







Synthesis of Trajectory Planning Algorithms Using Evolutionary Optimization Algorithms

Dmitry Malyshev¹ , Vladislav Cherkasov¹ , Larisa Rybak¹ ,
and Askhat Diveev^{2,3} 

¹ Belgorod State Technological University named after V.G. Shukhov,
Belgorod, Russia
rlbgtu@gmail.com

² Federal Research Center “Computer Science and Control”
of Russian Academy of Sciences, Moscow, Russia

³ RUDN University, Moscow, Russia

Abstract. The article considers the problem of planning the optimal trajectory of a delta robot. The workspace of the robot is limited by the range of permissible values of the angles of the drive revolute joints, interference of links and singularities. Additional constraints related to the presence of obstacles have been introduced. Acceptable values of the robot’s input coordinates are obtained based on the inverse kinematics, taking into account the constraints of the workspace, represented as a partially ordered set of integers. For the given initial and final coordinates, a randomly generated family of trajectories belonging to a valid set is obtained. Optimization of each of the trajectories of the family based on evolutionary algorithms is performed. The optimization criterion is a function proportional to the duration of movement along the trajectory. The results of modeling are presented.

Keywords: Trajectory planning · Genetic algorithm · Grey Wolf optimization · Particle swarm optimization

1 Introduction

Robot trajectory planning is essential for avoiding various obstacles and obtaining the optimal path in terms of various criteria. Currently, there are a number of methods that can be used for trajectory planning. Some of the well-known methods are based on route networks, including the method based on the application of Visibility Graphs [1]. An alternative method for determining routes is based on the use of Generalized Voronoi Diagrams [2]. Thus, the method of uninformed search allows implementing breadth-first search, depth-first search, search by cost criterion [3]. Heuristic pathfinding algorithms are designed to quickly find a

This work was supported by the state assignment of Ministry of Science and Higher Education of the Russian Federation under Grant FZWN-2020-0017.

route in a graph by propagating towards more promising vertices [4,5]. In recent years, a number of methods for trajectory planning have been proposed. A two-stage method for planning the trajectory of two mobile manipulators for joint transportation in the presence of static obstacles is considered in [6]. At the first stage of path planning in progress, the shortest possible path between the initial and target configuration is executed in the workspace. The second stage consists in calculating a sequence of time-optimal trajectories for passing between consecutive points of the path, taking into account non-holonomic constraints and maximum permissible joint accelerations. A new method of spatial decomposition is presented in [7]. It was applied to define a space for trajectory planning, called RV-space. In [8], the method of directed reachable volumes (DRVs) is presented, which makes it possible to obtain the area taking into account constraints on the positions of the robot's links and end-effector. The current work considers the application of evolutionary and bio-inspired algorithms for planning the trajectory of a delta robot, taking into account the limitations of its workspace.

2 Setting an Optimization Problem

The delta robot [9] with 3 degrees of freedom is showed on Fig. 1. The end-effector of delta robot is the center P of the moving platform with x_P, y_P, z_P coordinates.

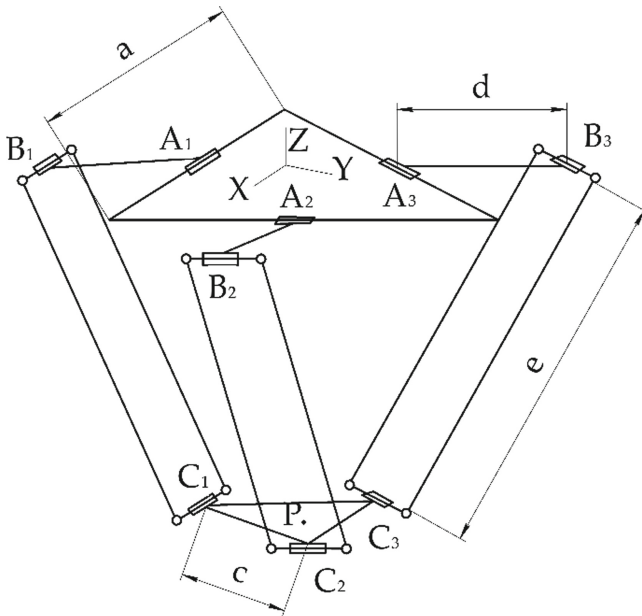


Fig. 1. Structure of the delta robot

The input coordinates for a delta robot are the angles θ_i of rotation of the drive revolute joints A_i . Inverse kinematics allows to transfer the described constraints of the workspace to the space of input coordinates. Inverse kinematics [10]:

$$\theta_i = 2 \tan^{-1} \left(\frac{-F_i \pm \sqrt{E_i^2 + F_i^2 + G_i^2}}{G_i - E_i} \right) \quad (1)$$

where E_i , F_i and G_i defined as $E_1 = 2d \left(y_P + \frac{a-2c}{2\sqrt{3}} \right)$, $F_1 = 2z_P d$,

$$\begin{aligned} G_1 &= v + \left(\frac{a-2c}{2\sqrt{3}} \right)^2 + 2y_P \left(\frac{a-2c}{2\sqrt{3}} \right) - e^2, E_2 = -d \left(\sqrt{3} \left(x_P + \frac{2c-a}{4} \right) + y_P + \frac{2c-a}{4\sqrt{3}} \right), \\ F_2 &= 2z_P d, G_2 = v + \left(\frac{2c-a}{4} \right)^2 + \left(\frac{2c-a}{4\sqrt{3}} \right)^2 + 2 \left(x_P \left(\frac{2c-a}{4} \right) + y_P \left(\frac{2c-a}{4\sqrt{3}} \right) \right) - e^2, \\ F_3 &= 2z_P d, G_3 = v + \left(\frac{2c-a}{4} \right)^2 + \left(\frac{2c-a}{4\sqrt{3}} \right)^2 + 2 \left(y_P \left(\frac{2c-a}{4\sqrt{3}} \right) - x_P \left(\frac{2c-a}{4} \right) \right) - e^2, E_3 \\ &= d \left(\sqrt{3} \left(x_P - \frac{2c-a}{4} \right) - y_P - \frac{2c-a}{4\sqrt{3}} \right), v = x_P^2 + y_P^2 + z_P^2 + d^2 \end{aligned}$$

