IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received January 27, 2021, accepted February 10, 2021, date of publication February 22, 2021, date of current version November 24, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3060738

Picking Path Optimization of Mobile Robotic Arm Based on Differential Evolution and Improved A* Algorithm

LIANG CHEN AND HANXU SUN
School of Automation, Beijing University of Posts and Telecommunications, Beijing 100876, China
Corresponding author: Liang Chen (caojietougao@163.com)

ABSTRACT With the development of artificial intelligence technology, robotic arm picking and path finding based on AI technology has attracted more and more attention. How to use the path search algorithm in a relatively short time is a huge challenge in the optimization of robot picking path. Most of the existing path finding methods based on the A* algorithm have the problems of long search time and non-optimal path. Most of them achieve poor results in the optimization of robotic arm picking paths. This paper proposes an AI intelligent path finding method based on differential evolution and improved A* algorithm. The method mainly achieves fast path search by accurately filtering abnormal conditions to achieve a higher convergence rate. The experimental results show that the method proposed in this paper has faster search speed and better path selection than the existing methods. The AI intelligent path finding method based on differential evolution and improved A* algorithm for robotic arm picking is feasible and effective.

INDEX TERMS Artificial intelligence, A* algorithm, differential evolution, intelligent path finding.

I. INTRODUCTION

With the development of artificial intelligence (AI) technology, more and more industries have combined with artificial intelligence to produce better landing application effects, such as path planning, computer vision, service robots, etc., [1]–[3]. Among them, the intelligent path finding method has received more and more attention due to the maturity of artificial intelligence algorithms, and has become a current research hotspot.

With the rapid development of aerospace technology, space robots play an important role in space exploration and on-orbit services due to their ability to perform satellite component assembly, space target capture, and on-orbit maintenance tasks in complex environments.. For the space redundant manipulator system, for a given operation task, path planning is the first problem that needs to be solved [4]. When considering the free-floating mode, the angular momentum of the space manipulator system is not integrable, and the entire system has incomplete characteristics. At this time, even if the angular motions of all joints are the same, when different closed paths are selected, the end pose of the manipulator will not be integrated. the same. In addition, the generalized Jacobian matrix of the inter-manipulator system is not only related to the kinematic parameters, but also closely related to the dynamic parameters, and dynamic singularities will occur during the solution process. Therefore, the path planning of the free-floating space manipulator is very complicated due to the characteristics of nonholonomic constraints and singular dynamics [5]. In addition, during the actual operation, there may be obstacles in the working space of the robotic arm, such as space debris, space cabin peripheral test devices, external truss structures, antennas, solar panels, etc., which will prevent the end of the robotic arm from reaching the feeder. Set the target point, which requires the ability to plan a safe, collision-free motion trajectory of the robotic arm. Therefore, it is of great significance to study the obstacle avoidance planning of the free-floating space manipulator system.

At present, there are three kinds of solutions for dynamic singular avoidance path planning of free floating space robots, which are singular point elimination method [6], singular problem transformation method [7], and singularity avoidance path method [8]. All of them can effectively realize dynamic singular avoidance, but obstacle avoidance is not considered in the study of such problems. Many scholars have

The associate editor coordinating the review of this manuscript and approving it for publication was Shiqi Wang.

also studied the obstacle avoidance planning of manipulator. The methods used can be divided into three categories: geometric model method, artificial potential field method and polynomial interpolation method. Geometric model method mainly has A* algorithm, fast extended random tree, C space method, etc. The A* algorithm [9] is a global optimal planning algorithm, but it requires a lot of scene information and a huge amount of calculation, and the general path planning is not optimal. The fast extended random number method [10] is suitable for multi degree of freedom trajectory planning. It is a step-by-step algorithm based on random sampling. However, this method has some shortcomings, such as poor repeatability, unsmooth path planning, difficult adjustment of planner parameters, and lack of systematic theoretical analysis of the algorithm. The main idea of C-space method [11] is to divide the manipulator workspace into two subspaces: obstacle space and free space, and then use heuristic search algorithm to find a collision free path in the self space of the manipulator. However, it is necessary to reconstruct the C-space for the new working environment, which is difficult to meet the requirements of real-time and universality of the manipulator. Artificial potential field method [12] is a kind of obstacle avoidance method which can be applied in various fields, but its search amount is large, and when there are many gravitational and repulsive fields in Cartesian space, some regions are affected by multiple potential points, resulting in local minima and singularity. Polynomial interpolation method [13] mainly uses the method of finding the middle point to ensure that the end effector avoids obstacles, but does not consider the obstacle avoidance of joints and connecting rods. Although these planning methods can realize the obstacle avoidance planning of manipulator, each has its own shortcomings, and the planning results may not be optimal.

In order to solve the problem that A* algorithm faces in the path finding of robot arm picking, this paper proposes an AI intelligent path finding algorithm based on differential evolution and improved A* algorithm. The algorithm eliminates redundant operations by pruning the A* algorithm [8], [9], so as to achieve real-time requirements and improve the overall global optimization ability through differential evolution algorithm.

II. RELATED THEORIES

A. DIFFERENTIAL EVOLUTION ALGORITHM

Differential evolution algorithm [14] is shown in table 1. Differential evolution algorithm is an effective global optimization method, which is mainly based on population search, and then through crossover, mutation, selection and other operations to obtain the final optimal solution.

B. A* ALGORITHM DEFINITION

A* algorithm [15] is one of the most effective methods to find the shortest path in robot picking. It uses direct search [16]–[18] and uses heuristic function to calculate the minimum cost between the end point and the starting point.

TABLE 1. Differential evolution algorithm.

