AGV Path Planning for Logistics Warehouse by Using an Improved D*Lite Algorithm

Yongyang Zhang, Junhao Luo(✉), Xiaotong Cai, Ying Chen, Engao Peng, and Xinfeng Zou

School of Industrial Automation, Beijing Institute of Technology, Zhuhai, China
1870631048@qq.com

Abstract. In order to solve the problem that the traditional automated guided vehicle (AGV) path planning algorithm has slow convergence speed and is easy to fall into local optimal solution, an improved D*lite algorithm is being presented by combining Dijkstra algorithm and D*lite algorithm. Firstly, the topological map is generated with grid graph. Then, AGV path planning is divided into two stages: pre-planning and real-time planning. In the pre-planning stage, the local shortest path can be achieved through forward searching with Dijkstra algorithm from start vertex to the goal vertex. In the real-time planning stage, the total shortest path can be computed through backward searching with improved D*lite algorithm which searches from the goal vertex to the start vertex and uses heuristics to focus the search, and uses similar ways to minimize having to reorder the priority queue. Finally, the simulation results indicate that the convergence speed and path distance of AGV are optimized by using the improved D*lite algorithm.

Keywords: AGV · Improved D*lite algorithm · Path planning · Simulation

1 Introduction

Automated guided vehicle (AGV) is playing an increasingly important role in the area of logistics due to its high efficiency for material delivering among workstations [1]. The applications of AGV systems face several issues: AGVs number determining, path planning and constraint exerting [2–4]. However, the path planning is most important in ensuring an efficient flow of materials during the storage process.

Several methodologies had been proposed to optimize the AGV path planning distance, such as an improved Dijkstra algorithm was utilized to solve the shortest route [5, 6], a space-time A^* algorithm was developed to handle the path planning at the lower 1evel [7, 8], an improved genetic algorithm was investigated to optimize the running path [9, 10], and an improved ant colony algorithm was implemented to avoid local optimization and accelerating convergence speed [11]. Apparently, there existing various constraints in AGV path planning: collision-free constraints [12–14], time window constraints [15, 16], and distance constraints [17, 18]. For this reason, Mohammad Saidi-Mehrabad et al. built a mathematical model which was composed of Conflict-Free Routing Problem (CFRP) and Job Shop Scheduling Problem (JSSP) to minimize the total completion time [19], Sun Maomao and Kuang Bing discussed a collision free path planning method based on time window for AGV [20], and Wang Dingxin introduced artificial neural network into Q-learning algorithm to find an optimal path without collision by itself through independent learning [21].

However, the genetic algorithm, ant colony algorithm and neural network algorithm have the problem of slow convergence speed. In addition, the Dijkstra and A^* algorithm are easy to fall into local optimal solutions. Therefore, an improved D*lite algorithm for AGV path planning is presented to obtain the total shortest path.

2 Methodology

2.1 Model Assumption

There is an AGV in the logistics warehouse, which is responsible for delivering materials to N locations ($N \geq 1$). For the distribution, the following assumptions are considered for describing the details.

1) AGV travels one route with the predefined path and the fixed speed.
2) AGV starts from the same starting point and comes back to the starting point.
3) Only one delivering task for AGV each time.
4) Loading/unloading time is negligible and traveling path is clean and smooth.
5) AGV is always full of power during delivering materials.
6) AGV cannot be interrupted or recalled during delivering materials.

The work diagram can be stated as Fig. 1 shows:

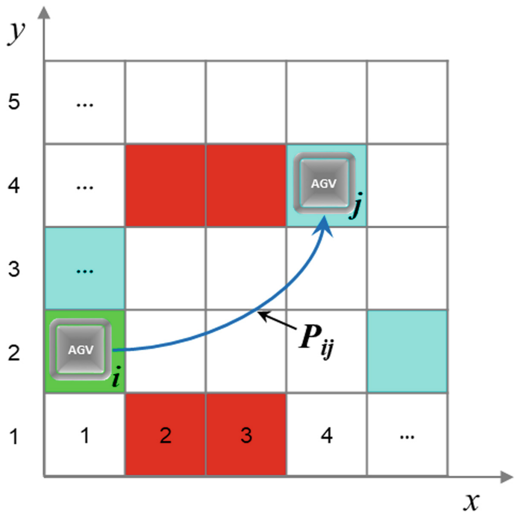

Fig. 1. Work diagram of AGV based on grid map

Figure 1 shows that the green grid is the starting area, the red grids are shelves or obstacles, the blue grids are pick-up locations, and P_{ij} is the traveling route between i and j.

2.2 Model Establishment

The modeling process of the improved D*lite algorithm can be divided into three steps: pre-planning, real-time planning and reach the goal, as Fig. 2 shows.