An arbitrary trajectory can be represented as a set of movements (steps), during which the revolute joint drives operate at a constant angular velocity and the movement in the space of input coordinates is rectilinear. In order to reduce the duration of such small movements, the highest of the angular speeds of the drives at each step should correspond to the maximum possible. The duration of a move is proportional to the sum of individual steps defined in accordance with the Chebyshev metric:

$$t = \frac{1}{\omega_{\max}} \sum_{i=1}^n \rho_i \quad (2)$$

where $\rho_i = \max_{j \in \{1,2,\dots,m\}} |\theta_{i,j} - \theta_{i-1,j}|$ - according to Chebyshev distance between the points of the beginning of $C_{i-1}(\theta_{i-1,1}, \theta_{i-1,2}, \dots, \theta_{i-1,m})$ and end with $C_{i-1}(\theta_{i,1}, \theta_{i,2}, \dots, \theta_{i,m})$ i -th step; m is the number of the input coordinates (for the Delta robot $m = 3$); ω_{\max} is the maximum angular velocity of the drive revolute joints. However, the direct application of this indicator as a criterion function for optimizing the trajectory is impractical, since the Chebyshev metric introduces significant ambiguity. On the other hand, using the criterial “usual” length of the trajectory (the sum of the Euclidean lengths of all steps) as a criterion function also not allowed. The duration of movement in this case may be far from optimal as a result. Therefore, it is proposed to supplement the criterion function with a Euclidean metric taken with a small weighting coefficient ϵ :

$$F = \sum_{i=1}^n \left(\max_{j \in \{1,2,\dots,m\}} |\theta_{i,j} - \theta_{i-1,j}| + \epsilon \sqrt{\sum_{j=1}^m (\theta_{i,j} - \theta_{i-1,j})^2} \right) \rightarrow \min \quad (3)$$

The approach based on the application of a criterion function of this type was successfully tested by the authors to optimize the 3-RPR mechanism’s trajectory

[11]. Optimization should be carried out with constraints on the size of the workspace. In the framework of previous works, the authors proposed to use the representation of the workspace in the form of a partially ordered set of integers A_P [12]. Therefore, checking the optimization constraint consists of two steps.

The First Stage. Definition of the Set B of Trajectory Coordinates in the Space of Integers

For this purpose, an algorithm based on a modification of the algorithm is developed Bresenham’s algorithm [13], which assumes that the trajectory is represented as a polyline consisting of many segments. In [14], a modification of the algorithm was proposed for the 3D case, but the coordinates of the beginning and end of the segments belong to the space of integers, which leads to a displacement of the trajectory segment and the set B (Fig. 2). Cells that intersect the orthosis are highlighted in red for coordinates represented as integers, yellow for coordinates represented as real numbers, and orange for both cases. As you can see from the figure, using integer coordinates does not allow you to accurately determine the set B .

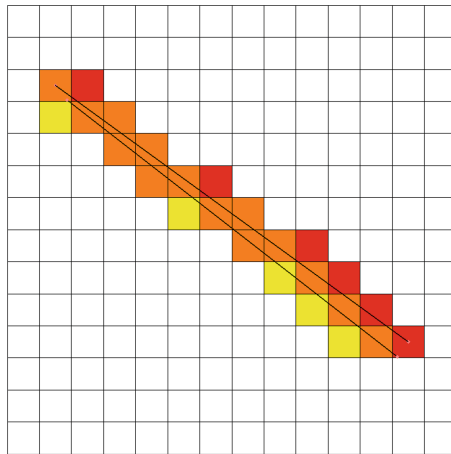


Fig. 2. Offset of the trajectory segment depending on the source data

Modifying the algorithm considering the original data that belongs to the 3D space of real numbers (coordinates of the beginning x_1, y_1, z_1 and end x_2, y_2, z_2 segments). In this case, the coordinates must correspond to the covering set of the workspace, represented as a partially ordered set of integers, respectively, they must be obtained taking into account the accuracy of the approximation Δ_j and the displacement k_j along the j coordinate axes by the formula

$$x_i = \frac{\theta_{i,1} + k_1}{\Delta_1}, \quad y_i = \frac{\theta_{i,2} + k_2}{\Delta_2}, \quad z_i = \frac{\theta_{i,3} + k_2}{\Delta_2} \tag{4}$$

The algorithm works as follows:

Input: $x_1, y_1, z_1, x_2, y_2, z_2$

- 1: $\delta_x = x_2 - x_1, \delta_y = y_2 - y_1, \delta_z = z_2 - z_1$
- 2: **if** $\delta_x = 0$ **then** $x_1 = \lfloor x_1 \rfloor, x_2 = \lfloor x_2 \rfloor$ **end if**
- 3: **if** $\delta_y = 0$ **then** $y_1 = \lfloor y_1 \rfloor, y_2 = \lfloor y_2 \rfloor$ **end if**
- 4: **if** $\delta_z = 0$ **then** $z_1 = \lfloor z_1 \rfloor, z_2 = \lfloor z_2 \rfloor$ **end if**
- 5: $B = B \cup \{\lfloor x_1 \rfloor, \lfloor y_1 \rfloor, \lfloor z_1 \rfloor\}$
- 6: **if** $\delta_x < 0$ **then** $x = \lfloor x_1 + 0.5 \rfloor - 0.5$ **else** $x = \lceil x_1 - 0.5 \rceil + 0.5$ **end if**
- 7: **if** $\delta_y < 0$ **then** $y = \lfloor y_1 + 0.5 \rfloor - 0.5$ **else** $y = \lceil y_1 - 0.5 \rceil + 0.5$ **end if**
- 8: **if** $\delta_z < 0$ **then** $z = \lfloor z_1 + 0.5 \rfloor - 0.5$ **else** $z = \lceil z_1 - 0.5 \rceil + 0.5$ **end if**
- 9: **while** $x \cdot \text{sign}(\delta_x) \leq x_2 \cdot \text{sign}(\delta_x)$ **do**
- 10: $x_Z = x + 0.5 \cdot \text{sign}(\delta_x), y_R = y_1 + \delta_y \cdot (x - x_1) / \delta_x, z_R = z_1 + \delta_z \cdot (x - x_1) / \delta_z$
- 11: **if** $(\lfloor y_R \rfloor \neq y_R - 0.5)$ **or** $(\text{sign}(y_R) = \text{sign}(\delta_y))$ **then**
- 12: $y_Z = \lfloor y_R \rfloor$
- 13: **else**
- 14: $y_Z = \text{sign}(y_R) \cdot \lfloor |y_R| \rfloor$
- 15: **end if**
- 16: **if** $(\lfloor z_R \rfloor \neq z_R - 0.5)$ **or** $(\text{sign}(z_R) = \text{sign}(\delta_z))$ **then**
- 17: $z_Z = \lfloor z_R \rfloor$
- 18: **else**
- 19: $z_Z = \text{sign}(z_R) \cdot \lfloor |z_R| \rfloor$
- 20: **end if**
- 21: $B = B \cup \{x_Z, y_Z, z_Z\}, x = x + \text{sign}(\delta_x)$
- 22: **end while**
- 23: **while** $y \cdot \text{sign}(\delta_y) \leq y_2 \cdot \text{sign}(\delta_y)$ **do**
- 24: $x_R = x_1 + \delta_x \cdot (y - y_1) / \delta_y$
- 25: **if** $\lfloor x_R \rfloor \neq x_R - 0.5$ **then**
- 26: $x_Z = \lfloor x_R \rfloor, y_Z = y + 0.5 \cdot \text{sign}(\delta_y), z_R = z_1 + \delta_z \cdot (x - x_1) / \delta_z$
- 27: **if** $(\lfloor z_R \rfloor \neq z_R - 0.5)$ **or** $(\text{sign}(z_R) = \text{sign}(\delta_z))$ **then**
- 28: $z_Z = \lfloor z_R \rfloor$
- 29: **else**
- 30: $z_Z = \text{sign}(z_R) \cdot \lfloor |z_R| \rfloor$
- 31: **end if**
- 32: $B = B \cup \{x_Z, y_Z, z_Z\}$
- 33: **end if**
- 34: $y = y + \text{sign}(\delta_y)$
- 35: **end while**
- 36: **while** $z \cdot \text{sign}(\delta_z) \leq z_2 \cdot \text{sign}(\delta_z)$ **do**
- 37: $x_R = x_1 + \delta_x \cdot (z - z_1) / \delta_z, y_R = y_1 + \delta_y \cdot (z - z_1) / \delta_z,$
- 38: **if** $\lfloor x_R \rfloor \neq x_R - 0.5$ **and** $\lfloor y_R \rfloor \neq y_R - 0.5$ **then**
- 39: $x_Z = \lfloor x_R \rfloor, y_Z = \lfloor y_R \rfloor, z_Z = z + 0.5 \cdot \text{sign}(\delta_z)$
- 40: $B = B \cup \{x_Z, y_Z, z_Z\}$
- 41: **end if**
- 42: $z = z + \text{sign}(\delta_z)$
- 43: **end while**

The algorithm for determining the coordinates of the trajectory in the space of integers for a 2D trajectory assumes the exclusion of the z dimension. Otherwise, it is similar to the algorithm given above for the 3D case.

The Second Stage. Checking Whether the Resulting set B Belongs to the Workspace Set A_θ

Thus the optimization constraint condition has the form

$$B_i \subset A_\theta, i \in 1, \dots, n \quad (5)$$

where n is the number of segments that make up the trajectory.

Thus, the optimization problem looks like this.

- parameters: coordinates of intermediate points of the trajectory $x_i, y_i, z_i, i \in 1, \dots, (n - 1)$. For a delta robot, the coordinates are the rotation angles of the drive revolute joints, i.e. $[(x_i y_i z_i)]^T = [(\theta_{i,1} \theta_{i,2} \theta_{i,3})]^T$.
- parameter change range: overall dimensions of the workspace in the space of input coordinates $\theta_{i,j} \in [\theta_{j,min}; \theta_{j,max}]$.

It should be noted that the optimization parameters change in the space of real numbers. The transition to the space of integers to calculate the cells checked at the second stage is carried out using the formula (4) and the modified Bresenham's algorithm.

- criterion: the function F calculated by formula (3).
- constraint: condition (5).