Method 1: Differential Evolution Algorithm

Input: population: M; Cross factor: D; Number of iterations t

$t \leftarrow 1$
for $i-1$ *to* M **do**
  for $j=1$ *to* D **do**
    $x_{i,t}^{j} = x_{\min}^{j} + rand(0,1) * \left(x_{\max}^{j} - x_{\min}^{j}\right)$;
  end
end
while $\left(\left|f(\Delta)\right| \geq \varepsilon\right)$ *or* $(t \leq T)$ **do**
  for $i=1$ *to* M **do**
    $\Rightarrow (Mutation\ and\ Crossover)$
    for $j=1$ *to* D **do**
      $v_{i,t}^{j} = Mutation\left(x_{i,t}^{j}\right)$;
      $u_{i,t}^{j} = Crossover\left(x_{i,t}^{j}, v_{i,t}^{j}\right)$;
    end
    $\Rightarrow (Selection)$
    if $f\left(u_{i,t}\right) < f\left(x_{i,t}\right)$ **then**
      $x_{i,t} \leftarrow u_{i,t}$;
      if $f\left(x_{i,t}\right) < f(\Delta)$ **then**
        $\Delta \leftarrow x_{i,t}$;
      end
    else
      $x_{i,t} \leftarrow x_{i,t}$;
    end
  end
  $t = t+1$
end
return *the best* Δ

A* algorithm is not only used in robot picking and routing, but also in the shortest path planning of intelligent robot and the best route planning of closed scene truck [20]. Estimation function is the most important step in A* algorithm.

Definition 1: $V = \{v_i | i \in 1, 2, \ldots\}$ represents the set of path nodes passing through the starting point and the end point of the picking point of the manipulator, and represents a node in the picking start point to the end point of the manipulator.

Definition 2: for node a, its cost function B can be defined as follows:

$$f(v_n) = \sigma(v_n) + \rho(v_n) \tag{1}$$

where $\sigma(v_n)$ is the actual cost between the starting point and the current node, and $\rho(v_n)$ is the estimated cost between the current node and the end point.

According to definition 2, cost function $f(v_n)$, also known as distance function, is a kind of heuristic function. In robot picking and routing, cost is distance. The closer the estimated cost $\rho(v_n)$ is to the real value, the stronger the function of the cost function is. When $\rho(v_n) = \sigma(v_n)$, the A* algorithm is changed to Dijkstra algorithm, that is, by increasing the

number of horizontal nodes to improve the path finding ability of the algorithm. At the same time, with the increase of horizontal nodes, its running speed will also slow down, yes, the overall efficiency of the algorithm will slow down. So how to determine the weight of actual cost $\sigma(v_n)$ and estimated cost $\rho(v_n)$ in the total cost function $f(v_n)$ is very important.

$\rho(v_n)$ must satisfy the principle of cost authenticity and cost consistency. Cost authenticity means that the value of $\rho(v_n)$ cannot exceed the real cost of the current node from the target node. The cost consistency principle requires $\rho(v_n)$ to satisfy the conditions shown in formula (2):

$$\rho(v_{i+1}) + \kappa(v_i, v_{i+1}) \geq \rho(v_i) \tag{2}$$

where $\kappa(v_i, v_{i+1})$ is the real cost between node v_i and the next node v_{i+1}, and $\rho(v_{i+1})$ is the estimated cost between the next node v_{i+1} and the final node.

If $\rho(v_i) \leq \rho(v_{i+1}) + \kappa(v_i, v_{i+1})$, then all the paths $f(v_n)$ from the final node are monotone cost increasing functions.

Suppose v_{i+1} is the next target node of v_i, then

$$\begin{aligned} f(v_{i+1}) &= \sigma(v_{i+1}) + \rho(v_{i+1}) \\ &= \sigma(v_i) + \sigma(v_i, v_{i+1}) + \rho(v_{i+1}) \\ &\geq \sigma(v_n) + \rho(v_n) \end{aligned} \tag{3}$$

A* The algorithm is a classical solution to the path finding problem, but when it is applied to the robot arm routing problem, its efficiency and accuracy have a certain conflict, and cannot be controlled by effective means to produce better path planning under low delay. Therefore, this paper proposes an AI intelligent routing algorithm based on differential evolution and improved A* algorithm Path algorithm is used to solve this problem.

C. A* ALGORITHM PRINCIPLE

The A* algorithm is an ordered search algorithm. It searches for the node with the least cost at each step through the function xf)(to find the path with the least cost. Generally, it is necessary to construct two tables: OPEN table and CLOSED table. OPEN table is used for storage For nodes that have been evaluated but have not yet been expanded, the CLOSED table is used to store the expanded points that do not need attention [21]. The A* algorithm cannot determine the final result in the search process. For the path, each node needs to be searched and evaluated and the intermediate value stored in order to finally select the optimal path.

The specific simulation steps are as follows:

Step1: Define two lists named OPEN and CLOSED; OPEN table is used to store valid nodes required by the search path, and CLOSED table is used to store useless nodes;

Step2: A is the starting node, B is the target node, the initial state of the CLOSED table is set to empty, and the starting node A is placed in the OPEN table;

Step3: Look at the point n adjacent to point A (n is called the child node of A, and A is called the parent node of n), the passable points are added to the OPEN table, and their F, G and H values are calculated. Move point A into the CLOSED list;

Step4: Determine whether the OPEN table is empty, if it is, it means the search has failed, if not, proceed to the next step;

Step5: Remove the point n from the OPEN table and add it to the CLOSED table, and judge whether n is the target node B. If it is, it means the search is successful and the algorithm operation ends;

Step6: If not, then expand the search for the child nodes of n: If the child node is not passable or in the CLOSED table, ignore it. If the child node is not in the OPEN table, it will be added to the OPEN table, and the current point is set as its parent node, and the F, G and H values of the point are calculated.

Step7: Jump to **Step4**;

Step8: Save the node after finishing. Find the path from the end point along the direction of the parent node to the start point, which is the optimal path.

The A* algorithm flow chart is shown in Figure 1.

