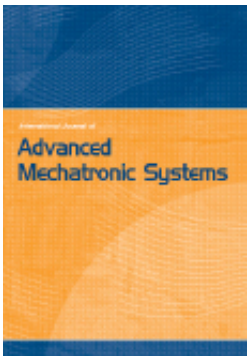


International Journal of Advanced Mechatronic Systems

ISSN online: 1756-8420 - ISSN print: 1756-8412
https://www.inderscience.com/ijamechs

Finding the optimal path in a 3D environment with predefined obstacles

Gabriel Mansour, Ilias Chouridis, Apostolos Tsagaris

DOI: 10.1504/IJAMECHS.2024.10063118

Article History:

Received:	04 June 2023
Last revised:	06 November 2023
Accepted:	04 December 2023
Published online:	25 March 2024

Finding the optimal path in a 3D environment with predefined obstacles

Gabriel Mansour

Department of Design and Structures,
Polytechnic School of the Aristotle University of Thessaloniki,
Thessaloniki, Greece
Email: mansour@auth.gr

Ilias Chouridis* and Apostolos Tsagaris

Department of Industrial Engineering and Management,
International Hellenic University,
Thessaloniki, Greece
Email: iliachour@iem.ihu.gr
Email: tsagaris@ihu.gr
*Corresponding author

Abstract: Robotics has substantially improved people's daily lives, especially industrial production and manufacturing. An offline programming method is proposed for robot's path planning in a 3D environment with obstacles. The purpose of this method is to find the shortest 3D path between two or more points avoiding obstacles. Two types of paths are created: in the first type, the shortest path between the points is created based on their input order; in the second type, the shortest path that connects the input points is formed. It is accomplished by using a hybrid algorithm that combines the ant colony optimisation algorithm with a genetic algorithm called the roulette wheel method. The proposed method takes into consideration the robot's capabilities and the variability of different environments, so that it can be effectively applied to a multitude of cases. The method has been tested and applied to real world industrial robots successfully.

Keywords: robotics; industrial robotics; hybrid algorithm; ant colony optimisation; genetic algorithm; offline programming; path planning; 3D environment; industrial robot navigation; mechatronic system.

Reference to this paper should be made as follows: Mansour, G., Chouridis, I. and Tsagaris, A. (2024) 'Finding the optimal path in a 3D environment with predefined obstacles', *Int. J. Advanced Mechatronic Systems*, Vol. 11, No. 1, pp.50–62.

Biographical notes: Gabriel Mansour is a Professor of Mechanical Engineering, the Head of Design and Structures Department, Polytechnic School of the Aristotle University of Thessaloniki. His research interest includes machine tools, machine tools foundation, CAD/CAM, robotics, vibration measurements, 3D measurements, reverse engineering, composite and nanocomposite materials. His publications are three books and more than 200 papers, in international scientific journals and conference proceedings.

Ilias Chouridis acquired his Diploma in Mechanical Engineering from the Polytechnic School of the Aristotle University of Thessaloniki. He is currently pursuing his PhD in Robotics and Mechatronics Systems Navigation in International Hellenic University. His research interest includes bio-inspired optimisation methods and evolutionary algorithms based on unmanned aerial vehicle path planning.

Apostolos Tsagaris is an Associate Professor in Robotics, CAD/CAM/CAE and Mechatronic Systems in the Department of Industrial Engineering and management at International Hellenic University of Thessaloniki, Greece and at this time he is the Head of the Department. He received his Bachelor's in Automation Engineer, MSc in Design of Interactive and Industrial Products and Systems, MSc in Mechatronics, MEd in Adult Education and MBA. He has published a book and more than 84 scientific papers at conferences and journals. He also holds a patent. Finally, he has participated in over 18 research projects.

1 Introduction

Robotics has benefited industrial manufacturing and has also proven able to execute dangerous and repetitive assignments smoothly. The industrial applications of robotics have been developing since 1960. The robots operated in fixed, unchangeable conditions, which was an especially important achievement for that time because they could successfully operate with great precision in unchangeable environments. Nowadays, as Malone et al. (2020) report, robots are becoming more adept at operating in unknown and human-centred environments thanks to sophisticated software and algorithms.

In robotics, the role of path planning is crucial. It is essential to find a collision free path so that the robot can navigate from a starting point to an end point without being damaged by obstacles. Usually, as Zafara and Mohanta (2018) mention, there are several paths, that meet the above goal, therefore some additional criteria are taking into consideration in the path planning process. Some of these may be the shortest distance path, the path's smoothness, the minimum energy consumption, or a combination of the above. The shortest distance with the least amount of time is the most frequently used criterion. The path planning methods are divided into two major categories, the classical approach and the heuristic approach.

Patle et al. (2019) pointed out that classical approaches were initially quite popular, because artificial intelligence methods had not been developed in those days. It has been noted that when a task is performed using a classical approach, either a result will be obtained or it will be confirmed that there is no result. The main downside of classical approaches is the increased computational cost and the inability to acclimate to environmental unpredictability. Therefore, in real-time applications, they are less adopted.

Mac et al. (2016) utterance that classical approaches tend to be trapped in some local minimal and fail to find the optimal solution, especially when obstacles are included. Heuristic approaches are used to overcome classical approaches weaknesses. The heuristic approaches adopt artificial intelligence techniques to solve the problem.

Due to the technological advances that have been made in the fields of mechatronics and computers, complex algorithms can be developed so that machines can adapt to a changeable environment. Up until now, industrial production was established based on the machine's capabilities, it was adjusted to the environment and allowed minimal variation. Nowadays, Andreu-Perez et al. (2017) alleged that the production can be integrated in an environment that already exists thanks to artificial intelligence's convergence with the robotics. The integration of artificial intelligence with robotics is used for path planning optimisation.

Floreano and Wood (2015) reviewed the capabilities of flying robots and the importance of path planning in their operations. Santos et al. (2020) studied the application of path planning to ground robots in agriculture, taking into consideration the constraints imposed by the robot composition or the type of terrain. Ma et al. (2019) researched path planning in autonomous underwater robots by using ant colony optimisation algorithm for finding the optimal path. Their model accumulates the path's length, the energy consumption, the collision risk and the steering window constraint. Among heuristic-based methods, there is a method called ant colony optimisation. Ant colony optimisation is extensively used for solving path planning problems. Ant colonies are highly organised, the ants interact with each other by using pheromones. Several optimisation problems can be solved by simulating their behaviour. After the first appearance of ant colonies optimisation algorithms were considerably developed by the researchers. Nonetheless, Ostfeld (2011) noticed that high computational power is required and sometimes are ineffective.

The real ants are navigating the environment by detecting the higher concentration of pheromone. The pheromone concentration is increased by other ants that deposited it as they wander in the environment. The pheromone deposition is continuous. In ant colony optimisation algorithm, Dorigo et al. (2006) proposed that a group of artificial ants build solutions to the optimised problem. They cooperate with each other to find the optimal solution by changing the pheromone value, like the real ants.

The travel salesman problem (TSP) is a famous non-deterministic polynomial (NP) time hard problem. The TSP consists of a known set of cities and the distances between a pair of cities. The salesman must visit each city once and then return to the starting city at the end by travelling the shortest distance. Li et al. (2008) mentioned that ant colony optimisation was mainly used to solve this problem.

As Melanie (1990) reported, John Holland contrived the genetic algorithms in the '60s. Holland with his students and colleagues at Michigan's university expanded the genetic algorithms. Holland's original objective was to study how adaptation happens in nature and to develop ways to use the same mechanisms in computer systems as opposed to evolution strategies and evolutionary programming that use algorithms to solve a specific problem.

