



Research on Multi-modal Multi-objective Path Planning by Improved Ant Colony Algorithm

Juan Jing¹, Ling Zhang¹, Chaonan Shen^{1(✉)}, and Kai Zhang^{1,2}

¹ School of Computer Science and Technology, Wuhan University of Science and Technology, Wuhan 430065, China

449777215@qq.com

² Hubei Province Key Laboratory of Intelligent Information Processing and Real-Time Industrial System, Wuhan University of Science and Technology, Wuhan 430065, China

Abstract. A research about the path planning problem has been popular topic nowadays and some effective algorithms have been developed to solve this kind of problem. However, the existing algorithms to solve the path problem can only find a single optimal path, cannot satisfactorily find multiple groups of optimal solutions at the same time, and it is very necessary to propose as many solutions as possible. So this paper carries out a research on the Multi-modal Multi-Objective Path Planning (MMOPP), the objective is to find all sets of Pareto optimal path solutions from the start point to the end point in a grid map. This paper proposes a multi-modal multi-objective ant colony path planning optimization algorithm based on matrix preprocessing technology and Dijkstra algorithm (MD-ACO). Firstly, a new method of storing maps that reduces the size of the map and reduces the size of the decision space has been proposed in this paper. Secondly, using the characteristics of the Dijkstra algorithm that can quickly find the optimal path, generate an initial feasible solution about the problem, and improve the problem that the initial pheromone of ant colony algorithm is insufficient and searching for solutions is slow. Thirdly, a reasonable threshold is set for the pheromone to avoid algorithm getting stuck in local optimal solution. Finally, the algorithm is tested on the MMOPP test sets to evaluate the performance of the algorithm, and the results show that MD-ACO algorithm can solve MMOPP and get the optimal solution set.

Keywords: Path Planning · Multi-modal Multi-objective Optimization · Ant Colony Optimization (ACO) · Dijkstra

1 Introduction

Multi-Objective problem means that solving a problem needs to consider multiple objectives, and the objectives are conflicting and restricting each other. Generally speaking, optimizing one of the objectives will lead to the weakening of other objectives. For example, in the path planning problem, it may be necessary to consider multiple objectives such as the quantity of blocked paths, path length, and the quantity of intersections. There may be exist where the length of path is short but the quantity of blocked paths

or the quantity of intersections is relatively large. Therefore, there may be more than one optimal solution for the Multi-objective optimization problem (MOP) [1–5], and there may be a set composed of multiple optimal solutions. The front composed of these optimal solutions in the objective space are called Pareto Front (PF). In MOP, there may be two or more global or local Pareto optimal sets, some of which may correspond to the same PF, which is the Multi-modal Multi-objective Optimization Problem (MMOP) [6].

MMOP can be described as

$$\min_x^{f(x)} = \min_x^{[f_1(x), f_2(x), \dots, f_m(x)]}, x \in \Omega, f_i(x) \in R^m \quad (1)$$

$$PS = \{PS_1, \dots, PS_k\}, k > 1 \quad (2)$$

where the decision vector $X = (x_1, x_2, \dots, x_n)$ belongs to the non-empty decision space Ω , the objective function vector $f : \Omega \rightarrow \Lambda$ consists of $m(m \geq 2)$ objectives and Λ is the objective space. A few relevant definitions are briefly described below: a solution $x \in \Omega$ in a decision space that satisfies these constraints is called a feasible solution. Given two solutions $x, y \in \Omega$ and their corresponding objective vectors $f(x), f(y) \in R^m$, x dominates y (denoted as $x < y$) when $\forall i \in \{1, 2, \dots, m\}, f_i(x) \leq f_i(y)$ and $\exists j \in \{1, 2, \dots, m\}, f_j(x) < f_j(y)$. A solution that is not dominated by any other solution is defined as Pareto optimal solution. In Eq. (2), it is assumed that PS_k is the k -th Pareto optimal solution, $\{PS_1 \dots PS_k\}$ denotes multiple equivalent Pareto optimal solutions, corresponding to the same PF, PS is a set consisting of PS_k .