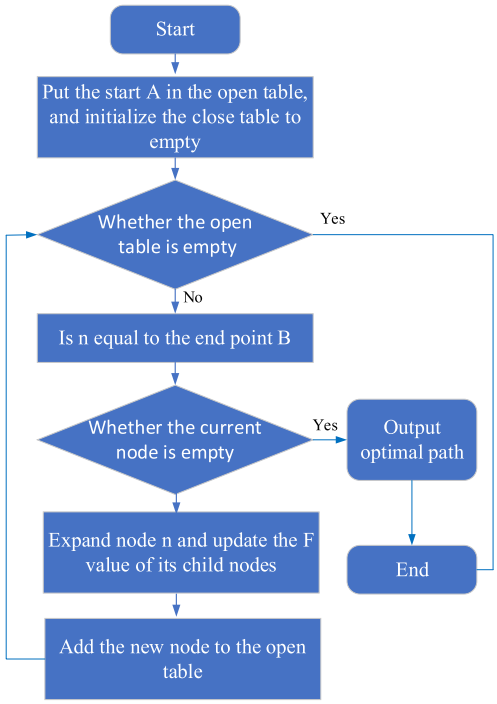

FIGURE 1. Algorithm flow of A* algorithm.

III. COMBINATION OF DIFFERENTIAL EVOLUTION AND IMPROVED A* ALGORITHM

Based on the research of A* algorithm, this paper uses the differential evolution method combined with improved A* algorithm to generate the optimal robot picking path. The main reason is that in the actual application scenario, the NPC of the manipulator picking is real-time changing, and the role online cannot completely determine the unique path according to the initial path planning method. It is necessary to try to detect the dynamic maximum of the distance between

the current path point and the end point Therefore, this paper establishes a replanning algorithm to determine the best path of the game, and considers the optimization of the path and the real-time of the re planning. Because A* algorithm takes a long time and cannot be used as a real-time path re planning algorithm, in order to avoid useless intermediate steps and nodes in the algorithm search space, A* algorithm is pruned, which is called improved A* algorithm. First of all, this paper designs the following cost function of improved A* algorithm. The improved A* algorithm is similar to A* algorithm and needs to design the actual cost and the estimated cost at the same time. As follows:

$\rho\ (v_n)$ is the estimated cost of role P at the selected node of manipulator:

$$\rho\ (v_n) = w_1 * \Gamma\ (P) \tag{4}$$

where, W is the weight coefficient of the actual cost of character P, and is the sum of the distances between adjacent tracks. Γ can be calculated as follows:

$$\Gamma = \sum_{i=0}^{N-1} S_{i.i+1} \tag{5}$$

where $S_{i.i+1}$ is the distance between two adjacent nodes.

$\sigma\ (v_n)$ is the actual cost of character P at the robot picking node v_n. in this paper, it is set as the Euclidean distance between the current node v_n and the target node v_m.

$$\sigma\ (v_n) = \sqrt{(x_n - x_m)^2 + (y_n - y_m)^2} \tag{6}$$

where (x_n, y_n) is the abscissa and ordinate of current node v_n, and (x_m, y_m) is the abscissa and ordinate of target node v_m.

Therefore, the overall cost function $f\ (v_n)$ of the improved A* algorithm in this paper is as follows:

$$f\ (v_n) = w_1 * \Gamma\ (P) + w_2 * \sigma\ (v_n) \tag{7}$$

where w_2 is the weight coefficient of the actual cost function.

The algorithm flow of improved A* algorithm is as follows:

According to the process of improved A* algorithm, the specific improved A* algorithm is shown in table 2:

The improved A* algorithm can find the optimal path better, and can be better reprogrammed in the face of any sudden situation. However, in the process of robot arm picking and routing, if the cost of neighbor paths in both directions is greater than the local optimal path, then the improved A* algorithm will fall into the local optimal situation It is difficult to find the global optimal path of the manipulator, so we introduce the differential evolution algorithm to optimize the improved A* algorithm. By combining the differential evolution with the improved A* algorithm, it overcomes the defect that the differential evolution method cannot re plan in the path finding, and also enhances the ability of the improved A* algorithm in finding the global optimal path. The following is the AI intelligent path finding algorithm based on differential evolution and improved A* algorithm, as shown in algorithm III:

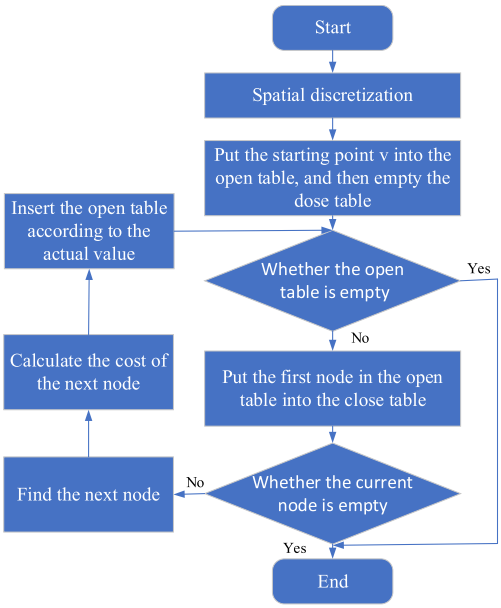

FIGURE 2. Flow of improved A* algorithm.

TABLE 2. A* Algorithm.

Method2：A* Algorithm

Input: Game map data
Set the start and end points v_i and v_j of this Path;
Put the inaccessible points in the Close table, and the start points in Open table;
While Current node is not a target point
Calculate cost of all feasible successors

$$\sigma(v_n) = \sqrt{(x_n - x_m)^2 + (y_n - y_m)^2}$$

$$f(v_n) = w_1 * \Gamma(P) + w_2 * \sigma(v_n)$$

For $i = 1$ to total number of reachable grids
If Nodes are in the Open table and cost less
Update Open table
End
If node v_i not within Open table
Put v_i join Open table
End
End
Sorting the nodes in the Open table by cost;
If Open table $\neq \varnothing$
Put node with smallest cost into Close table
Else
No accessible trail
End
End
Output: the best path

IV. SIMULATION RESULTS AND ANALYSIS

A. EXPERIMENTAL ENVIRONMENT CONFIGURATION

In order to verify the effectiveness and applicability of the robot arm routing method proposed in this paper, we select

TABLE 3. A* Algorithm.