To increase the efficiency of optimization in the presence of obstacles, we transfer the optimization constraint to the criterion function

$$F' = F + \sum_{i=1}^n \left(\vartheta_i \left(p_1 \sqrt{\sum_{j=1}^m (\theta_{i,j} - \theta_{i-1,j})^2} + p_2 \right) \right) \rightarrow \min \quad (6)$$

where p_1, p_2 are the penalty coefficient, and ϑ_i is the Heaviside function:

$$\vartheta_i = \begin{cases} 0, & \text{if } B_i \subset A \\ 1 & - \text{otherwise} \end{cases} \quad (7)$$

3 Algorithms for a Path Optimization

The choice of algorithms is justified by their efficiency and high level of applicability to a number of different problems. However, the authors do not conclude that these algorithms are better than other evolutionary algorithms for solving this particular problem. The purpose of this investigation is an initial assessment of the applicability of some of the most widely used evolutionary and bio-inspired algorithms for optimizing a trajectory within a workspace represented as a partially ordered set of numbers. This creates the prerequisites for further in-depth research, including a comparative analysis of the application of a larger number of algorithms for this problem and the selection of their parameters.

We apply the following evolutionary and bio-inspired algorithms to solve optimization problem.

3.1 Genetic Algorithm (GA)

The basic principles of GA were first rigorously formulated by Holland [15]. The GA works with a population of “individuals”, each representing a possible solution to a given problem. Genetic algorithms are widely applied, including for the synthesis of a control system for robots [16], for planning the trajectory of collaborative robots [17]. We use a modification of the genetic algorithm described earlier in [18]. To speed up the algorithm, we apply parallel computing (Fig. 3). Dashed lines indicate areas where calculations are performed simultaneously.

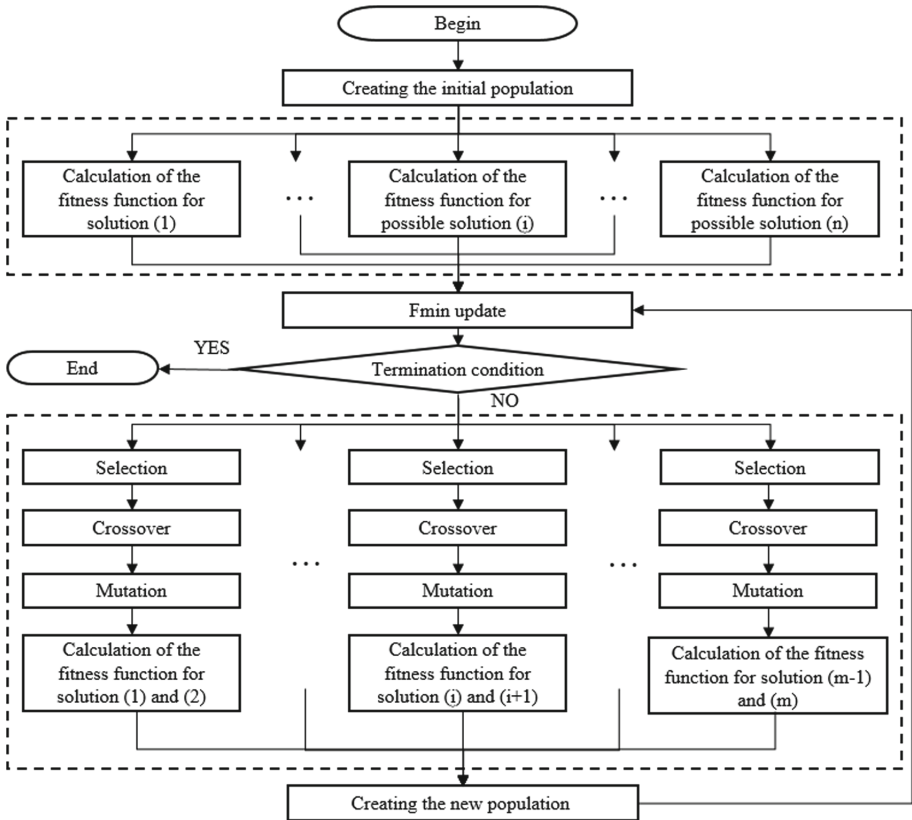


Fig. 3. A genetic algorithm using parallel computing

3.2 Particle Swarm Algorithm (PSO)

Particle Swarm Optimization (PSO) is a widely applied two-component swarm-based evolutionary optimization method [19, 20]. The particle swarm algorithm solves the problem by having a population of potential solutions, here called

particles, and moving these particles in the search space according to a simple mathematical formula over the position and velocity of the particle. The movement of each particle depends on its local best-known position, but is also directed to the best-known positions in the search space, which are updated as other particles find better positions. This is expected to move the swarm towards better solutions [18]. New position of the s_i particle At time t is determined by the vector of its coordinates q_i , and its velocity by the vector v_i :

$$q_{i,t+1} = q_{i,t} + v_{i,t+1}; \quad (8)$$

$$v_{i,t+1} = \alpha v_{i,t} + P_n[0; \beta] \times (q_{G,t} - q_{i,t}) + P_n[0; \gamma] \times (q_{P,t} - q_{i,t}) \quad (9)$$

where P_n [a; b] is an n -dimensional vector of pseudorandom values uniformly distributed over the interval [a,b]; $q_{G,t}$ is the coordinate vector of the best particle in the group; $q_{B,t}$ is the coordinate vector of the best in population particle; α , β , γ are free parameters of the algorithm with the following recommended values: $\alpha = 0.7$, $\beta = 1.4$, and $\gamma = 1.4$ according to [18]. To speed up the work, parallel calculations are applied in the same way when determining the new position of particles and the value of the criterion function in the new position.