Kwaśniewski and Kwaśniewski (2018) used genetic algorithms for finding the path on a 2D map with obstacles. Demir et al. (2021) developed a time optimisation model for path planning in an RRR robot by using genetic algorithms. Zhang et al. (2020) used an improved A* algorithm for path planning in a 2D environment, applying it to pathfinding in a game's map. Dai et al. (2019) proposed an algorithm that used the characteristics of the A* algorithm and the max-min ant system for path planning in a 2D environment. Brand et al. (2010) investigate the application of the ant colony optimisation algorithm to robots' path planning in a 2D dynamic environment. Reshamwala and Vinchurkar (2013) reviewed the variations of the ant colony optimisation algorithm in robots' path planning and compared three of the investigated methods. Zhang et al. (2021) presented a hybrid approach by combining the

enhanced ant colony system with a local optimisation algorithm for mobile robots' path planning. Chaari et al. (2012) proposed a hybrid algorithm for a 2D robot's path planning by combining an improved ant colony optimisation method with a genetic algorithm. Ravankar et al. (2017) proposed a knowledge sharing mechanism for multiple robots to achieve efficient path planning in a dynamic 3D environment. Liu et al. (2022) proposed an algorithm for optimal mission assignment and path planning in a 3D environment for multiple UAVs by using the adaptive genetic algorithm and the improved artificial bee colony method. Sanchez-Lopez et al. (2019) proposed a real-time path planning method for aerial robots in a complex, dynamic environment with obstacles. Wang et al. (2022) used a topological algorithm for multi-robot path planning in a complex 3D environment, aiming to reduce the robot's congestion in the environment while avoiding communication and coordination between robots.

In the real world, robots are operating in a 3D environment with several obstacles and challenges. A vital issue in robot's navigation and efficiency is path planning. An offline programming method for effective path planning of robot's end effectors and mechatronics systems is proposed in this paper. The method's purpose is to find the shortest collision-free path between two or more locations in a 3D environment with predetermined obstacles, similar to the real world.

There are two types of paths that the algorithm can generate. In the first type of path, also called the manual option, the algorithm searches for the shortest route that connects two or more points located anywhere in a 3D environment. The formed path is determined by the input order of the points. In the second type of path, also called the auto option, the algorithm finds the optimal path connecting a set of points regardless of their input order. In this manner, a complicated route with multiple target points can be optimised. As a result, the algorithm can optimise the path and sequence in which the robotic arm executes multiple assignments.

A hybrid algorithm that combines the ant colony optimisation algorithm with a genetic algorithm called the roulette wheel method is used to find the optimal path in a 3D environment with several obstacles. The proposed method takes into account the movement capabilities of the industrial robot as well as the characteristics and uniqueness of its operational space. Furthermore, if the requested route is short, the algorithm discovers the ideal path in a few iterations. In case the requested path is more complicated, the algorithm needs more iteration to determine the optimal and fine tuned parameters required.

Moreover, it is worth to mentioning that this method also contributes to the adaptation of a classical TSP solution to a more intricate and advanced 3D environment with obstacles and greater movement possibilities, suitable for real world applications.

A variety of simulations and tests were performed to evaluate the results of the proposed method which was also applied in a real world scenario by using an industrial robotic arm.

2 Proposed method

The proposed method was developed using MATLAB R2016a by MathWorks. To facilitate data management and avoid repeated calculations, the program was divided into two main parts: the Pre-processor and the Solver. In the pre-processor, the 3D environment is modelled and some basic and immutable data from the solver are calculated. in the solver, based on pre-processors data and some extra parameters for the ant colony optimisation and roulette wheel methods, the shortest path is calculated in two different ways. In the first way, called manual, the shortest path that connects the points is calculated according to the order of their input. In the second option, called auto, the program calculates the shortest path connecting the input points.

2.1 Pre-processor

2.1.1 Environment modelling

Industrial robotic arms, due to their flexibility, could be programmed to move their end effector in a three dimensional space. In order to find a path that corresponds to the real conditions and advantages of industrial robotic arms, instead of a 2D grid method, a 3D grid method was adopted. In the 3D grid method, the space is divided into several horizontal planes perpendicular to the Z axis. Each horizontal plane is divided into grids.

First of all, an initial O-XYZ coordinate system is defined. Based on this system, the coordinates of operating space are defined. The operating space is a 3D rectangular prism, the sides OB, OD and OE are defined based on the real operation space's dimensions. The definition of the point coordinates starts from the XOZ plane OBCD region and then continues across the Y-axis EFGHD region.

Each side consists of a certain number of nodes n along the X-axis, l along the Z-axis and m along the Y-axis. The numbers n, l and m determine the mesh density of the corresponding axis. Node numbering starts from the OBCD plane along the OX axis, with Y and Z coordinates fixed. After defining the coordinates of the first line, the Z coordinate is changed and the same process is repeated until all the coordinates of the nodes on the OBCD plane are precisely defined. These coordinates are entered into a 2D matrix named nodes. The nodes matrix consists of n*l*m rows and 4 columns. In each row is registered the number on the node and its X, Y and Z coordinates.

After creating the matrix of nodes, the matrix of elements of each smaller rectangular prism is created. Each 3D rectangle prism consists of 12 elements. Elements are a sequence of connecting nodes. Nodes are joined in pairs. These sequences are entered into a 2D matrix called elements. The elements matrix consists of (n – 1) * (l – 1) * (m – 1) * 12 rows and 3 columns. The number of each

element is registered in the first column, followed by the number of the first node and then the number of the second node. The correspondence between the element and its nodes starts with the left and right edge rectangles. It starts from the lower right node and goes clockwise, as shown in Figure 2(b). After the elements of parallelograms are registered, the elements of the compounds are registered, starting clockwise from the bottom to the top. Table 1 shows a transpose matrix example of Figure 2(a) elements registering, the different colours depict the different registering phases. The first row contains the number of each element, the second and third rows contain the number of the first and second element's nodes, respectively.

Figure 1 Path planning operating space (see online version for colours)

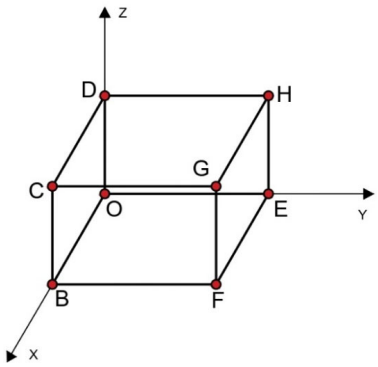
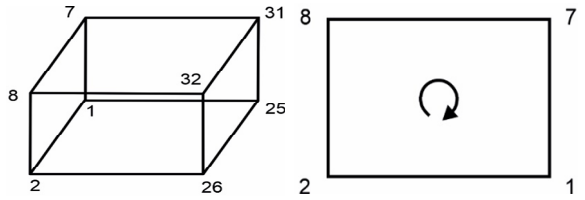


Figure 2 (a) Numbering of nodes of a 3D rectangular prism (b) Clockwise nodes registering (see online version for colours)



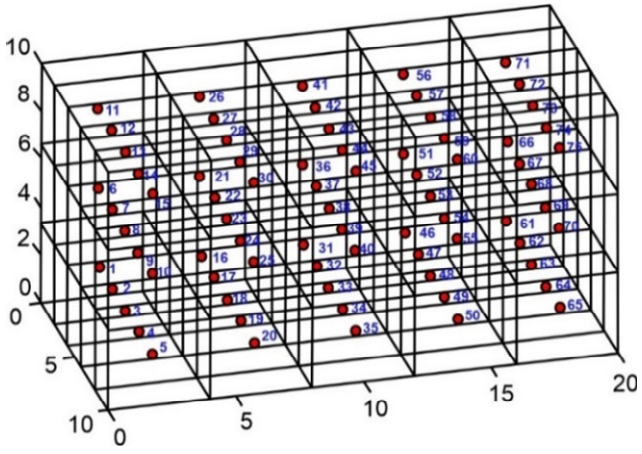
(a) (b)

Table 1 Transpose matrix of elements registering (see online version for colours)

1	2	3	4	5	6	7	8	9	10	11	12
1	2	8	7	25	26	32	31	1	2	8	7
2	8	7	1	26	32	31	25	25	26	32	31

After generating the 3D operating space, a set of movement points is created on which the robot can move. These points are located at the centre of gravity of each 3D rectangle prism and are defined by X, Y and Z coordinates. These points are directly related to the previous 3D modelling data. The method of their registration is the same as that of nodes. The number of points also represents the number of 3D rectangle prisms. They are stored in a 2D matrix named nodes2. The nodes2 matrix consists of (n – 1) * (l – 1) * (m – 1) rows and 4 columns. Each row contains the number of movement points and their X, Y and Z coordinates. Figure 3 represents the plotting results of Movement points and the 3D operating space.