Method3：Game Map Path-Finding Method Based on Differential Evolution and Improved A* Algorithm
Input: Game map data
Use *Differential Evolution* to plan an optimal path $\{v_i,...,v_j\}$ for an NPC from the start point v_i to the goal point v_j;
The NPC advances along $\{v_i,...,v_j\}$, constantly detecting whether NPC has reached the target point as it does so;
If NPC not reach the end
Reprogramming using the *improved A* algorithm*;
Else
Statistical optimal path;
Output: the best path

mountain terrain and urban building complex terrain to simulate the algorithm and the original improved A* algorithm. Table 4 shows the hardware platform and software environment used in this experiment.

TABLE 4. Hardware platform and software environment.

project	Environmental Science
CPU	IntelCore i7-9700K
Graphics card	GTX 2060
Memory	16 G
operating system	Windows 10
software	Matlab

B. SIMULATION EXPERIMENT

In this paper, the proposed algorithm is simulated on mountain terrain. Firstly, we set the starting point and end point of NPC in the experiment, where the starting point of manipulator picking is (0, 0, 0), and then we set the end point of manipulator picking to (100100). In the experiment, we assume that the NPC is moving forward at a constant speed of 3m / s. Fig. 3 and Fig. 4 show the performance of improved A* algorithm in picking and routing of manipulator, which are displayed on 2D plane and 3D actual plane respectively.

In this paper, the improved A* algorithm is simulated on mountain terrain. Similarly, the starting point in the map is (0, 0, 0), and the ending point is also set to (100). Fig. 5 and Fig. 6 show the path finding of the algorithm in this paper, which are the display on the 2D plane and the 3D actual plane respectively.

In this paper, the algorithm is better than the algorithm A* in the case of improved mountains, and it can be seen that the algorithm A* is better than the algorithm A* in the case of improved mountains.

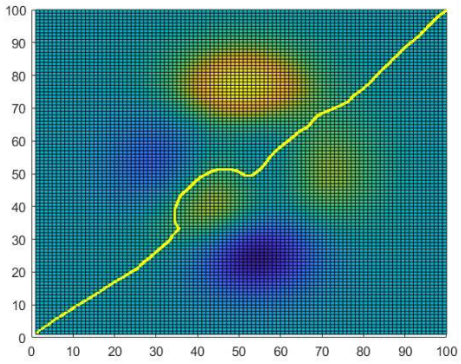

FIGURE 3. Performance of improved A* algorithm on mountain terrain (2D).

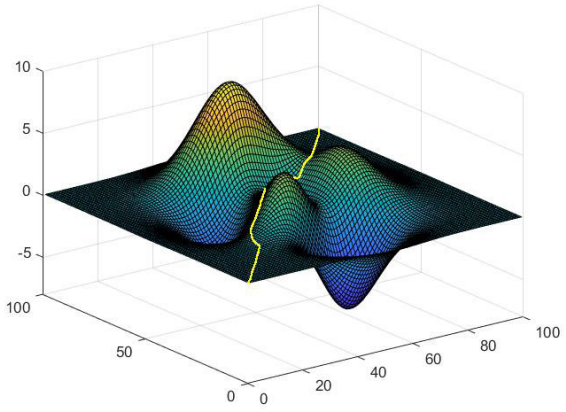

FIGURE 4. Performance of improved A* algorithm on mountain terrain (3D).

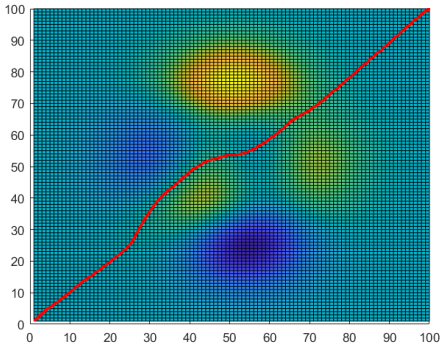

FIGURE 5. The representation of this method on mountain terrain (2D).

In this paper, the improved A* algorithm is simulated on the terrain of urban buildings. Figure 7-10 shows the performance of the improved A* algorithm and the algorithm in this paper, which are respectively displayed on the 2D plane and the 3D actual plane.

As can be seen from figure 7-10, compared with improved A* algorithm, this method can avoid urban buildings earlier in the case of more urban buildings, so it has better path planning.

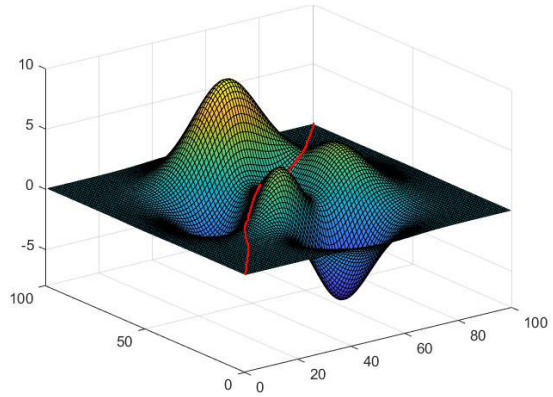

FIGURE 6. The representation of this method on mountain terrain (3D).

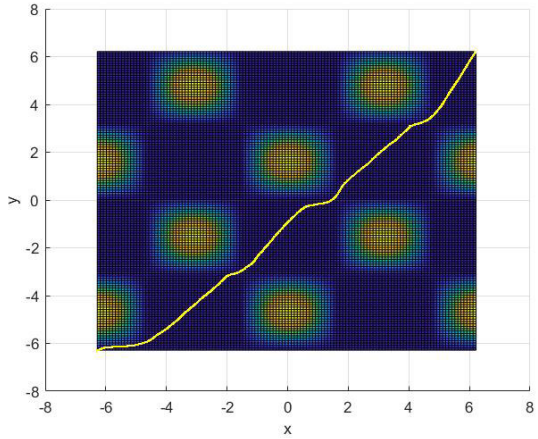

FIGURE 7. Performance of improved A* algorithm on urban complex terrain (2D).