3.3 Grey Wolf Algorithm (GWO)

Algorithm The Grey Wolf Optimization (GWO) algorithm [21] shows its reliability in solving real optimization problems where the objective function is not linear. The study in [21] shows that the PSO and GWO algorithms show better results in comparison with a number of other algorithms. The paper [22] presents a method for optimal trajectory generation (OTG) for creating a simple and error-free continuous motion along a trajectory with fast convergence using the GWO method. The authors of [23] compared the GA, PSO, and GWO algorithms for optimizing efficient hybrid robot control for controlling the foot trajectory of a robot during walking. The results showed that the GWO algorithm performs more efficiently and quickly at similar torques for configuring a hybrid controller based on LQR (Linear quadratic regulator) and PID (proportional–integral–derivative controller) than other traditional algorithms. Based on these works, newer methods for controlling robot navigation were also developed, which uses a hybrid concept of using the GWO algorithm and the artificial potential field (APF) method for planning the trajectory of a mobile robot [24].

We apply a modification of the GWO algorithm described earlier in [25] using parallel calculations to modify the parameters of a possible solution and determine the value of the fitness function.

4 Numerical Results

The problem of determining the workspace A_P for a delta robot in the space of output coordinates is considered by the authors in [26]. The constraints of the workspace A_P are transferred from the space of output coordinates x_P , y_P , z_P

to the space of input coordinates θ_i using the solution of the inverse kinematics (1). The workspace A_θ in the input coordinate space is similarly represented as a partially ordered set of integers after the transfer. An object was added as an obstacle with a parallelepiped shape (Fig. 4a). Elements of the covering set of the workspace in the output coordinate space (Fig. 4b) belonging to the obstacle were excluded $A'_P = A_P \setminus C$. The transfer of constraints $A'_P \rightarrow A_\theta$ (Fig. 4c) was performed for the updated set A'_P .

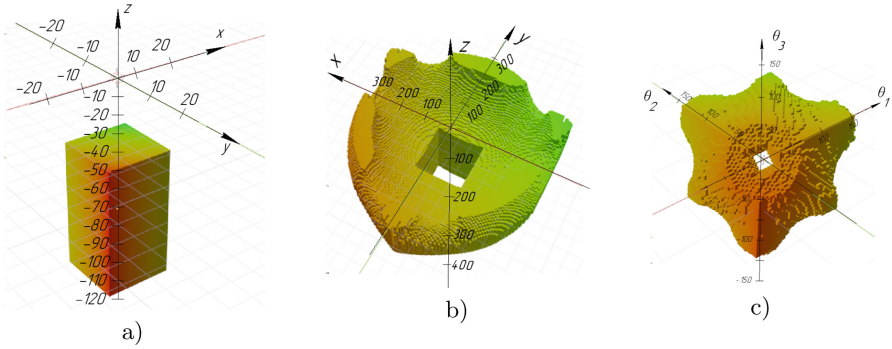


Fig. 4. Additional boundaries, related to the overall dimensions of the obstacle: a) obstacle C ; b) Workspace set A_P ; c) A'_P ; c) A_θ

We perform trajectory optimization using the above-mentioned algorithms for the 2D and 3D case of forming a trajectory inside the workspace of a delta robot, as well as for a randomly generated 2D contour with a large number of obstacles. A C++ software package has been developed for this purpose. Parallel computing is implemented using the OpenMP library. Visualization of 2D results is performed using developed Python scripts Python (Matplotlib and JSON libraries). Visualization of 3D results is performed by exporting an ordered set of integers describing the workspace in STL format and arrays of co-ordinates of trajectory points in JSON format, and then importing data in the Blender software package using developed Python scripts.

4.1 2D Case

Let's make a slice of the workspace A_θ of the delta robot in the space of input coordinates $\theta_1 \theta_2 \theta_3$, taking the angle $\theta_1 = 0$. In this case, the set A_θ will be 2D (Fig. 5). We assume that the starting point of the trajectory is $\theta_{1,2} = -70^\circ$, $\theta_{1,3} = 20^\circ$, the end point is $\theta_{n,2} = 50^\circ$, $\theta_{n,3} = 0^\circ$, and the number of vertices of the trajectory is $n = 3$. Accordingly, the number of optimization parameters $p = 2n = 6$. The weight coefficient $\epsilon = 0, 1$, the penalty coefficients: $p_1 = 5$, $p_2 = 500$. Parameters of the GA algorithm: the number of individuals in the initial population $H = 1000$, the number of generations $W = 250$, the number

of crossovers at each iteration $S_{GA} = 500$, the number of possible values of each of the parameters $g = 2^{25}$, the probability of mutation $p_m = 70\%$. Parameters of the GWO algorithm: $H = 1000$, $W = 250$, number of new individuals at each iteration $S_{GWO} = 1000$. Parameters of the PSO algorithm: $H = 1000$, $W = 250$, number of groups $G = 2$, values of free parameters $\alpha = 0.7$, $\beta = 1.4$, $\gamma = 1.4$. Optimization for each of the tests was performed in four stages. At the first stage, the range of parameters was changed to the ranges corresponding to the overall dimensions of the workspace for each of the coordinates. The parameter ranges at each subsequent stage were reduced by a factor of 10^2 . At the same time, the center of the ranges corresponded to the best result obtained at the previous stage. The optimization results are shown in Table 1. The PSO algorithm provides the best average value of the criterion function.

Table 1. Results table for the 2D case

Trials	GA	GWO	PSO	Trials	GA	GWO	PSO
1	160,089	160,197	160,044	6	160,107	160,378	160,044
2	160,821	160,145	160,044	7	160,007	160,457	160,044
3	160,344	160,340	160,045	8	160,060	160,397	160,042
4	160,048	160,228	160,040	9	160,198	160,212	160,045
5	160,123	160,070	160,044	10	160,471	160,189	160,044
				Avg. values	160,227	160,261	160,044

The obtained trajectories for all tests are almost identical. Examples of trajectories for the first and second tests are shown in Fig. 5. In the second test, the GWO algorithm obtained the trajectory with the largest value of the criterion function. This is clearly seen in Fig. 5b.