Figure 3 3D operating space with movements points (see online version for colours)



2.1.2 Navigation

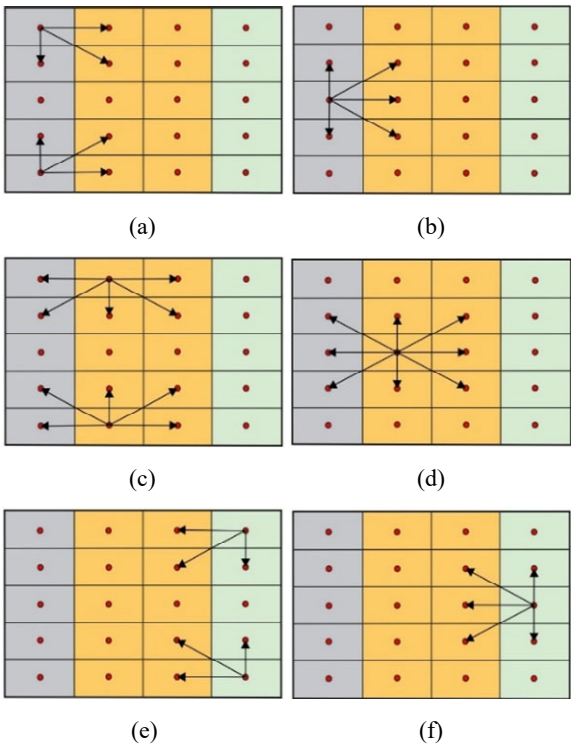
The robot can only move on the movements points, as a result, there are specific sequences of nearby points that can navigate from a definite position. Due to the fact that the process of finding the next feasible points from a current position is repeated several times in an ant colony optimisation algorithm, to avoid repeated calculation and calculation time waste, it is considered beneficial to create in advance a matrix that contains all the possible navigation choices for every movement point. The advantages of the robotic system are also taken into consideration in the possible choices. The first column of each row contains the number of the movement point at which the robot is located at a current time, then the numbers of all the next feasible points are registered in each column. During the algorithm's operation time, the matrix row for the given position is recalled.

The robot's movement choices within its working space are divided into three main areas, depending on its movement ability at the surrounding points. In Figure 4, these three areas can be distinguished by their colour. The first one is on the left (grey), the second is in the middle (yellow) and the third is on the right (green).

These three regions are subdivided according to the number of nearby movements points on the same horizontal plane. In Figures 4(a) and 4(b) for the left area, three individual cases can be distinguished, one for the top point with three possible movement's points, one for the bottom with three possible movement's points and one for the middle with five possible movement's points.

In Figures 4(c) and 4(d) for the left area, three individual cases can be distinguished, one for the top points with five possible movement's points, one for the bottom with five possible movement's points and one for the middle with eight possible movement's points.

Figure 4 The three main movement's areas, (a) movement points on the edge of the left region (b) movement points on the middle of the left region (c) movement points on the top and bottom of the middle region (d) movement points on the middle of the middle region (e) movement points on the edge of the right region (f) movement points on the middle of the right region (see online version for colours)

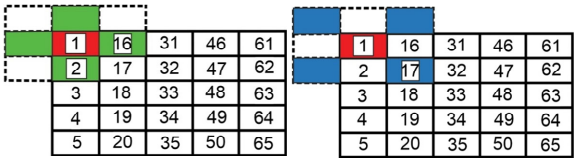


In Figures 4(e) and 4(f) for the left area, three individual cases can be distinguished, one for the top point with three possible movement's points, one for the bottom with three possible movement's points and one for the middle with five possible movement's points.

The industrial arm has a great degree of flexibility, in addition to the points shown in Figure 4, it can also move along the z-axis to the corresponding points of the planes above and below the reference plane, including the points above and below the current position point. These points are stored in a 2D matrix named mov. The mov matrix consists of (n – 1) * (l – 1) * (m – 1) rows and (8 + 1) * (l – 1) columns. For points where the number of allowed movement nodes is less than the number of columns, the remaining positions are filled with 0 until the row is completed.

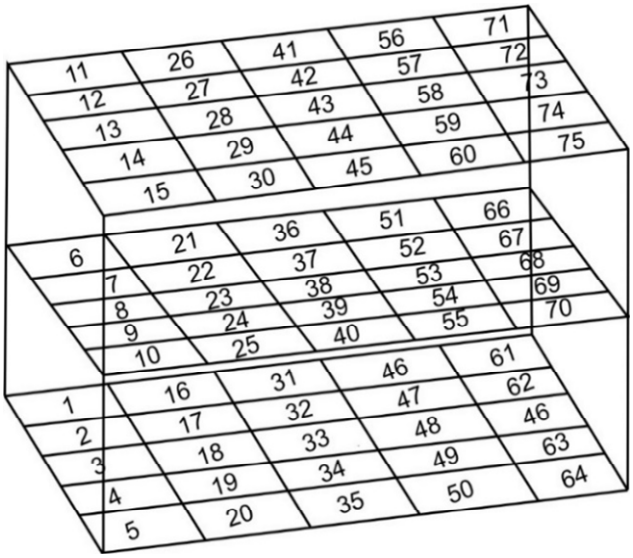
In every row, the first column contains the number of the movement point at which the robot is located at a current time, then the next feasible points are registered following a specific process. The registration starts on the reference plane and then expands along the Z-axis. Initially, the points that are perpendicular to the position point are recorded in a counter-clockwise direction, starting from the one below it, as shown in Figure 5(a). During the execution of this process, some vertical sections are encountered, in which there are not movement's points, such as those of the dotted lines. In this case, no point is recorded and the process continues to the next vertical section.

Figure 5 (a) Vertical sections recording (b) Diagonal sections recording (see online version for colours)



When the recording of the vertical points is completed, the recording of the diagonals begins in the same way, as shown in Figure 5(b). The recording of the diagonal elements starts from the lower right in a counter-clockwise direction. When some empty diagonal sections are encountered, as in the case of the vertical, no point is recorded and the process continues to the next diagonal section.

Figure 6 The three planes of 3D operating space depicted in Figure 4



After recording the points of the reference plane is completed, start recording the corresponding points on the parallel planes along the Z-axis. The points are recorded in columns parallel to the Z-axis and it takes place at all the points of the reference plane, including the point where the robot's end effector is located. The Z-axis point recording starts from the plane closest to the XOY plane and each point is recorded only once. If the number of filled row's columns is less than the number of matrices' columns, the remaining columns are filled with 0. Each column contains the number of a moving point.

Figure 6 shows the 3D operation environment of Figure 3. This operation environment consists of 3 planes. In this example, the robot is located at point number 1. The reference plane also happens to be the closest to the XOY plane. Table 2 shows the point registration in the first row of the mov matrix. The matrix will consist of 75 rows and 27 columns.