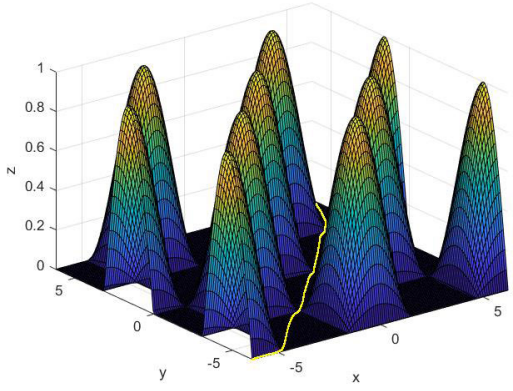

FIGURE 8. Performance of improved A* algorithm on urban building complex terrain (3D).

Next, this paper compares the performance of the proposed algorithm with the improved A* algorithm. Table 5 shows the performance comparison between the proposed algorithm and the improved A* algorithm. It can be seen that the path distance planned by this method is shorter and the total cost is smaller.

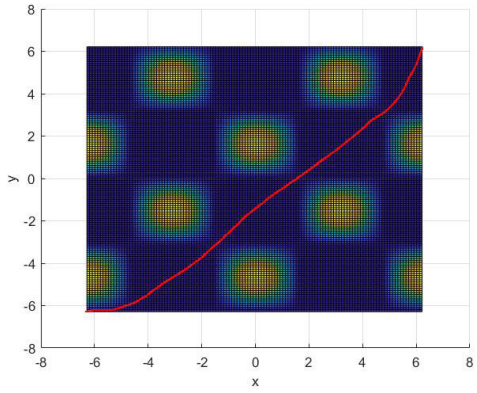

FIGURE 9. The representation of the method in urban complex terrain (2D).

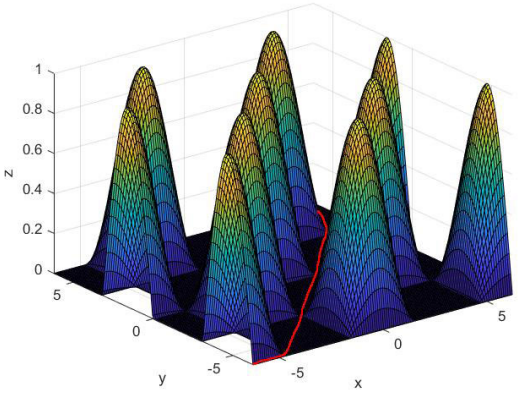

FIGURE 10. The representation of this method on the terrain of urban complex (3D).

TABLE 5. Performance comparison between the proposed method and improved A* algorithm.

Method	Running time	path length	Overall cost
Improved A*	21.97	197.67	247.39
Our Method	22.78	156.52	201.44

C. CONSTRACT EXPERIMENT

The purpose of the experiment design in this article is to verify that the improved A* algorithm is better than the traditional A* algorithm. The simulation environment is a preset grid map. The specific simulation process: first draw a rectangular grid map with matlab tools; Then, the path search is performed through the traditional A* algorithm and the improved A* algorithm; finally, the two parameters of the number of access nodes and the time consumption are compared for analysis.

In order to verify the ideality of the simulation results, a total of 100 sets of experiments were carried out when designing the experiment. In order to reflect the effectiveness of the collected data, the experiment was repeated 5 times,

and then the average value of the 5 simulation data was obtained. Difference test results. In view of the large amount of data, 10 sets of data were selected for comparison and analysis from 100 sets of data. For the A* algorithm before and after the improvement, the corresponding experimental data will be obtained for each path finding. The improvement effect is analyzed by selecting the data related to the number of access nodes and the consumption time, and it is concluded that the improved A* algorithm is effective to realize the path finding design. Table 6 is the traditional A* real algorithm path finding test data.

TABLE 6. Traditional A* algorithm path finding experiment data table.

Number of experital groups	Visit node Number	Path finding time consumption (ms)
1	125	2.74
2	278	4.43
3	458	3.46
4	503	5.20
5	60	6.77
6	580	1.91
7	669	7.43
8	639	8.15
9	871	10.32
10	778	9.21

In order to test the performance of the algorithm under dense and sparse historical trajectories, this paper first divides the road network into dense trajectory areas and sparse trajectory areas according to the distribution of vehicle trajectories, and then selects 10 pairs of starting nodes and 10 pairs of starting nodes and Target node pair (OD pair for short). The 10 OD pairs from the dense trajectory area are numbered from 1 to 10 in the order of the distance between them, and the other 10 OD pairs from the sparse trajectory area are sequentially numbered from the shortest to the longest distance. ∼20 number. In the experiment, 20 OD pairs are used as the input of three path planning algorithms, and then the accuracy, length and travel time of the return path of each OD pair are compared with each algorithm.

1) COMPARISON OF HEURISTIC RESULTS

When traditional A* algorithm uses Euclidean distance formula as heuristic function, it will visit useless nodes many times. This paper proposes differential evolution and improved A* algorithm to solve this problem. In the improved heuristic function of angle cosine, we can choose appropriate weights to reduce the number of invalid nodes in the search process of A* algorithm. In the simulation experiment, the weights of 5 and 12 are selected to test, and the experimental data as shown in Table 6 are obtained.

From the experimental data in Table 7, the angle cosine can be further analyzed as a heuristic function, as shown in Figure 4.

TABLE 7. Units for magnetic properties.

Number of experital groups	Visit node Number	After improvement Number of access nodes(ω=5)	After improvement Number of access nodes (ω=12)
1	125	115	98
2	278	208	187
3	458	138	95
4	503	367	301
5	60	419	348
6	580	61	40
7	669	510	420
8	639	439	364
9	871	637	510
10	778	587	461

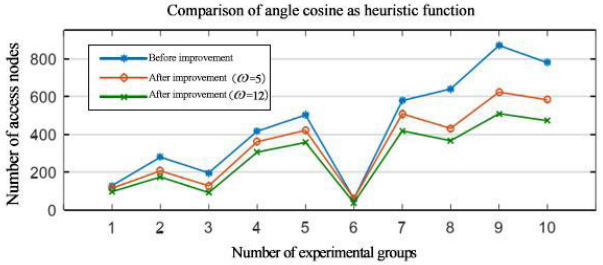

FIGURE 11. T Comparison of angle cosine as heuristic function.