In this paper, a new path planning combinatorial algorithm is proposed to solve the MMOPP problem. Firstly, to improve the speed of the global path search of ACO, Dijkstra is used to improve the initial pheromone concentration allocation for different costs and increase the accuracy of the initial search. Secondly, in order to maintain the diversity of solutions in the decision space, non-dominated solutions are retained in the ant colony algorithm by judging dominant relationships. Finally, in order to maintain the diversity of the objective space and avoid getting stuck in the local optimal solution, the pheromone threshold is set.

2 Related Works

2.1 MMOPP Example

The path problem is a typical MOP problem. As shown in Fig. 1, from the start point to the end point, there are three optimal paths. These three paths correspond to the same path length and the same congestion point. The optimal solutions of these paths correspond to the same PF in the objective space. The purpose of this study is to obtain multiple paths with optimal and equal target values, such as the length of multiple paths, the number of blocked road sections, and the number of cross-road conditions.

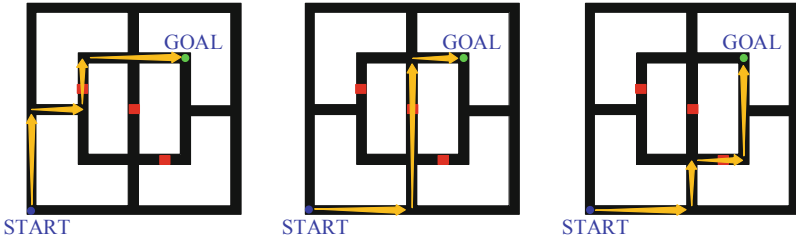


Fig. 1. Three paths corresponding to the same point in PF

2.2 Research on MMOPP Problems

From the perspective of practical application, the research of MMOPP problem is usually divided into three types. One is path planning that simulates real-life road conditions [6], decision makers need to consider planning requirements such as path length, congestion points, and must-pass points at the same time. Two is the vehicle path planning considering the logistics distribution requirements [7], which usually needs to consider the load constraints of the distribution vehicles and the time constraints of the distribution points. Another is to consider robot path planning [8], which needs to consider planning requirements such as path length and obstacles. This paper considers the first MMOPP problem, and studies the Multi-objective Path Planning problem under the requirement of multi-modal optimization.

From the point of view of selection algorithm, classical path planning algorithms such as Dijkstra algorithm and A* algorithm only consider the optimization of path length, which is hard to resolve the MMOPP problem. Genetic algorithm [9], particle swarm algorithm [10], evolutionary algorithm NSGA-II [11], etc. can also solve the MMOPP problem, but these algorithms focus more on approaching the PF, ignoring the diversity of solution distributions in the decision space. Therefore, most scholars choose to introduce swarm intelligence algorithms with self-organization, self-adaptation and self-learning characteristics to realize MMOPP. The ACO used in this paper is used by many scholars to solve the path problem. People are committed to improving ACO to improve its effectiveness and applicability. At present, it mainly focuses on two aspects: process improvement and parameter setting. In terms of process improvement, Mao [12] proposed an improved A* algorithm to quickly find a better path, and adjust the pheromone concentration on the path through coefficients. MOHAMED et al. [13] proposed a solution method combining local search and ACO to solve the vehicle routing problem with loading capacity constraints. In terms of parameter setting, Yuan et al. [14] studied the path problem of the tea picking robot, improved the ant colony algorithm, and improved the global search ability and computational efficiency by changing the adaptive adjustment pheromone concentration value and iterative termination conditions. Liu et al. [15] proposed adaptive pheromone concentration and dynamic pheromone volatilization factor when designing a path navigation system for indoor service robots. The improved ACO has higher global search ability.

The MD-ACO used in this paper combines the process improvement and parameter setting to solve the MMOPP problem. The MMOPP test set used in the algorithm

validation phase includes numerical quantification of targets such as path length, road width, road congestion, and blocked road segments, and includes solution constraints for the must-pass point problem. Therefore, the test experiments in this paper have a tightly coupled relationship with the actual application scenarios. The experimental results verify that the MD-ACO can solve the multi-modal and multi-objective path planning requirements in parallel. The comparison with other algorithms shows that the algorithm proposed in this paper has stronger search ability and optimization ability, and the diversity of solutions is better.