Table 2 A part of the example's first row

1	2	16	17	6	11	7	12	21	26	22	27

2.1.3 Obstacles

The presence of obstacles in the 3D operation space prohibits the robot from moving at certain points and alternative routes are explored. Two main types of obstacles are introduced: the single point obstacle and the continuous points obstacle. Any physical space can be modelled with these two obstacle types. The obstacles can be used to transform the environment into any 3D shape, so that it is not necessarily rectangular. In this way, the simulation converges as closely as possible to the real conditions.

The existence of obstacles is registered in a 2D matrix called obs. The obs matrix consists of (n – 1) * (l – 1) * (m – 1) rows and 2 columns. Each line has information on whether or not movement is allowed at a specific movement point. In the first column is registered the number of movement point and in the second is the number 0, or 1. The number 1 allows the movement to this movement point and the number 0 indicates the existence of the obstacle at this point. Different obstacles can be introduced for each operation space. Obstacles are plotted as black dots, as shown in Figure 7.

2.1.4 Safety value

Due to the complexity of the 3D operation space, the high number of potential paths and the existence of the obstacles, the safety value of movement points is introduced. Wang et al. (2019) proposed the calculation of safety value to improve the understanding of the space and obstacles when forming the path. The safety value is calculated as:

$$S(i, j, k) = \frac{v - u}{v} \tag{1}$$

where v indicates the total number of movement points at the current position point (i, j, k) and u indicates the number of obstacles at the same position point. The safety value is used in the solver. Because the safety value as well as the movement points for navigation need to be calculated several times in the ant colony optimisation algorithm, the are calculated in the pre-processor and stored in a 2D matrix called S. The matrix S consists of (n – 1) * (l – 1) * (m – 1) rows and 2 columns. In each row is registered the number of movement point and the result of safety value calculation.

2.1.5 Pre-processor txt file extraction

Before the pre-processor operation is finished, a txt file is created with the necessary data for running the Solver independently. The data entered in the txt file are the nodes, elements, nodes2 and obs matrices to model the 3D operation space and the number of nodes on the Z axis, as well as the mov and S matrices to define the ant movement and the safety value, respectively. All the above data are specified by the user and are necessary for the independent operation of the Solver. The txt export allows the simulation environment to be maintained constant without repeating the creation process every time the solver input parameters change.

2.2 Solver

2.2.1 Mathematical formulas of ant colony optimisation

The mathematical model, equation and method can be described as follows.

The total number of ants is described by the variable rAnt. Assuming that the total number of the ant's movement points is s, the distances between the point where the ant is located and its possible movement points are known. The pheromone concentration for the path connecting a point e with a point f for a specific time is $\tau_{e,f}(t)$. Before the iterations and the changing of a point's pheromone value begin, the pheromone's value for each point is defined by an initial value $\tau_{e,f}(0) = \tau_0$. The ant g = (1, 2, …, rAnt) determines its movement from the current movement point to another through the pheromone's concentration of possible points. The variable $P^g_{e,f}(t)$ represents the probability that ant g is moving from the current point e to point f. It is calculated as:

$$P^g_{e,f}(t) = \begin{cases} \dfrac{[\tau_{e,f}(t)]^\alpha \times [\eta_{e,f}(t)]^\beta}{\sum_{R \in allow_g} \left([\tau_{e,f}(t)]^\alpha \times [\eta_{e,f}(t)]^\beta \right)}, & R \in allow_g \\ 0, & R \notin allow_g \end{cases} \tag{2}$$

where R represents all the next possible points, $\eta_{e,f}(t)$ is the heuristic function, $allow_g$ g = (1, 2, …, rAnt) is a set of movement points that ant g can visit. The exponent α is the importance factor of the pheromone and the exponent β is the importance factor of the heuristic function. The larger the value β, the more important the heuristic function's role is in determining the probabilities for the ant's movement.

The pheromones help with information exchange between ants by changing their environment. They act as a pole of attraction for the ants during the process of food's searching. In the algorithm, they influence its coverage speed and the path's result. Each movement point has its own pheromone value. The higher the value of the pheromone, the more it attracts ants. Local and general pheromone updates of movement points is used in the algorithm.

2.2.2 Local pheromone update

Every time an ant passes a movement point A(i, j, k) it directly invokes the pheromone update rule for that point. In a local update, the pheromone decreases. In this way, the probability of visiting points previously passed by the ants is reduced and the probability of selecting a different point

and exploring new routes is increased. The local pheromone update is calculated as:

$$\tau_{i,j,k}(t+1) = (1-\xi)\times\tau_{i,j,k}(t) \quad (3)$$

where $\tau_{i,j,k}(t)$ is the concentration of the pheromone at the movement point A(i, j, k), t is the update number of the specific pheromone, while ξ is the attenuation coefficient of the pheromone and must be $0 < \xi < 1$. The attenuation coefficient ξ represents the percentage reduction of pheromone between time t and t + 1. A percentage reduction of pheromone is used to avoid negative values in pheromone throughout several iterations and to consistently reduce the reselection rate of that point based on how many times it has already been selected. In this way, the ants explore more paths to find the optimal solution. The coefficient is also directly related to the coverage speed of the algorithm.

2.2.3 Global pheromone update

After a computational cycle of ants is completed, each colony ant has formed a path, the optimal path is selected and the global pheromone update is applied to the set of its movement points. The global pheromone update is calculated as:

$$\tau_{i,j,k}(t+1) = (1-\rho)\times\tau_{i,j,k}(t) + \rho\times\Delta\tau_{i,j,k} \quad (4)$$

$$\Delta\tau_{i,j,k} = \frac{\lambda\times(N-M)+K}{\min(\{\text{length}(g)\})} \quad (5)$$

where ρ is the global pheromone update coefficient, $0 < \rho < 1$ and length(g) indicates the path length set of ant g, g = 1, 2, …, rAnt, min ({length(g)}) indicates the shortest path calculated by ant g. In equation (5), N indicates the maximum iteration number, M indicates the current iteration number and K, λ are constants.

N, M, λ variables decrease the increase of pheromone while the iteration's number increases. Therefore, the pheromone concentration on the optimal path increases less, which reduces the probability of other ants choosing only the optimal path and increases the possibility of searching for a solution near the optimal. In this way, ants are more easily oriented in space and search for the overall optimal solution, avoiding at the same time the continuous selection of an optimal path and the transformation of the algorithm into a local optimum.

2.2.4 Heuristic function

The heuristic function uses the heuristic rules to guide the ant from the starting point to the end point. The heuristic rules should not only contain finding of the shortest path but also avoiding obstacles. The safety value calculated in the pre-processor is used in the heuristic function. The heuristic function is calculated as:

$$Q(i, j, k) = U(i, j, k)^{w1} + V(i, j, k)^{w2}\times S(i, j, k)^{w3} \quad (6)$$

$$U(i, j, k) = \frac{1}{\sqrt{(i-i_c)^2+(j+j_c)^2+(k-k_c)^2}} \quad (7)$$

$$V(i, j, k) = \frac{1}{\sqrt{(i_e-i)^2+(j_e+j)^2+(k_e-k)^2}} \quad (8)$$

where (i, j, k) indicates the next candidate movement point, (i_c, j_c, k_c) are the coordinates of the current movement point, (i_e, j_e, k_e) are the coordinates of the end point. Q(i, j, k) is the heuristic function of the next candidate movement point, U(i, j, k) expresses the reciprocal of distance between the current point and the next candidate point, V(i, j, k) expresses the reciprocal of distance between the next candidate point and the end point. S(i, j, k) is the safety value of the next candidate point. w1, w2 and w3 are the coefficients, which represent the importance of U(i, j, k), V(i, j, k) and S(i, j, k) respectively, w1, w2 and w3 ∈ [0, 1]. The heuristic function is used for the calculation of the probability in equation (2).