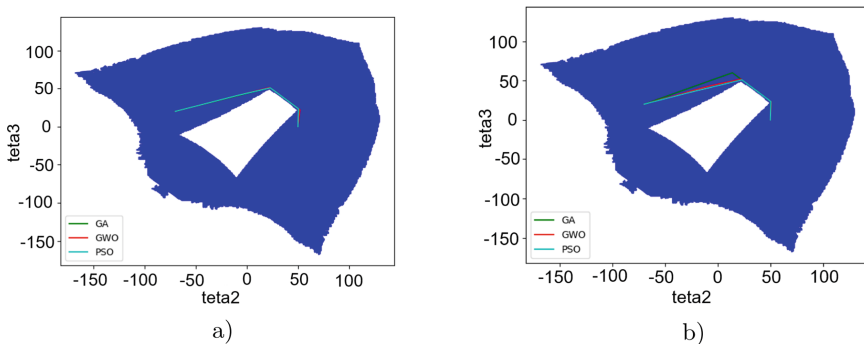


Fig. 5. Results of optimization of the test path: a) 1, b) 2.

The convergence graphs for each of the algorithms are shown in Fig. 6. The minimum value of the criterion function obtained as a result of optimization is applied as the reference value of the function. The minimum value of the criterion function in one of the tests was obtained using the GA algorithm, but in other tests, the PSO algorithm has better convergence rates.

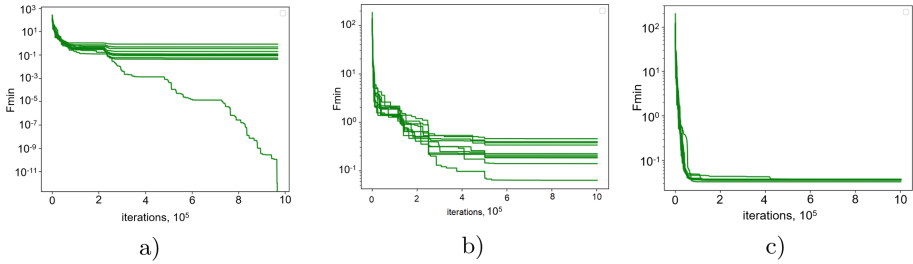


Fig. 6. Convergence of algorithms for planning a 2D trajectory: a) GA, b) GWO, c) PSO

4.2 3D Case

A computational experiment is performed, which consists in planning the trajectory inside the 3D workspace B_θ of a delta robot in the space of input coordinates, taking into account the obstacle, shown in Fig. 4c. Set the starting and ending points of the trajectory in the output coordinate space: $x_{p1} = 250$ mm, $y_{p1} = 250$ mm, $z_{p1} = -500$ mm, $x_{p2} = -270$ mm, $y_{p2} = -270$ mm, $z_{p2} = -450$ mm, and the number of vertices of the trajectory $n = 3$. Accordingly, the number of optimization parameters $p = 3n = 9$. Let's take the parameters of the algorithms $H = 250$, $W = 200$, $S_{GA} = 125$, and $S_{GWO} = 250$. The remaining parameters of the computational experiment coincide with the 2D case. Optimization for each of the tests is performed in four stages, similar to the 2D case. The optimization results are shown in Table 2. In this case, The GA algorithm showed the best average value of the criterion function.

Table 2. Results table for the 3D case

Trials	GA	GWO	PSO	Trials	GA	GWO	PSO
1	152,499	149,852	149,737	6	131,130	149,863	149,891
2	151,070	150,136	130,477	7	130,719	130,433	130,459
3	131,368	130,477	149,778	8	150,506	131,111	149,915
4	137,201	150,394	130,385	9	152,649	149,941	149,914
5	149,876	150,427	149,674	10	149,772	150,083	149,769
				Avg. values	143,679	144,272	144,000

The convergence graphs are shown in Fig. 7. The minimum value of the criterion function is obtained using the PSO algorithm.

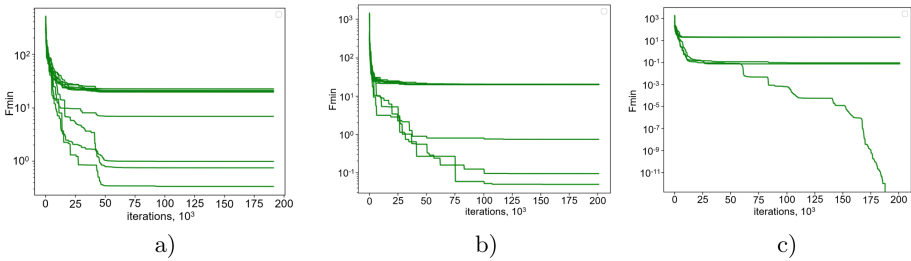


Fig. 7. Convergence of algorithms for planning a 3D trajectory: a) GA, b) GWO, c) PSO.

Figure 8 shows the trajectories for Test 3 inside the workspace. As can be seen from the figure, the PSO algorithm found only a local minimum of the criterion function for avoiding the obstacle.