3 Problem Description

In this chapter, we describe the MMOPP problem based on the documentation provided by the test set [16–18]. This test set models the road features in the actual road network map into a regular grid map, which is divided into three categories according to the types of optimization objectives, with a total of 12 test questions. The first type of test problem simulates the congested road sections in the actual road network map, and its optimization goal is to minimize the path length, the quantity of congested road sections, and the quantity of intersections. According to the size of the problem, this type of problem contains 5 test questions. In the second type of test problem, different F values are used to simulate information such as road congestion or road width, and the optimization objective is to minimize the path length and each F value. According to the number of optimization objectives, this type of problem contains 5 tests question. The third type of test problem simulates the path planning requirements that have must-pass points constraints, and allows the existence of a re-entrant path. The optimization goal is to minimize the path length and each F value. According to the different scale of the problem, this type of problem contains two test problems.

The test set provides some definitions. The map is a two-dimensional matrix consisting of 0 or 1. The matrix value of 0 means that this area is passable in the map, and the value of 1 means that this area is not passable. Walking in four directions is permitted, repeating the same path is not allowed, and it is forbidden to go beyond the boundary of the map. As shown in Fig. 1, the map also gives the starting point and ending point, red congestion points, yellow must-pass points. At the same time, passing a certain area in the map will generate different costs, this cost has multiple dimensions, including road width, road congestion, the number of intersections, and must-pass points.

The cost of each area is expressed as (3), the cost of the path is expressed as (4).

$$cost \rightarrow [f_1, f_2, \dots, f_m], f_m \in F \quad (3)$$

$$cost_{path}[i] = \sum_{j=1}^n cost[j][i], i \in M \quad (4)$$

In formula (3), the cost array consists of the costs of different optimization objectives. In formula (4), $cost_{path}[i]$ is defined as the i -th cost sum of this path, $cost[j][i]$ is defined as the i -th cost of the j -th region on this path (Fig. 2).

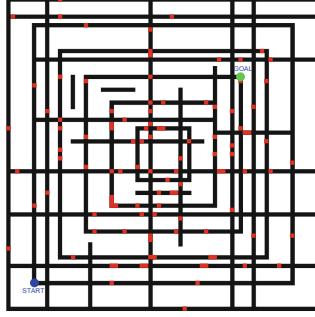


Fig. 2. Map example

4 Proposed Approach

4.1 Matrix Pre-processing

Since the test set provides us with a 0–1 matrix, we need to preprocess the matrix at first. The optimization of matrix preprocessing also helps to reduce the scale and complexity of the problem. Matrix preprocessing includes three steps. First, we need to get rid of useless roads, it means to turn some 1 in the matrix that are considered as useless areas into 0 in a 0–1 matrix. The result obtained at this time is still a 0–1 matrix. Second, we need to get the set of valid points, it means that we need to get a set of some key points in the matrix, then check the feasibility of the problem. Third, we need to turn the 0–1 matrix into a weighted directed graph.

Removal of Unless Areas. We can process the 0–1 matrix and delete unnecessary areas in the map to shrink the problem scale. Abridged regions can be divided into two types. First, disconnected regions belonging to two road segments are useless. Second, we get all the key points from the map. If the must pass point is not in this group of areas separated by key points, we can ignore this group. The following Fig. 3(a) is the original map generated by the two-dimensional matrix, and the following Fig. 3(b) is the reduced map after matrix preprocessing.

I will then describe the algorithms and algorithm-related auxiliary arrays used to implement this section. $Depth[x][y]$ represents the depth of DFS search. $Minor[x][y]$ represents depth of the nearest common ancestor in DFS search. $Children[x][y]$ represents the subset of children of area (x, y) in DFS search. $Must[x][y]$ represents whether the current point is a necessary point.

After creating the auxiliary matrix, we can build a DFS spanning tree by first traversing the map, and then traversing the DFS spanning tree to do the following operations. If the child node (x', y') of (x, y) belongs to $children[x][y]$, and $minor[x'][y'] \geq depth[x][y]$ and (x', y') is not a necessary point, then the subtree rooted at (x', y') can be deleted.