For solving the TSP, the heuristic function is calculated as:

$$Q_{k,l} = \frac{1}{d_{k,l}} \quad (9)$$

where k represents the current moving point and l the next candidate movement point, $d_{k,l}$ is the reciprocal of the distance between the current point and the next candidate point.

2.2.5 Roulette wheel method

The roulette method was implemented as a separate function. Its input is a vector with the selection probabilities of each point and its output is the number of the positions of the chosen probability in the vector. As Chipperfield and Fleming (2020) explained, the roulette wheel is the sum of all possible movement points, each movement point has different selection probability. The selection probability is calculated using equation (2). To select a point, a random number is generated in the interval [0, sum of the point's expected selection probabilities]. The point whose interval segment spans the random number is selected. Any option whose probability is greater than 0 can be the exit option of the roulette wheel.

2.2.6 Path length calculation

The movement points in space are defined by specific coordinates (x, y, z).The distance between two points is given by the following formula:

$$d(A_1, A_2) = \sqrt{(x_1-x_2)^2+(y_1-y_2)^2+(z_1-z_2)^2} \quad (10)$$

The calculation of the paths' total length can be expressed as the sum of the distances of the movement points and is calculated as:

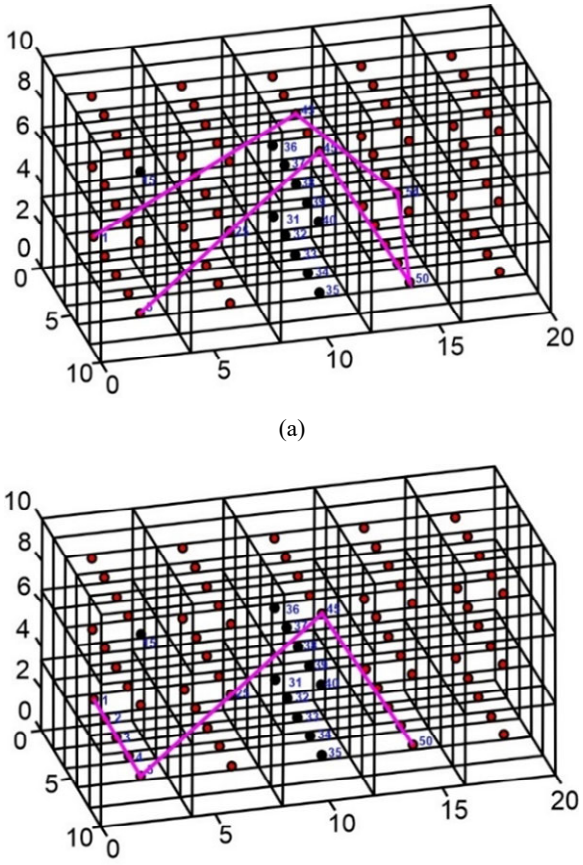
$$L = \sum_{a=1}^{q-1}\sqrt{(x_{a+1}-x_a)^2+(y_{a+1}-y_a)^2+(z_{a+1}-z_a)^2} \quad (11)$$

where q is the total number of path's movement points.

2.2.7 Initial pheromone value

The pheromone value is stored in a 2D matrix called tau0. The tau0 matrix consists of rows and 2 columns. In the first column, the number of each movement point is stored and in the second column, the pheromone value of each point is stored. Initially, all the points have the same amount of pheromone except the obstacles that have a number that approaches 0, like 10^{-8}.

Figure 8 (a) Results of manual method (b) Results of auto method (see online version for colours)



(a)

(b)

2.2.8 Solver explanation

The solver can solve the problem in two ways. The first way, called manual, calculates the shortest path according to the order of input points. The second way, called auto, calculates the shortest path that connects the input points. Figures 8(a) and 8(b) show the two method's results for the same input order of points: 1, 50, 5, and 25. Both solutions are derived from the equations and the methodologies described in previous sections.

In the beginning, the pre-processor's data are entered from the txt file. Additionally, one of the methods must be selected, the manual or the auto method. Then the quantity of target points, the parameters of equations (2)–(6), the number of ants in colony and the colonies number are entered. After this, the target points are entered, which are considered the starting and ending points of a path. These points are recorded in a row vector called point. The vector's column size is equal to the quantity of target points, one column for each point. The first column contains the starting point. After this process is completed, the calculation of the solution begins according to the selected method, manual or auto.

2.2.9 Manual option

In the manual option, a procedure is followed to find the optimal path between two consecutive points of the point2 vector and then expand for all defined points. A total of f – 1 optimal paths are constructed, where f is the number of points2 vector's columns. A starting and an end point are defined based on the point2 vector, after finding the optimal path is completed, the ending point is considered the starting point of the next path. This process is repeated f – 1 times.

The methodology and process followed for the creation of each route are as follows. Each ant creates its own path separately. Each ant colony has a specific number of ants and the number of colonies is already inputted. Initially, according to the position and with the help of the mov matrix from the pre-processor, a vector r is created. Depending on the number of the ant's position point, vector r is the row of the mov matrix. When the row contains 0, they are deleted, so the vector r contains only the next feasible points. Moreover, the ant's position point is not registered in the r vector.

After creating the vector r, the heuristic function is calculated for all its points. Then the product $\tau(i, j, k) \times Q(i, j, k)$ is calculated to calculate the probability of each point. After calculating the probability of each point using the roulette wheel method, the next point of the ant's movement is selected. After the next point is selected, the local pheromone update rule is used for this point. This process is repeated until the ant reaches its final determination.

The optimal route is stored in a 2D matrix called storemat. This matrix consists of f – 1 rows, where f is the number of points2 vector's columns and p columns, and p is the number of movement points in the space, the same as the number of rows in the mov matrix. In each row of the storemat, the optimal route between two points is entered. The number of optimal path's movement point's is registered in each column. The remaining columns of mov matrix are filled with 0. In addition to the above, a 2D matrix named Ddiag is also created. Ddiag contains the optimal path's length of each colony and is used to create a diagram with the number of iterations and the optimal path's length. This diagram assists in the solver's input parameters optimisation.

2.2.10 Auto option

The auto option uses a part of the manual option to solve the tsp problem between the input points. In the beginning, the same data as the manual are inputted. A square matrix

called Dtsp is created. The Dtsp matrix consists of f rows and f columns, where f is the number of points2 vector's columns. The matrix's rows represent the starting point and its columns represent the ending point. In each position of the table, the length of the route between the starting point and the end point of the corresponding row and column is registered. The main diagonal of the matrix is zero. In addition, the matrix is symmetric since the distance between two identical points remains constant regardless of which is considered the starting or ending point. It follows that Dtsp(i, j) = Dtsp(j, i). Table 3 shows an example of Dtsp matrix for 4 points.

Table 3 Dtsp matrix 4 points example

n/n	*1*	*2*	*3*	*4*
1	0	20.62	8	11.77
2	20.62	0	18.18	15.77
3	8	18.18	0	6.66
4	11.77	15.77	6.66	0

The path's length between two points is calculated in the same way as described in the manual option. A square matrix similar to the Dtsp matrix called tsptauis is created. A Tsptau matrix consists of f rows and f columns. Tsptau contains the initial value of tsp pheromones. Then the heuristic function is calculated by using equation (9). Later, the probability of each point is calculated and based on the roulette wheel method, the next point is chosen. This process is repeated until the final tsp path is formed. Based on this formed tsp path, the local update of the pheromone takes place, namely the reduction of the corresponding values of the pheromone in the tsptau matrix. Reduction occurs when an ant is transported from one point to another. When decrementing the tsptau(i, j) value, the ant moved from point i to j. The values of the diagonal of the matrix remain with the initial value of the pheromone. Table 4 shows an example of pheromone reduction. The formed tsp path describes the transportation from points 1 to 2, 2 to 4 and 4 to 3. The reduction of the pheromone has an indicative value of 15. The example's initial pheromone value is 100.