From the comparison results in the above table, it can be seen that the traditional A* algorithm has a large number of nodes to visit in the routing process, while the A* algorithm with improved heuristic function based on angle cosine significantly reduces the number of nodes to be visited in the routing process regardless of the value of weight, and improves the searching efficiency compared with the traditional A* algorithm.

2) COMPARATIVE ANALYSIS OF COMPREHENSIVE IMPROVEMENT RESULTS

This paper mainly improves the traditional A* algorithm from three aspects: firstly, the minimum heap storage structure is used to optimize the data structure of the open table, which effectively improves the efficiency of extracting the minimum F value node; secondly, the hash table is used to optimize the index of the open table, which can reduce the membership judgment complexity of the open table elements from O (n) to 0 (n) Finally, the angle cosine function is used to improve the heuristic function, which effectively avoids the search of useless nodes under the premise of using appropriate weights, and thus improves the search efficiency of A*. By comparing the traditional A* algorithm with the comprehensive improved A* algorithm, the comparison is shown in table 7 Experimental data.

Through the analysis of table 7, we can get the comparison chart of the number of access nodes and the

TABLE 8. Units for magnetic properties.

Number of experital groups	Visit node Number	Patchfinding cost duration (ms)	After improvement Number of access nodes (ω=12)	Improved path finding time-consuming (ω=12)
1	125	2.53	97	2.29
2	278	4.27	172	3.67
3	458	3.89	90	2.81
4	503	5.02	304	4.43
5	60	6.27	358	5.97
6	580	1.89	40	1.89
7	669	7.68	417	6.24
8	639	8.24	365	7.89
9	871	10.58	507	8.45
10	778	9.19	457	7.79

routing consumption time after the comprehensive improvement of A* algorithm as shown in Figure 12.

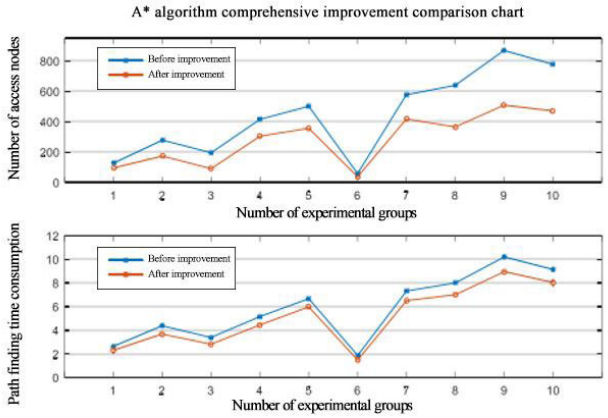

FIGURE 12. T Comparison of A* algorithm before and after comprehensive improvement.

The experimental results show that the improved heuristic A* algorithm is faster than the traditional A* algorithm, and with the increase of the number of search nodes, the optimization range is more obvious, which lays a good foundation for the path planning service in the actual static environment.

D. PATH ACCURACY COMPARISON

The accuracy of the planned path P is defined as the similarity between P and the real path P*. The higher the similarity, the better the accuracy of the path planning algorithm. Use the path similarity function pSim (P, P*) defined in the literature [16] to calculate the similarity between the planned path P and the real path P*. The calculation formula is

$$pSim(P, P^*) = \left(\sum_{e \in P \bigcap P^*} len(e)\right) \times \left(\sum_{e \in P^*} len(e)\right)^{-1}. \quad (8)$$

The path accuracy obtained by the three algorithms for 20 OD pairs is shown in Figure 2. Among them: (A) is the accuracy of the algorithm planning the path during peak time; (B) is the accuracy of the algorithm planning the off-peak time.

It can be seen from Figure 13: 1) Regardless of peak time period or off-peak time period, the accuracy of the 2P++ algorithm is basically the same in the trajectory dense area (the first 10 OD pairs) and the trajectory sparse area (the last 10 OD pairs), A* The algorithm results are similar. It shows that the A* and 2P++ algorithms are relatively stable and are not easily affected by the distribution of historical trajectory data. 2) The accuracy of the L2R algorithm in dense trajectory areas is significantly higher than its accuracy in sparse trajectory areas, indicating that the L2R algorithm is highly dependent on the distribution of historical trajectories, which is consistent with the essence of the L2R algorithm, that is, when there is a trajectory between a pair of nodes When, return to the path contained in the most popular trajectory. At this time, the path obtained by the L2R algorithm must be the most similar to the popular path. Therefore, the L2R has the highest accuracy. When some trajectories between nodes are not popular trajectories or there is no trajectory between them, the L2R infers the paths between them through the preference transfer learning method, and splices them with some of the shortest paths between them, thereby reducing the cost of the L2R algorithm. Accuracy of sparse trajectory area. 3) On the whole, for all 20 OD pairs, the accuracy of the L2R and 2P++ algorithms is higher than that of the A* algorithm, whether it is peak or off-peak hours, indicating that the driving experience extracted from the historical trajectory effectively guides the path Planning: For nodes from dense trajectory areas, the L2R algorithm is more accurate than the 2P++ algorithm; for nodes from sparse trajectory areas, the 2P++ algorithm is more accurate than the L2R algorithm.

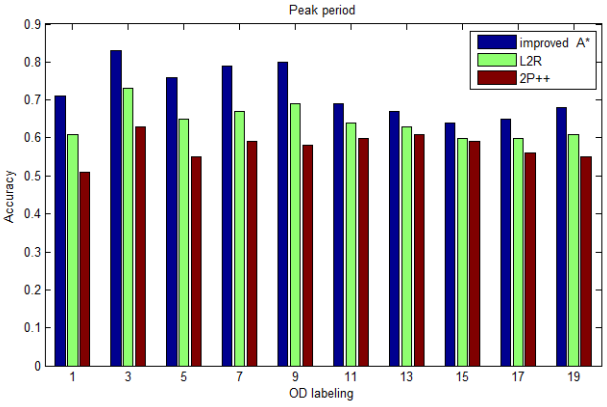

FIGURE 13. Comparison of accuracy of three algorithms.