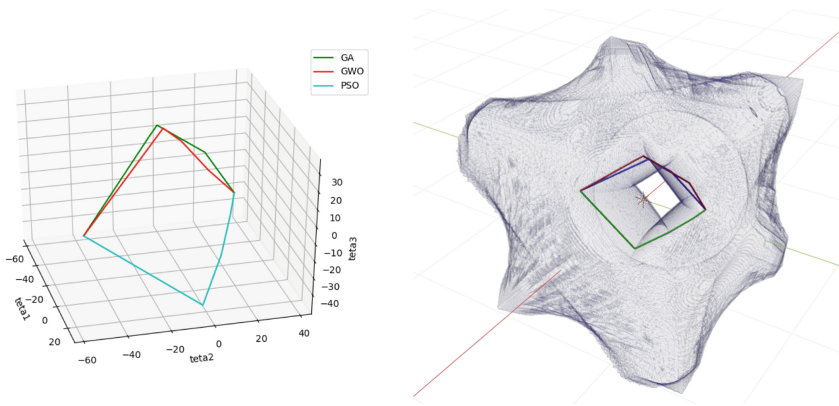


Fig. 8. Results of trajectory optimization

4.3 Trajectory Planning When There are a Large Number of Obstacles

In the first two cases, the trajectory was planned inside the workspace of the delta robot in the presence of a single obstacle. To test the algorithms on the problem of planning a trajectory with a large number of obstacles, a 2D domain was generated, similarly represented as an ordered set of integers. During the experiment, 10 tests were performed similarly for the following initial

data: $x_1 = -95$ mm, $y_1 = -95$ mm, $x_2 = 85$ mm, $y_2 = 85$ mm, the number of vertices of the trajectory $n = 7$. Accordingly, the number of optimization parameters $p = 2n = 14$. We assume the parameters of the algorithms $H = 2000$, $W = 1000$, $S_{GA} = 1000$, $S_{GWO} = 2000$, $p_m = 90\%$. The other parameters were not changed. Optimization for each of the tests was performed in two stages, rather than four. Figure 9 shows examples of the trajectories obtained as a result of optimization. In Fig. 9a, the path that allows you to avoid all obstacles is obtained only for the GA algorithm, in Fig. 9b and c - by the GA and GWO algorithms, in Fig. 9d - by all algorithms. Figure 10 shows an example of the convergence graph of the algorithms. As a result of performing 10 tests for each of the algorithms, the GA algorithm showed the best results GA (Fig. 10a), each time reaching a trajectory that allows to avoid all obstacles. The GWO algorithm (Fig. 10b) it allowed to exclude the interference with an obstacle in 4 cases, and the PSO algorithm-only in one case.

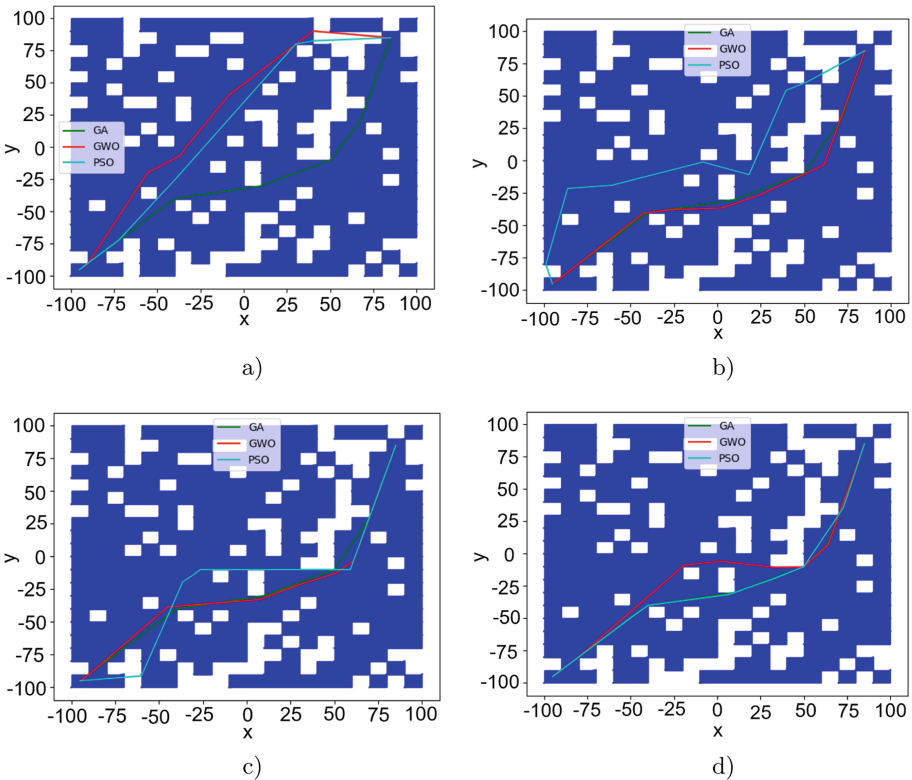


Fig. 9. The result of planning a trajectory in the presence of a large number of obstacles

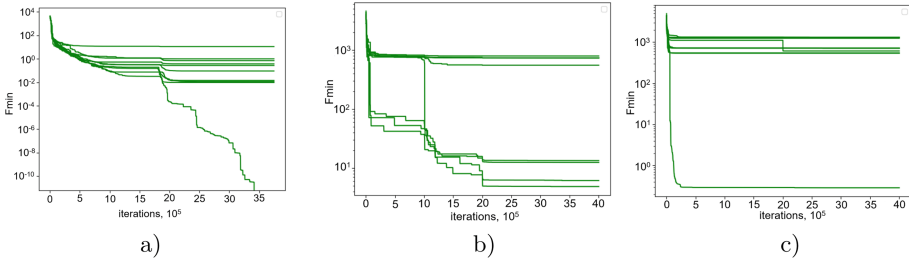


Fig. 10. Convergence of algorithms when planning a trajectory with a large number of obstacles: a) GA, b) GWO, c) PSO.