Table 4 Tsptau matrix pheromone reduction example

n/n	*1*	*2*	*3*	*4*
1	100	85	100	100
2	100	100	100	85
3	100	100	100	100
4	100	100	85	100

After the colony completes finding the shortest path between the points, the general pheromone is updated in a similar way as the local. In addition, a tspDdiag is created. TspDdiag contains the optimal path's length of each colony and is used to create a diagram with the number of iterations and the optimal path's length for tsp problem solving.

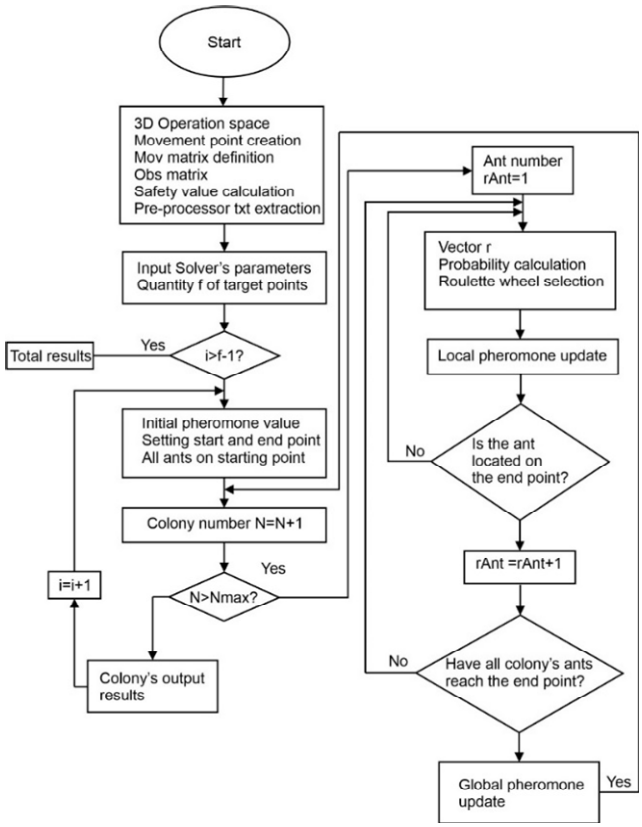
Finally, after finding the shortest path that connects these points to each other based on the storemat matrix, the path is drawn in the 3D operating space diagram. In addition, the Ddiag chart is generated separately with the number of iterations and the shortest path for every two consecutive points. The tspDdiag diagram is also created in a separate window. As mentioned in the manual option, these diagrams assist in the solver's input parameters optimisation.

At the end of the solver, a txt file is created with the coordinates of the movement points of the optimal path. The txt file is created for both the manual and auto options.

2.2.11 Algorithm's parameter selection

The number of ants should be determined according to the size of the problem. The grid density and the size of the operating spaces have a leading role in ant's number selection. The final values should be determined after performing a number of tests. Indicatively, for an operating space with 75 movement points, the number of ants could range from 20 to 100. The coefficients a, b, ρ and ξ are interdependent. Several combinational simulations are needed to set them up. The values of the coefficients in this paper were set to a = b = 1 and ρ = ξ = 0.2.

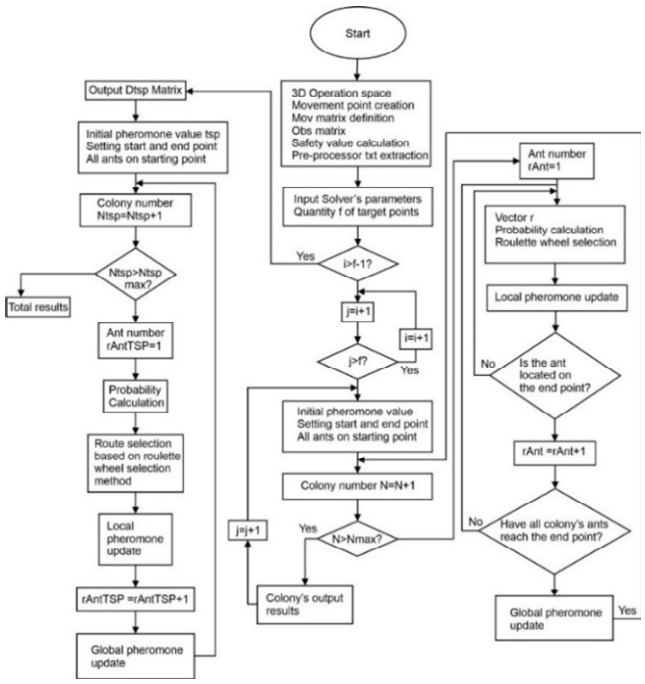
Figure 9 Flowchart of manual option



The heuristic function is significantly affected by the values of the parameters w1, w2, and w3. The routes that are constructed up until the optimal route is found alter as a result of the various combinations of these parameters. They also alter the number of iterations needed until the optimal

solution is found. The values of w1, w2 and w3 in this paper were set to w1 = w2 = w3 = 1.

Figure 10 Flowchart of auto method



Another parameter that can be adjusted is the number of ant's colonies. A large number of ant's colonies costs computing time. On the other hand, a small number may not find the optimal path. Ant's colony number depends on the grid's density, operation space size, and the quantity and complexity of obstacles. λ and K variables significantly affect the pheromone update process and, therefore, the path generation. The values of ant's colonies, λ and K in this paper were set: ant's colonies number = 100, K = 200 and λ = 0.3.

2.2.12 Flowcharts of manual and auto methods

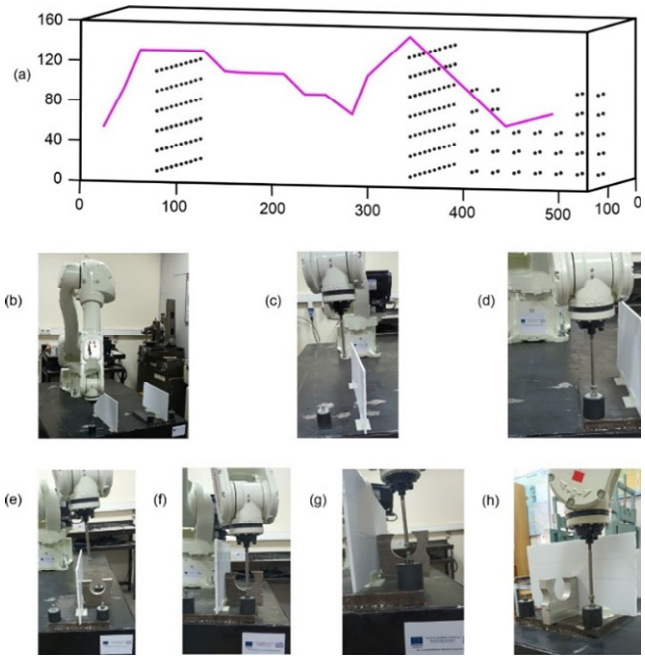
Figures 9 and 10 show the flow charts for manual and auto options, respectively.

3 Experimental results

The results of the proposed method were further investigated by applying them to a real experimental challenge. In order to simulate a 3D environment with obstacles and determine an optimal path for safe navigation of a tool attached to the robot's end effector, the proposed method has been used. A Kawasaki RS010N industrial robot was employed for the experiment. The environment consists of three obstacles and four target points, as shown in Figure 11. Figure 11(a) shows the results of the proposed algorithm in a simplified manner and Figures 11(b)–11(h) show the different stages of path execution.