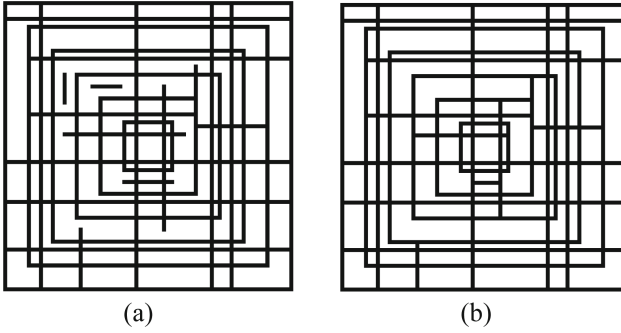


Fig. 3. Comparison of original and reduced maps

Algorithm 1 Removal of useless area

Input: Two-dimensional matrix map

Output: Reduced two-dimensional matrix map

```

1: procedure INITTREE( $x, y, h$ )
2:    $depth[x][y] \leftarrow h$   $minor[x][y] \leftarrow h$ 
3:   for  $(x', y')$  : children list of  $(x, y)$ 
4:     if  $(x', y')$  is not in matrix map then continue
5:     if  $depth[x', y'] = 0$  then
6:        $children[x][y]$  append  $(x', y')$ 
7:       INITTREE( $x', y', h + 1$ )
8:       if  $must[x'][y'] == true$  then
9:          $must[x][y] \leftarrow true$ 
10:      end if
11:       $minor[x][y] \leftarrow \min(minor[x][y], depth[x'][y'])$ 
12:    end if
13:  end for
14: end procedure
15: procedure REDUCETREE( $x, y$ )
16:  set  $(x, y)$  as retained point
17:  for  $(x', y')$  :  $children[x][y]$  do
18:    if  $minor[x'][y'] < depth[x][y]$  or  $must[x][y] == true$ 
19:      REDUCETREE( $x', y'$ )
20:    end if
21:  end for
22: end procedure

```

Get Key Points and Check Problem Feasibility. We first traverse the retained points obtained in the previous step, and then select the start point, end point, must-pass point, and intersection point among the retained points to form a key point array. In addition, before starting to solve the MMOPP problem, we need to check whether the problem has a feasible solution. For some scenarios with necessary points, we need to check whether all feasible solutions have passed the necessary points. This verification is achieved by traversing all retained points and determining whether all necessary points are included.

Build a Weighted Directed Graph. We first traverse the key point set obtained in the previous step, calculate all the costs between the key points, and construct a weighted directed graph, where the weights refer to different types of costs. The data structure of the edge includes the starting point, the ending point, the cost array of the edge, the path composed of all points, and the unique identifier id of the edge. The data structure of the edge is as follows (Fig. 4).

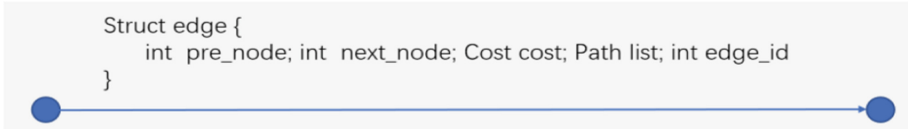


Fig. 4. The data structure of the edge

Algorithm 2 Build a weighted directed graph

Input: key point set

Output: edges sets

```

1: create Path e, Edge edge
2: for current point : key point set do
3:   for next point near to current point do
4:     if next point is not in matrix map then continue
5:     e.add(current point)
6:     while next point is not in valid point sets do
7:       e.add(next point)
8:       calculate the cost according to formula (4)
9:       continue traversing the four directions to get next point
10:    end while
11:   edges.add({current point, next point, e, cost arrays, edges id})
12: end for
13: end for

```

4.2 Initial Pheromone Assignment Based on Dijkstra Algorithm

ACO is designed to simulate the action mode of ant colony looking for food. Ant colonies secrete different pheromones along the way when searching for food, and the amount of pheromone is related to the distance of the path. Other ants will make corresponding decisions according to the concentration of pheromone on the path, and finally the ant colony will gradually gather on the shortest path. Since the initial pheromone concentration of the traditional ACO is evenly distributed, the initial search of the algorithm has strong blindness, which leads to a slow convergence speed and easily makes the algorithm get stuck in local optimum and affects the optimization ability of the algorithm. In order to avoid the algorithm getting stuck in local optimum due to pheromone and reduce the misleading of ants caused by wrong heuristic information, improve the path

planning ability of the algorithm, this paper improves the allocation of initial pheromone based on the minimum cost planned by Dijkstra algorithm.

