers for routes that include from 8 to 200 route points was tested.

Directions for further research include testing other possible approaches to varying reference trajectories, particularly through clustering route points.

References

[1] I. Kuzminykh, et al., Investigation of the IoT Device Lifetime with Secure Data Transmission, Internet of Things, Smart Spaces, and Next Generation Networks and Systems, vol. 11660 (2019) 16–27. doi: 10.1007/978-3-030-30859-9_2.

[2] V. Sokolov, et al., Method for Increasing the Various Sources Data Consistency for IoT Sensors, in: IEEE 9th International Conference on Problems of Infocommunications, Science and Technology (2023) 522–526. doi: 10.1109/PICST57299.2022.10238518

[3] V. Astapenya, et al., Last Mile Technique for Wireless Delivery System using an Accelerating Lens, in: 2020 IEEE International Conference on Problems of Infocommunications. Science and Technology (2020). doi: 10.1109/picst51311.2020.9467886.

[4] V. Sokolov, P. Skladannyi, A. Platonenko, Video Channel Suppression Method of Unmanned Aerial Vehicles, in: IEEE 41st International Conference on Electronics and Nanotechnology (2022) 473–477. doi: 10.1109/elnano54667.2022.9927105.

[5] M. Vladymyrenko, et al., Analysis of Implementation Results of the Distributed Access Control System. in: IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Tech-nology (2019). doi: 10.1109/picst47496.2019.9061376.

[6] R. Chen, Dijkstra's Shortest Path Algorithm and Its Application on Bus Routing, International Conference on Urban Planning and Regional Economy (2022). doi: 10.2991/aebmr.k.220502.058.

[7] R. Bellman, S. Dreyfus, Dynamic Programming, Princeton University Press (2021). doi: 10.2307/j.ctv1nxcw0f.

[8] M. Horsch, D. Poole, An Anytime Algorithm for Decision Making Under Uncertainty, Fourteenth Conference on Uncertainty in Artificial Intelligence (1998) 246–255.

[9] K. Li, N. Fard, A Novel Nonparametric Feature Selection Approach Based on

Mutual Information Transfer Network, Entropy 24(9) (2022) 1255. doi: 10.3390/e24091255.
[10] D. Szeszlér, Sufficient Conditions for the Optimality of the Greedy Algorithm in Greedoids, J. Comb. Optim. 44 (2022) 287–302. doi: 10.1007/s10878-021-00833-y.
[11] S. Bouamama, C. Blum, An Improved Greedy Heuristic for the Minimum Positive Influence Dominating Set Problem in Social Networks, Algorithms 14(3) (2021) 79. doi: 10.3390/a14030079.
[12] Y. Wang, Review on Greedy Algorithm, Theor. Nat. Sci. 14. (2023) 233-239. doi: 10.54254/2753-8818/14/20241041.
[13] S. Zilberstein, Using Anytime Algorithms in Intelligent Systems, AI Magazine 17 (3) (1996) 73–83. doi: 10.1609/aimag.v17i3.1232.
[14] J. Grass, S. Zilberstein, Anytime Algorithm Development Tools, SIGART Bulletin. 7(2) (1996) 20–27. doi: 10.1145/242587.242592.
[15] V. Shevchenko, Optimization Modeling in Strategic Planning, TsVSD NUOU (2011).
[16] F. Asadova, et al., A Survey of Usage of Anytime Algorithm in Fault Detection in Cloud Systems. 21st World Symposium on Applied Machine Intelligence and Informatics (2023) 69–74. doi: 10.1109/SAMI58000.2023.10044521.
[17] V. Lymperakis, A. Panagopoulos, Buggy Pinball: A Novel Single-point Meta-heuristic for Global Continuous Optimization, Artificial Intelligence and Soft Computing, LNAI 13589 (2022) 264–276. doi: 10.1007/978-3-031-23480-4_22.
[18] V. Shevchenko, et al., Models and Information Technologies of Coverage of the Territory by Sensors with Energy Consumption Optimization, Mathematical Modeling and Simulation of Systems, LNNS 344 (2022) 17–30. doi: 10.1007/978-3-030-89902-8_2.
[19] V. Shevchenko, et al., A Method for Maximum Coverage of the Territory by Sensors with Minimization of Cost and Assessment of Survivability, Appl. Sci. 12(6) (2022) 3059. doi: 10.3390/app12063059.