The method was tested for various values and combinations of w1, w2 and w3 coefficients. There is no mathematical restriction on the values of w1, w2 and w3. However, as mentioned above, it is recommended w1, w2, w3 ∈ [0, 1]. The values of w1, w2, w3 represents the importance of U(i, j, k), V(i, j, k) and S(i, j, k) respectively. As a result, they affect the heuristic function's value. When w1, w2 and w3 > 1, the values of U(i, j, k), V(i, j, k) and S(i, j, k) are decreasing. Hence, this decrease affects the contribution of the heuristic function to the probability calculation equation (2). When the value of the heuristic function is decreased, the value of probability's calculation is more impacted by the value of the pheromone, thus, the formed path depends almost exclusively on the pheromone's concentration. This fact may lead to a local optimal solution, an increase in computation time, or an incorrect finding of the optimal path.

Figure 11 Experimental results of a real world path planning challenge, (a) simplified results of the proposed method (b) robot's tool at starting point (c) robot's tool before 1st obstacle (d) robot's tool at 2nd target point (e) robot's tool after 2nd obstacle (f) robot's tool at 3rd target point (g) robot's tool crossing 3rd obstacle (h) Robot's tool at 4th target point (see online version for colours)



The following diagrams depict the algorithm's behaviour for the aforementioned tests. The tests were conducted in the same operation spaces with the same predefined obstacles. The tests carried out in the operation space depicted in Figure 8(a) are for the manual method. The purpose was to examine the algorithm's responsiveness and reliability in order to determine the optimal path from start point 1 to end point 50. The initial pheromone value was equal to 1,000, the ant's colonies number was equal to 100 and the number of ants in the colony was equal to 40. The coefficients a and b of equation (2) were equal to 1, the coefficients ξ and ρ of equations 3 and 4 were equal to 0.2, the constant λ of equation 5 was set to 0.3 and the constant K of equation 5 was equal to 200.

Figure 12 shows the results for w1 = w2 = 1 and the value of w3 is changed.

Figure 12 Diagrams for constant w1, w2 and changeable w3 (see online version for colours)

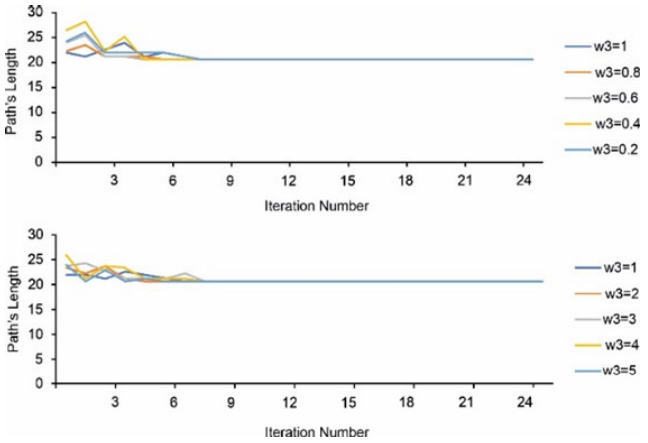
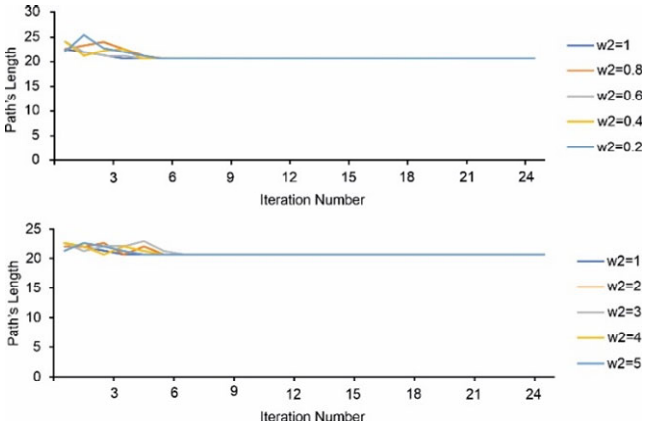


Figure 13 Diagram for constant w1, w3 and changeable w2 (see online version for colours)



In Figure 12, some variations are observed for different values of w3, but all tests converge to the optimal solution after a certain number of iterations.

Figure 13 shows the results for w1 = w3 = 1 and the value of w2 is changed.

In Figure 13, some variations are observed for different choices of the arithmetical values of w2, but after a certain number of iterations, all the experimental tests converge to the optimal solution.

Figure 14 shows the results for w2 = w3 = 1 and the value of w1 is changed.

In Figure 14, some variations are observed for different values of w1, however all tests converge to the optimal solution after a certain number of iterations. For the case of w1 = 5 the coverage of the optimal solution needs a higher number of ant's colonies compared to other tests.

Figure 15 shows the results for w1 = w2 = w3.

The graphs in Figures 12–15 show that for various values of w1, w2, and w3, the ants can efficiently navigate from a starting point to an end point, despite the fact that there are a vast number of alternative selections from one point to another. In addition, paths of varying length are formed, indicating that the ants are exploring several instances throughout the area. The length of the formed path is longer than the length of the ideal path in the initial iterations, indicating that the formed route improves as the algorithm executes and the methods of navigation and obstacle placement are taken into account during the path forming process, allowing a more q realistic simulation of the robot's movement. If the length of the created routes is initially unusually short, it indicates that the algorithm may not take into account the obstacles or that the movement strategy is improper. When the ant colony optimisation and roulette wheel methods are used with erroneous obstacle placement and movement strategy, they can create a path that approaches the target through the obstacles, which does not occur with the proposed method.

Figure 14 Diagram for constant w2, w3 and changeable w1 (see online version for colours)

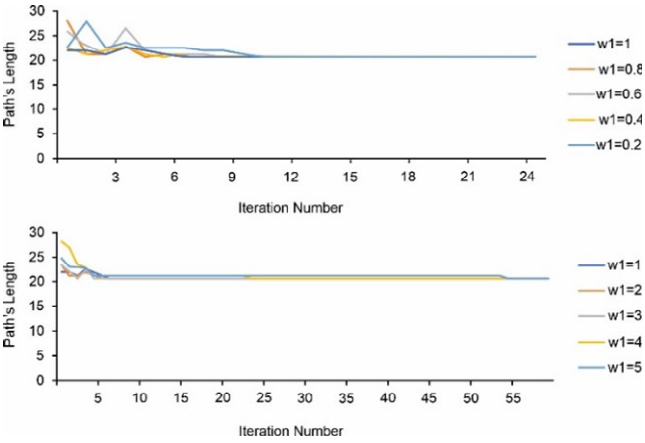
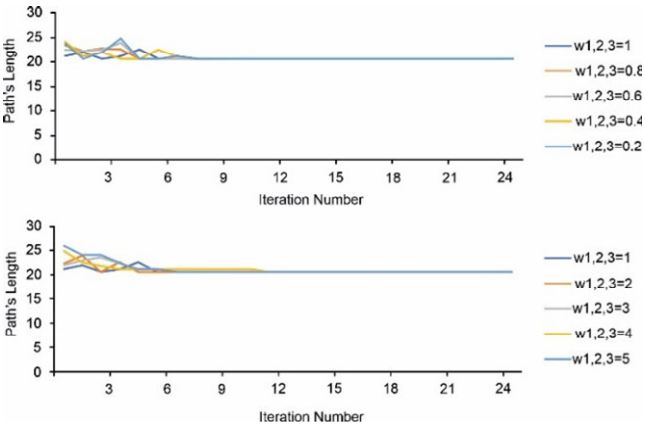


Figure 15 Diagram for w1 = w2 = w3 (see online version for colours)



4 Conclusions

Ant colony optimisation combined with the roulette wheel method can be used effectively for offline programming of industrial robots. The proposed method finds the optimal path in a 3D operation space with obstacles consistently. The manual method can find the shortest distance between two or more points while avoiding obstacles. The auto method, assisted by the fundamentals of the manual method, successfully applies the ant colony optimisation solution to the TSP in a 3D operation with obstacles. One of the disadvantages of applying the ant method in 3D space is the increased computational requirements, which are mitigated by calculating a priori some indispensable processes such as the calculation of safety value and the feasible movement points of the robot from a specified position. In this way, the demanding calculations are performed only once and the data can be accessed when necessary.