E. PATH LENGTH COMPARISON

Table 5 lists the average length of the return path of A*, the L2R and 2P++3 algorithms for 20 OD pairs during peak and off-peak hours.

It can be seen from Table 1: 1) Because the historical experience information is not considered, the path distance of the A* algorithm during peak hours and non-peak hours is

TABLE 9. Average path length of the three algorithms.

Method	peak hour
The L2R	9.97
The 2P++	10.67
Our Method	22.78

TABLE 10. Total travel time of three algorithms.

Method	Total travel time
The L2R	579.8
The 2P++	527.0
Our Method	521.1

the same; the paths planned by the L2R and 2P++ algorithms during peak hours are longer than those during non-peak hours. People tend to choose detours during peak hours to avoid road congestion. 2) Compared with the 2P++ and the L2R algorithms, the path of the A* algorithm is the shortest,

The path length calculated by the L2R algorithm during peak hours and off-peak hours is 12% and 7.6% longer than the path length calculated by the A* algorithm, respectively. The path length calculated by the 2P++ algorithm during peak and off-peak hours is relative to the A* algorithm. The calculated path lengths were extended by 7.4% and 4.1%. This is because the L2R and 2P++ algorithms usually choose empirical road sections, so the path is longer. However, the path planned by the 2P++ algorithm is shorter than the path calculated by the L2R algorithm. Compared with the path calculated by the L2R algorithm, the 2P++ algorithm performs better during peak hours and off-peak hours. The path length has been shortened by 4. 2% and 3. 4%.

F. TRAVEL TIME COMPARISON

The travel time of the three algorithms for the planned route of 20 OD pairs is shown in Figure 3. Among them: (A) is the travel time of the three algorithms in the peak time period; (B) is the three algorithms in the off-peak time period The travel time of the planned route. The shorter the travel time of the path, the better the algorithm.

It can be seen from Figure 14: 1) Whether it is peak time or off-peak time, the travel time of the three algorithm planning paths increases with the increase of the distance between the starting point and the destination point. 2) For the off-peak time period (Figure 3(B)), whether it is a node in a dense trajectory area or a node in a sparse trajectory area, when the distance between the starting point and the destination point is short (OD pair 1-5), A* algorithm The travel time of the planned route is less than that of the L2R algorithm and the 2P++ algorithm. However, as the path length increases, the A* algorithm tends to spend more travel time. This is because during off-peak hours, there are fewer vehicles on the road and relatively smooth, and the travel time is more advantageous when the travel distance is shorter. 3) For the peak time period (Figure 3(A)), whether it is a node in a dense trajectory area or a node in a sparse trajectory area, except for the ODs numbered 1, 2 and 11, on the other 17 OD pairs, the L2R and 2P++ algorithms The travel time of both is much better than that of the A* algorithm, and the travel time of the return path of 2P++ and the L2R algorithm is similar.

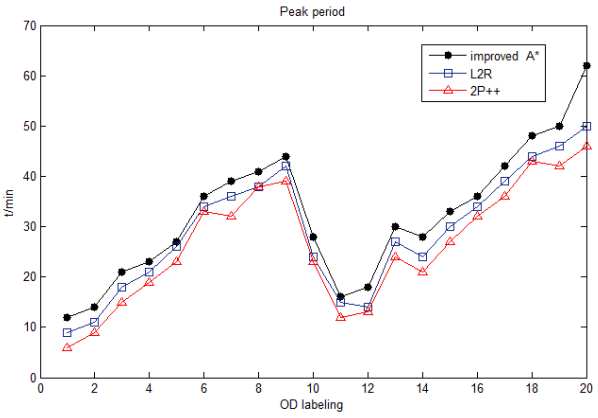

FIGURE 14. Comparison of travel time of three algorithms.

In order to show more accurately the travel time of the planned path obtained by the three algorithms, Table 2 lists the total travel time of the 20 planned paths obtained by the three algorithms. It can be seen from Table 2 that both the 2P++ and THE L2R algorithms are better than the A* algorithm, and the 2P++ algorithm is slightly better than the L2R algorithm, no matter in peak or off-peak hours. Comparing Table 1 and Table 2, it can be seen that the shortest route does not necessarily have a faster travel time. This is mainly because people tend to choose the route with faster road conditions instead of the shortest route during daily travel.

The above results show that in areas with sparse trajectory data, the path accuracy of the 2P++ algorithm is better than that of the L2R algorithm and the A* algorithm; in areas with dense trajectory data, the path accuracy of the 2P++ algorithm is lower than that of the L2R algorithm and higher than that of the A* algorithm; For the average path length, the performance of the 2P++ algorithm is better than that of the L2R algorithm, but lower than the A* algorithm, whether in peak or off-peak hours. For the total travel time, the 2P++ algorithm is better than the A* and the L2R algorithms. Therefore, the 2P++ algorithm is more stable, and its planned path has higher accuracy, shorter driving distance and travel time.

In summary, in view of the problem that the existing path planning methods cannot consider path length and user preferences at the same time, this paper combines the path planning method based on trajectory and the path planning method based on the shortest path, and proposes a new path planning method, namely 2P++ algorithm.

The 2P++ algorithm first uses the LSTM model to train the trajectory data and obtains the user's travel preferences, and then uses the MCMC sampling technology to add the acquired travel preferences to the search process of the A* algorithm, so that the planned route can be more consistent on the basis of a shorter distance User's travel preferences. Theoretical analysis shows that the time complexity of the 2P++ algorithm is the same as that of the A* algorithm. Experimental results show that the 2P++ algorithm is more stable, and its planned path has higher accuracy, shorter travel distance and travel time.