The Dijkstra algorithm is a classic algorithm for finding the shortest path. Since this paper studies multiple objectives, the objectives to be optimized are not only related to the path length, but also related to different types of cost. The algorithm steps are as follows. First, for each type of cost, we use Dijkstra algorithm to traverse the map, then we can obtain the minimum cost path and the minimum cost sum from the start point to the end point. Second, the pheromone of the point on the least-cost path is set according to formula (5). π is a constant, $cost_i$ represents the total cost of the i -th cost on the minimum cost path, The pheromone of points on a non-minimum-cost path is set to μ_i according to formula (6). Finally, the pheromone of each different cost is accumulated as the initial pheromone allocation.

$$pheromone[i] = \frac{\pi}{cost_i}, i \in costType \quad (5)$$

$$\mu_i = pheromone[i] - \theta \quad (6)$$

Algorithm 3 Dijkstra algorithm

Input: weighted directed graph

Output: Least-cost path and least-cost

```

1: for  $i = 1$  : cost type
2:   create auxiliary matrix  $cost, P, path$ 
3:   for each point  $v$  in graph
4:      $cost[v] \leftarrow Infinity$ 
5:     add  $v$  to  $P$ 
6:   end for
7:   while  $P$  is not empty
8:     choose  $u$  with the least cost
9:     remove  $u$  from  $P$  and add  $u$  to  $path$ 
10:    for each  $v$  near to  $u$ 
11:      if  $v$  is not visited and  $cost[v][i] > cost[u][i] + Cost(u \rightarrow v)$ 
12:         $cost[v][i] \rightarrow cost[u][i] + Cost(u \rightarrow v)$ 
13:      end if
14:    end for
15:  end while
16:  for each point  $k$  in graph
17:    if  $k$  in path
18:      update pheromone according to formula (5)
19:    else update pheromone according to formula (6)
20:  end for

```

4.3 Select the Next Node According to the State Transition Rules

In the process of ants walking, the ants will select the next point to reach according to the probability rule, and record the next point that the ants walk through in the taboo

table. The probability of choosing the next path is expressed as the formula (7).

$$P_{ij}^k(t) = \begin{cases} \frac{[\pi_{ij}(t)]^\alpha * [\varphi_{ij}(t)]^\beta}{\sum_{s \in allowed_k} [\pi_{is}(t)]^\alpha * [\varphi_{is}(t)]^\beta} & j \in allowed_k \\ 0 & other \end{cases} \quad (7)$$

$$\varphi_{ij} = \frac{1}{cost_{ij}} \quad (8)$$

where $P_{ij}^k(t)$ represents the state transfer probability of ants k from point i to point j at time t . $\pi_{ij}(t)$ denotes the pheromone concentration on the path from point i to point j .

In the process of ant walking, if it reaches the end point and has passed all the necessary points, and if the current solution is dominated by the solutions in the PS, it will not be added to the PS. If the current solution is not dominated by the solutions in the PS, it will join the PS.

4.4 Update Pheromone

After all ants have completed a traversal, the residual information on the paths is updated and the pheromones on each path are adjusted according to the following formula.

$$\pi_{ij}(t+1) = \pi_{ij}(t) * (1 - \rho) + \Delta\pi_{ij}(t), 0 < \rho < 1 \quad (9)$$

$$subject\ to\ a \leq \pi_{ij}(t+1) \leq b$$

$$\Delta\pi_{ij}(t) = \sum_{k=1}^m \Delta\pi_{ij}^k(t) \quad (10)$$

$$\Delta\pi_{ij}^k = \begin{cases} \frac{Q}{cost_k} \\ 0 \end{cases} \quad (11)$$

In formula (9), ρ denotes the pheromone volatility coefficient, $\Delta\pi_{ij}(t)$ denotes pheromone increment on path (i, j) after this cycle, if $\pi_{ij}(t+1)$ exceeded a threshold, we take the threshold as its value. In formula (10), $\Delta\pi_{ij}^k(t)$ denotes the pheromone content left on the path (i, j) by the k th ant in this cycle. In formula (11), if ant k passes through (i, j) in this cycle, then execute $\Delta\pi_{ij}^k = \frac{Q}{cost_k}$. $cost_k$ represents the total cost of all paths taken by ant k . Q is the total amount of pheromones released by ant k after completing a complete path search.