Ant colony optimisation depends on a variety of coefficients that, with proper tuning, can drastically affect the speed of finding the optimal path. The proper tuning can also help avoid local optimal solutions and form efficient and fast paths in a convoluted 3D operation space with several points and obstacles. Optimal coefficient values may vary proportionately with the complexity of the environment.

References

Andreu-Perez, J., Deligianni, F., Ravi, D. and Yang, G-Z. (2017) 'Artificial intelligence and robotics', *EPSRC UK-RAS*, https://doi.org/10.31256/wp2017.1.

Brand, M., Masuda, M., Wehner, N. and Yu, X-H. (2010) 'Ant colony optimization algorithm for robot path planning', in *International Conference on Computer Design and Applications*, 25–27 June, https://doi.org/10.1109/iccda.2010.5541300.

Chaari, I., Koubaa, A., Bennaceur, H., Trigui, S. and Ai-Shalfan, K. (2012) 'SmartPATH: a hybrid ACO-GA algorithm for robot path planning', in *IEEE Congress on Evolutionary Computation*, 10–15 June, https://doi.org/10.1109/cec.2012.6256142.

Chipperfield, A. and Fleming, P. (1994) *Genetic Algorithms: A Survey*, Research Report, in Department of Automatic Control and Systems Engineering, ACSE Research Report 518.

Dai, X., Long, S., Zhang, Z. and Gong, D. (2019) 'Mobile robot path planning based on ant colony algorithm with A* heuristic method', *Front. Neurorobot*, Vol. 13, https://doi.org/10.3389/fnbot.2019.00015.

Demir, H., Tolun, M.R. and Sari, F. (2021) 'Time optimal path planning model using genetic algorithm in RRR robot', *International Journal of Engineering Technologies and Management Research*, Vol. 8, https://doi.org/10.29121/ijetmr.v8.i5.2021.938.

Dorigo, M., Birattari, M. and Stützle, T. (2006) 'Ant colony optimization artificial ants as a computational intelligence technique', in *IEEE Computational Intelligence Magazine*, 18–20 October, pp.28–39, https://doi.org/10.1109/CI-M.2006.248054.

Floreano, D. and Wood, R.J. (2015) 'Science, technology and the future of small autonomous drones', *Nature*, Vol. 521, pp.460–466, https://doi.org/10.1038/nature14542.

Kwaśniewski, K.K. and Kwaśniewski, K.K. (2018) 'Genetic algorithm for mobile robot route planning with obstacle avoidance', *Acta Mechanica et Automatica*, Vol. 12, pp.151–159, https://doi.org/10.2478/ama-2018-0024.

Li, B., Wang, L. and Song, W. (2008) 'Ant colony optimization for the traveling salesman problem based on ants with memory', *Fourth International Conference on Natural Computation*, https://doi.org/10.1109/icnc.2008.354.

Liu, H., Ge, J., Wang, Y., Li, J., Ding, K., Zhang, Z., Guo, Z., Li, W. and Lan, J. (2022) 'Multi-UAV optimal mission assignment and path planning for disaster rescue using adaptive genetic algorithm and improved artificial bee colony method', *Actuators*, Vol. 11, https://doi.org/10.3390/act11010004.

Ma, Y-N., Xiao, C-F., Gong, Y-J. and Zhang, J. (2019) 'Path planning for autonomous underwater vehicles: ant colony algorithm incorporating alarm pheromone', *IEEE Transactions on Vehicular Technology*, Vol. 68, pp.141–154, https://doi.org/10.1109/tvt.2018.2882130.

Mac, T.T., Copot, C., Tran, D.T. andDe Keyser, R. (2016) 'Heuristic approaches in robot path planning: a survey', *Robotics and Autonomous Systems*, Vol. 86, pp.13–28, https://doi.org/10.1016/j.robot.2016.08.001.

Malone, T.W., Rus, A.D., Viterbi, E. and Laubacher, R. (2020) 'Artificial intelligence and the future of work', *MIT Work of the Future*, Research Brief 17.

Melanie, M. (1999) *An Introduction to Genetic Algorithms*, 5th ed., MIT Press, London.

Ostfeld, A. (2011) *Ant Colony Optimization Methods and Applications*, 4th ed., Intech Open, https://doi.org/10.5772/577.

Patle, B.K., Babu, L.G., Pandey, A., Parhi, D.R.K. and Jagadeesh, A. (2019) 'A review: on path planning strategies for navigation of mobile robot', *Defence Technology*, Vol. 15, pp.582–606, https://doi.org/10.1016/j.dt.2019.04.011.

Ravankar, A., Ravankar, A.A., Kobayashi, Y. and Emaru, T. (2017) 'Symbiotic navigation in multi-robot systems with remote obstacle knowledge sharing', *Sensors*, Vol. 17, https://doi.org/10.3390/s17071581.

Reshamwala, A. and Vinchurkar, D.P. (2013) 'Robot path planning using an ant colony optimization approach: a survey', *International Journal of Advanced Research in Artificial Intelligence*, Vol. 2, https://doi.org/10.14569/ijarai.2013.020310.

Sanchez-Lopez, J.L., Wang, M., Olivares-Mendez, M.A., Molina, M. and Voos, H. (2019) 'A real-time 3D path planning solution for collision-free navigation of multirotor aerial robots in dynamic environments', *Journal of Intelligent & Robotic Systems*, Vol. 93, pp.33–53, https://doi.org/10.1007/s10846-018-0809-5.

Santos, L., Valente, A., Dos Santos, F.N. and Costa, P. (2020) 'Path planning for ground robots in agriculture: a short review', in *IEEE International Conference on Autonomous Robot Systems and Competitions*, 15–17 April, https://doi.org/10.1109/icarsc49921.2020.9096177.

Wang, L., Kan, J., Guo, J. and Wang, C. (2019) '3D path planning for the ground robot with improved ant colony optimization', *Sensors*, Vol. 19, https://doi.org/10.3390/s19040815.

Wang, X., Sahin, A. and Bhattacharya, S. (2022) *Coordination-free Multi-robot Path Planning for Congestion Reduction Using Topological Reasoning*, https://doi.org/10.48550/arXiv.2205.00955.

Zafara, M.N. and Mohanta, J.C. (2018) 'Methodology for path planning and optimization of mobile robots: a review', *Procedia Computer Science*, Vol. 133, pp.141–152, https://doi.org/10.1016/j.procs.2018.07.018.

Zhang, C., Ao, L., Yang, J. and Xie, W. (2020) 'An improved A* algorithm applying to path planning of games', *Journal of Physics 2nd International Conference on Artificial Intelligence and Computer Science*, https://doi.org/10.1088/1742-6596/1631/1/012068.

Zhang, S., Pu, J., Si, Y. and Sun, L. (2021) 'Path planning for mobile robot using an enhanced ant colony optimization and path geometric optimization', *International Journal of Advanced Robotic Systems*, https://doi.org/10.1177/17298814211019222.