V. CONCLUSION

In recent years, with the rapid development of artificial intelligence technology, it has made important achievements in all aspects. This paper aims at the problem that A* algorithm is easy to fall into local optimum in the picking and routing of manipulator, which leads to the failure to find the optimal path. In addition, due to the problem that differential evolution algorithm can't re plan, this paper proposes a new method of robot arm Picking Based on AI technology. This method is based on the combination of differential evolution and improved A* method, and has achieved good results in robot arm picking and routing. The simulation results show that the proposed algorithm has lower overall cost and shorter path length than improved A* algorithm. Compared with the existing methods, the method in this paper is more suitable for the path finding of manipulator picking.

REFERENCES

[1] E. Weinan and E. Vanden-Eijnden, "Transition-path theory and path-finding algorithms for the study of rare events," *Annu. Rev. Phys. Chem.*, vol. 61, no. 1, pp. 391–420, Mar. 2010.
[2] Y. Liu and L. Jin, "Deep matching prior network: Toward tighter multi-oriented text detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 1962–1969.
[3] S. Jian *et al.*, "Research progress and prospect of fruit and vegetable picking robot," *Acta Agriculturalis Sinica*, vol. 43, no. 5, pp. 158–162, 2006.
[4] J. Li, D. Harabor, P. J. Stuckey, H. Ma, and S. Koenig, "Symmetry-breaking constraints for grid-based multi-agent path finding," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 6087–6095.
[5] A. Alyasin, E. I. Abbas, and S. D. Hasan, "An efficient optimal path finding for mobile robot based on Dijkstra method," in *Proc. 4th Sci. Int. Conf. Najaf (SICN)*, Apr. 2019, pp. 11–14.
[6] F. Hong and Y. H. Rong, "Greedy algorithm and compressed sensing theory," *Acta Autom. Sinica*, vol. 12, no. 12, pp. 1413–1421, 2011.
[7] X. Liu, G. Wei, J.-W. Sun, and D. Liu, "Nonlinear multifunctional sensor signal reconstruction based on least squares support vector machines and total least squares algorithm," *J. Zhejiang Univ.*, vol. 10, no. 4, pp. 497–503, Apr. 2009.
[8] H. Zhao and C. Kit, "A simple and efficient model pruning method for conditional random fields," in *Proc. Int. Conf. (ICCPOL)*, Hong Kong, Mar. 2009, pp. 145–155.
[9] L. He and J. Zhang, "Snowflakes removal for single image based on model pruning and generative adversarial network," in *Proc. IEEE 4th Int. Conf. Image, Vis. Comput. (ICIVC)*, Jul. 2019, pp. 172–176.
[10] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, 1997.
[11] A. R. Leach and A. P. Lemon, "Exploring the conformational space of protein side chains using dead-end elimination and the A* algorithm," *Proteins, Struct., Function, Bioinf.*, vol. 33, no. 2, pp. 227–239, Nov. 1998.
[12] T. G. Kolda, R. M. Lewis, and V. Torczon, "Optimization by direct search: New perspectives on some classical and modern methods," *SIAM Rev.*, vol. 45, no. 3, pp. 385–482, Jan. 2003.
[13] L. A. Silman, Z. Barzily, and U. Passy, "Planning the route system for urban buses," *Comput. Oper. Res.*, vol. 1, no. 2, pp. 201–211, Aug. 1974.
[14] Z. Smilansky, "Miniature autonomous agents for scene interpretation," *J. Sci. Eng.*, vol. 136, no. 5, pp. 393–412, 2020.
[15] X. Zhao, J. Yang, L. Li, W. Zhang, and J. Zeng, "Path tracking control for autonomous underground mining articulated dump truck," *Electrotehn. Electron. Autom.*, vol. 63, no. 3, pp. 75–82, Jul. 2015.
[16] T. Ding, K. H. Chan, Y. Zhou, X.-Q. Wang, Y. Cheng, T. Li, and G. W. Ho, "Scalable thermoelectric fibers for multifunctional textile-electronics," *Nature Commun.*, vol. 11, no. 1, pp. 1–8, Dec. 2020.
[17] Y. Mohammadi and K. Ahmadi, "Single degree-of-freedom modeling of the nonlinear vibration response of a machining robot," *J. Manuf. Sci. Eng.*, vol. 143, no. 5, May 2021, Art. no. 051003.
[18] J. J. Greiner, J. F. Wang, J. Mitchell, S. J. Hetzel, E. J. Lee, and R. L. Illgen, "Opioid use in robotic-arm assisted total knee arthroplasty: A comparison to conventional manual total knee arthroplasty," *Surg. Technol. Int.*, vol. 37, pp. 280–289, Nov. 2020.
[19] L. Lattanzi, C. Cristalli, D. Massa, S. Boria, P. Lépine, and M. Pellicciari, "Geometrical calibration of a 6-axis robotic arm for high accuracy manufacturing task," *Int. J. Adv. Manuf. Technol.*, vol. 111, no. 7, pp. 1813–1829, Dec. 2020.
[20] A. F. Tabak, "Bilateral control simulations for a pair of magnetically-coupled robotic arm and bacterium for *in vivo* applications," *J. Micro-Bio Robot.*, vol. 16, no. 2, pp. 199–214, Dec. 2020.
[21] Y. Zhu, Y. Li, J. Lu, and P. Li, "A hybrid BCI based on SSVEP and EOG for robotic arm control," *Frontiers Neurorobot.*, vol. 14, p. 95, Nov. 2020.

LIANG CHEN was born in Changchun, Jilin, China, in 1982. He received the master's degree from the Beijing University of Posts and Telecommunications, China, where he is currently pursuing the degree with the School of Automation. His research interests include space robot technology, industrial robot technology, and walking robot technology.

HANXU SUN was born in Hanzhong, Shaanxi, China, in 1960. He received the Ph.D. degree from Beihang University, China. He currently works with the Beijing University of Posts and Telecommunications. His research interests include space robot technology, industrial robot technology, walking robot technology, virtual reality technology, and its applications.