5 Experimental Results

All algorithms are implemented in C++ language and compiled on CLion. The test set of the experiment, the Pareto Set and the Pareto Front obtained by running the algorithm are as Table 1.

The evaluation criteria of the experimental results are as follows. The point in the PF is assigned 1, and N found paths corresponding to the point which actually existed M paths is regarded as an additional $(N - 1)/M$ score. As shown in the Fig. 5, the winner's score is $2 \times 1 + \frac{4-1}{6} + \frac{3-1}{6} = 2.8$. While the loser's score is $3 \times 1 + \frac{1-1}{6} + \frac{2-1}{6} + \frac{2-1}{6} = 3.3$. Compared with other algorithms, MD-ACO algorithm scores are as Table 2.

Table 1. Problem scale and results

Test problems	Map size	Objectives	Pareto Set	Pareto Front
Problem1	40 * 40	2	9	4
Problem2	40 * 40	3	24	7
Problem3	50 * 50	3	13	4
Problem4	50 * 50	3	9	7
Problem5	84 * 84	5	19	5
Problem6	40 * 40	2	5	3
Problem7	40 * 40	3	16	12
Problem8	50 * 50	4	47	35
Problem9	50 * 50	5	103	79
Problem10	84 * 84	7	1217	1010
Problem11	40 * 40	2	4	2
Problem12	40 * 40	3	11	6

Table 2. The score of the problem1–12

Test problems	MD-ACO	MSCL	ClusteringGA	NAN	INSGA-III
Problem1	5.3	5.3(=)	3.8(-)	5.3(=)	4.3(-)
Problem2	9.532	9.532(=)	7.5(-)	9.532(=)	8.207(-)
Problem3	5.875	5.875(=)	5.875(=)	5.875(=)	3.75(-)
Problem4	8.0	8.0(=)	7.0(-)	8.0(=)	7.0(-)
Problem5	7.583	6.333(-)	3.0(-)	6.333(-)	3.167(-)
Problem6	4.0	4.0(=)	3.5(-)	2.0(-)	3.5(-)
Problem7	14.0	14.0(=)	14.0(=)	12.5(-)	12.0(-)
Problem8	41.0	37.5(-)	42.0(+)	15.5(-)	14.5(-)
Problem9	90.0	77.5(-)	88.0(-)	12.5(-)	28.5(-)
Problem10	1102.5	199.0(-)	0.0(-)	9.0(-)	0.0(-)
Problem11	2.667	2.667(=)	2.0(-)	2.667(=)	1.0(-)
Problem12	8.5	10.5(+)	0.0(-)	7.5(-)	4.0(-)

“+” means the algorithm works better than MD-ACO, “-” means the algorithm works worse than MD-ACO, “=” indicates that the comparison algorithm behaves similarly to MD-ACO.

From the above experimental results, it can be concluded that the evaluation score of the MD-ACO algorithm is higher than other algorithms. Especially for problem 5, problem 9 and problem 10, MD-ACO algorithms can find more solutions than other algorithms.

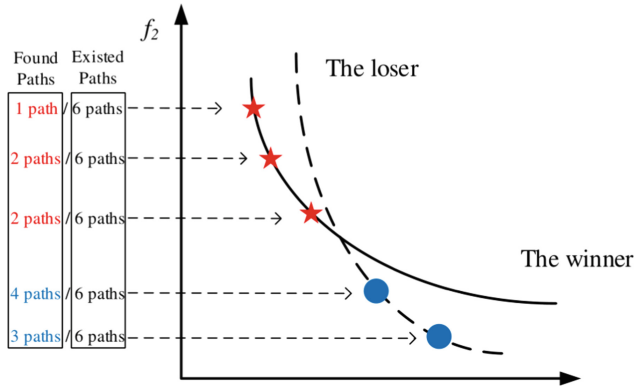


Fig. 5. An example of calculating fractions

6 Conclusions

The main research content of this paper is to solve the MMOPP problems through the MD-ACO. First, matrix preprocessing is performed to process the initial data. Second, Dijkstra is used to optimize the initial pheromone distribution of the ACO. Finally, the ACO is used to traverse the map. The experiment proves the feasibility and effectiveness of the MD-ACO algorithm to solve the MMOPP problems.