5 Conclusion

The application of heuristic algorithms made it possible to solve the problem of trajectory planning for both 2D and 3D domains represented as a partially ordered set of integers. The PSO algorithm showed better convergence rates for planning a trajectory within a 2D workspace of a robot with a single obstacle. In all other cases, the GA algorithm showed the better results. As part of future research in-depth research will be carried out, including a comparative analysis of the application of a larger number of algorithms for this problem and the selection of their parameters. Also, more experiments will be performed for accurate comparative evaluation of algorithms.

Acknowledgements. This work was supported by the state assignment of Ministry of Science and Higher Education of the Russian Federation under Grant FZWN-2020-0017.

References

1. El Khaili, M.: Visibility graph for path planning in the presence of moving obstacles. *IRACST - Eng. Sci. Technol. Int. J. (ESTIJ)* **4**(4), 118–123 (2014)
2. Choset, H., et al.: *Principles of Robot Motion-Theory, Algorithms, and Implementation*. MIT Press, Cambridge (2005)
3. Russell, S.J., Norvig, P.: *Artificial intelligence: a modern approach*. *Neurocomputing* **9**(2), 215–218 (1995)
4. Zeng, W., Church, R.L.: Finding shortest paths on real road networks: the case for A*. *Int. J. Geogr. Inf. Sci.* **23**(4), 531–543 (2009)
5. Korf, R.E.: Depth-first iterative-deepening. An optimal admissible tree search. *Artif. Intell.* **27**(1), 97–109 (1985)
6. Bolandi, H., Ehyaei, A.F.: A novel method for trajectory planning of cooperative mobile manipulators. *J. Med. Signals Sens.* **1**(1), 24–35 (2011)
7. Völz, A., Graichen, K.: An optimization-based approach to dual-arm motion planning with closed kinematics. In: *Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pp. 8346–8351. IEEE (2018)

8. McMahon, T., Sandstrom, R., Thomas, S., Amato, N.M.: Manipulation planning with directed reachable volumes. In: IEEE International Conference on Intelligent Robots and Systems 2017, pp. 4026–4033 (2017)
9. Clavel, R.: Conception d'un robot parallèle rapide à 4 degrés de liberté. Ph.D. Thesis, EPFL, Lausanne, Switzerland (1991)
10. Williams II, R.L.: The delta parallel robot: kinematics solutions. www.ohio.edu/people/williar4/html/pdf/DeltaKin.pdf. Accessed 9 Oct 2022
11. Khalapyan, S., Rybak, L., Malyshev, D., Kuzmina, V.: Synthesis of parallel robots optimal motion trajectory planning algorithms. In: IX International Conference on Optimization and Applications (OPTIMA 2018), pp. 311–324 (2018)
12. Rybak, L., Malyshev, D., Gaponenko, E.: Optimization algorithm for approximating the solutions set of nonlinear inequalities systems in the problem of determining the robot workspace. *Commun. Comput. Inf. Sci.* **1340**, 27–37 (2020)
13. Rogers, D.: Procedural Elements for Computer Graphics. McGraw-Hill (1985)
14. line3D - 3D Bresenham's (a 3D line drawing algorithm). <https://ftp.isc.org/pub/usenet/comp.sources.unix/volume26/line3d>. Accessed 9 Oct 2022
15. Holland, J.H.: Adaptation in Natural and Artificial Systems. MIT Press, Cambridge (1975)
16. Diveev, A.: Cartesian genetic programming for synthesis of control system for group of robots. In: 28th Mediterranean Conference on Control and Automation, MED 2020, pp. 972–977 (2020)
17. Zanchettin, A.M., Messeri, C., Cristantielli, D., Rocco, P.: Trajectory optimisation in collaborative robotics based on simulations and genetic algorithms. *Int. J. Intell. Robot. Appl.* **9**, 707–723 (2022)
18. Diveev, A.I., Konstantinov, S.V.: Evolutionary algorithms for the problem of optimal control. *RUDN J. Eng. Res.* **18**(2), 254–265 (2017)
19. Kennedy, J.; Eberhart, R.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, vol. IV, pp. 1942–1948 (1995)
20. Shi, Y.; Eberhart, R.C.: A modified particle swarm optimizer. In: Proceedings of IEEE International Conference on Evolutionary Computation, pp. 69–73 (1998)
21. Mirjalili, S., Mirjalili, S.M., Lewis, A.: Grey wolf optimizer. *Adv. Eng. Softw.* **69**, 46–61 (2014)
22. Choubey, C., Ohri, J.: Optimal trajectory generation for a 6-DOF parallel manipulator using grey wolf optimization algorithm. *Robotica* **39**(3), 411–427 (2021)
23. Sen, M.A., Kalyoncu, M.: Grey wolf optimizer based tuning of a hybrid LQR-PID controller for foot trajectory control of a quadruped robot. *Gazi Univ. J. Sci.* **32**(2), 674–684 (2019)
24. Zafar, M.N., Mohanta, J.C., Keshari, A.: GWO-potential field method for mobile robot path planning and navigation control. *Arab. J. Sci. Eng.* **46**(8), 8087–8104 (2021). <https://doi.org/10.1007/s13369-021-05487-w>
25. Diveev, A.I., Konstantinov, S.V.: Optimal control problem and its solution by grey wolf optimizer algorithm. *RUDN J. Eng. Res.* **19**(1), 67–79 (2018)
26. Malyshev, D., Rybak, L., Carbone, G., Semenenko, T., Nozdracheva, A.: Optimal design of a parallel manipulator for aliquoting of biomaterials considering workspace and singularity zones. *Appl. Sci.* **12**(4), 2070 (2022)