Acknowledgment. This work was supported by the National Natural Science Foundation of China (Grant No. 62176191 and 62272355).

References

1. Ma, Y., Hu, M., Yan, X.: Multi-objective path planning for unmanned surface vehicle with currents effects. *ISA Trans.* **75**, 137–156 (2018)
2. Ajeil, F.H., Ibraheem, I.K., Sahib, M.A., et al.: Multi-objective path planning of an autonomous mobile robot using hybrid PSO-MFB optimization algorithm. *Appl. Soft Comput.* **89**, 106076 (2020)
3. Wang, J., Weng, T., Zhang, Q.: A two-stage multiobjective evolutionary algorithm for multi-objective multidepot vehicle routing problem with time windows. *IEEE Trans. Cybern.* **49**(7), 2467–2478 (2018)
4. Wang, J., Zhou, Y., Wang, Y., et al.: Multiobjective vehicle routing problems with simultaneous delivery and pickup and time windows: formulation, instances, and algorithms. *IEEE Trans. Cybern.* **46**(3), 582–594 (2015)
5. Xiang, D., Lin, H., Ouyang, J., et al.: Combined improved A* and greedy algorithm for path planning of multi-objective mobile robot. *Sci. Rep.* **12**(1), 1–12 (2022)
6. Zhang, K., Shen, C.N., Gary, G.Y., et al.: Two-stage double niched evolution strategy for multimodal multi-objective optimization. *IEEE Trans. Evol. Comput.* **25**(4), 754–768 (2021)
7. Liu, J., Ji, J., Ren, Y., et al.: Path planning for vehicle active collision avoidance based on virtual flow field. *Int. J. Automot. Technol.* **22**(6), 1557–1567 (2022)

8. Liu, J., Anavatti, S., Garratt, M., et al.: Modified continuous ant colony optimisation for multiple unmanned ground vehicle path planning. *Expert Syst. Appl.* **196**, 116605 (2022)
9. Lin, J., He, C., Cheng, R.: Adaptive dropout for high-dimensional expensive multiobjective optimization. *Complex Intell. Syst.* **8**(1), 271–285 (2021)
10. Jing, Y., Chen, Y., Jiao, M., et al.: Mobile robot path planning based on improved reinforcement learning optimization. In: *Proceedings of the 2019 International Conference on Robotics Systems and Vehicle Technology*, pp. 479–486 (2021)
11. Korus, K., Salamak, M., Jasinski, M.: Optimization of geometric parameters of arch bridges using visual programming FEM components and genetic algorithm. *Eng. Struct.* **241**, 112465 (2021)
12. Ozsari, S., Uguz, H., Hakli, H.: Implementation of meta-heuristic optimization algorithms for interview problem in land consolidation: a case study in Konya/Turkey. *Land Use Policy* **108**, 105511 (2021)
13. Deb, K., Pratap, A., Agarwal, S., et al.: A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **6**(2), 182–197 (2002)
14. Mao, J.Q.: Research on robot path planning based on improved ant colony algorithm. *Comput. Appl. Softw.* **38**(5), 300–306 (2021)
15. Bullnheimer, B., Hartl, R.F., Strauss, C.: A new rank based version of the ant system—a computational study. *Cent. Eur. J. Oper. Res.* **7**(1), 25–38 (1999)
16. Li, B., Chiong, R., Gong, L.G.: Search-evasion path planning for submarines using the artificial bee colony algorithm. In: *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, pp. 528–535 (2014)
17. Ishibuchi, H., Imada, R., Setoguchi, Y., et al.: Performance comparison of NSGA-II and NSGA-III on various many-objective test problems. In: *2016 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, pp. 3045–3052 (2016)
18. Liang, J., Yue C.T., Li, G.P., et al.: Problem definitions and evaluation criteria for the cec 2021 on multimodal multiobjective path planning optimization. Technical Report (2021)