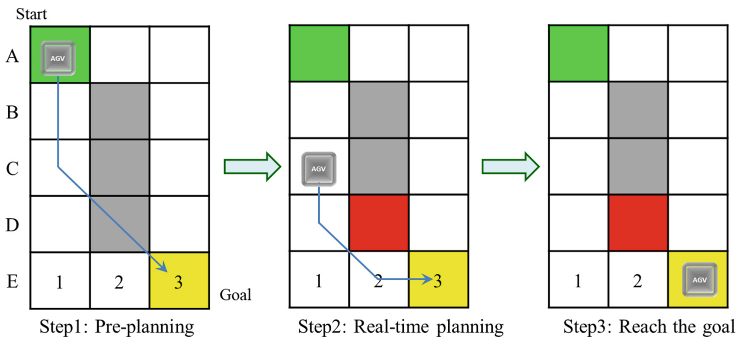

Fig. 2. Schematic diagram of improved D*lite algorithm

The improved D*lite algorithm can be utilized to solve the goal-directed navigation problem in unknown terrain. The terrain is modeled as an eight-connected graph. The costs of its edges are initially one. They change to infinity when the AGV discovers that they cannot be traversed. One can implement the AGV-navigation strategy by applying D* Lite to this graph with s_{start} being the current vertex and s_{goal} being the goal vertex [22].

(1) *In the pre-planning stage*

Dijkstra algorithm is used to compute the local shortest path from start vertex to the goal vertex. And the topological map based on the grid graph is shown in Fig. 3:

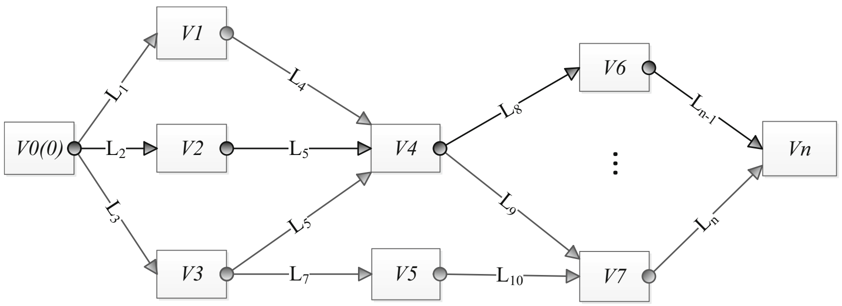

Fig. 3. Topological map based on the grid graph

The vertex v_1 can be labeled as P, so $d(v_1) = 0$, and the vertex v_j ($j = 2, 3, \ldots, n$) can be labeled as T, so $d(v_j) = l_{1j}$.

If $d(v_{j0}) = l_{1j}$, its label T will be updated with label P, and other vertices with label T will be recalculated to select the smallest one as a new label T. Repeat the above steps until all target vertices have been labeled as P. Ultimately, the local shortest path can be computed by using Matlab software.

(2) *In the real-time planning stage*

The improved D*lite algorithm is introduced to compute shortest path through backward searching from the goal vertex to the start vertex until the target vertex is found at this stage [23]. Therefore, the objective function is shown in Invariant (1):

$$g(s) = \begin{cases} 0, & \textit{if } s = s_{start} \\ \min_{s \in pred(s)}(c(s', s) + g(s'), & \textit{otherwise} \end{cases} \tag{1}$$

where S is the finite set of vertices of the graph. $pred(s) \subseteq S$ denotes the set of predecessors of vertex $s \in S$. $0 < c(s', s) \leq \infty$ denotes the cost of moving from vertex s to vertex s'. $g(s)$ denotes the length of a shortest path from s_{start} to s.

The *rhs*-values are one-step previous values based on the g-values and potentially better informed than the g-values. They always satisfy the following relationship as Invariant (2) shows:

$$rhs(s) = \begin{cases} 0, & \textit{if } s = s_{start} \\ \min_{s \in succ(s)}(c(s', s) + g(s')), & \textit{otherwise} \end{cases} \tag{2}$$

In the Invariant (2), $succ(s) \subseteq S$ denotes the set of successors of vertex $s \in S$. A vertex is called locally consistent if its g-value equals its *rhs*-value, otherwise it is called locally inconsistent. If all vertices are locally consistent, then the g-values of all vertices equal their respective start distances. Instead, it uses the heuristic approaches to focus the search and updates only the g-values that are relevant for computing a shortest path (Invariant 3).

$$h(s, s_{start}) = \begin{cases} 0, & \textit{if } s = s_{start} \\ c(s, s') + h(s', s_{goal}), & \textit{otherwise} \end{cases} \tag{3}$$

The priority of a vertex in the priority queue is always the same as its key, which is a vector with two components (Invariant 4 & 5):

$$k_1(s) = \min(g(s), rhs(s)) + h(s_{start}, s) + k_m \tag{4}$$

$$k_2(s) = \min(g(s), rhs(s)) \tag{5}$$

In the Invariant (4), k_m is a variable need to get added up if the AGV moves again and then detects cost changes again. It is usually set to 0 during initialization. Keys are compared according to a lexicographic ordering. The improved D*lite algorithm expands the vertex in the priority queue with the smallest key.

(3) *Reach the goal*

Finally, the variables and parameters are input into the simulation model. After Compute Shortest Path () returns, one can follow a shortest path from s_{goal} to s_{start} by moving from the current vertex s to any successor s' that minimizes $c(s, s') + g(s')$ until s_{goal} is reached. It means that the path planning has been completed.

2.3 Model Solution

The solution process of improved D*lite algorithm is drawn in Fig. 4.

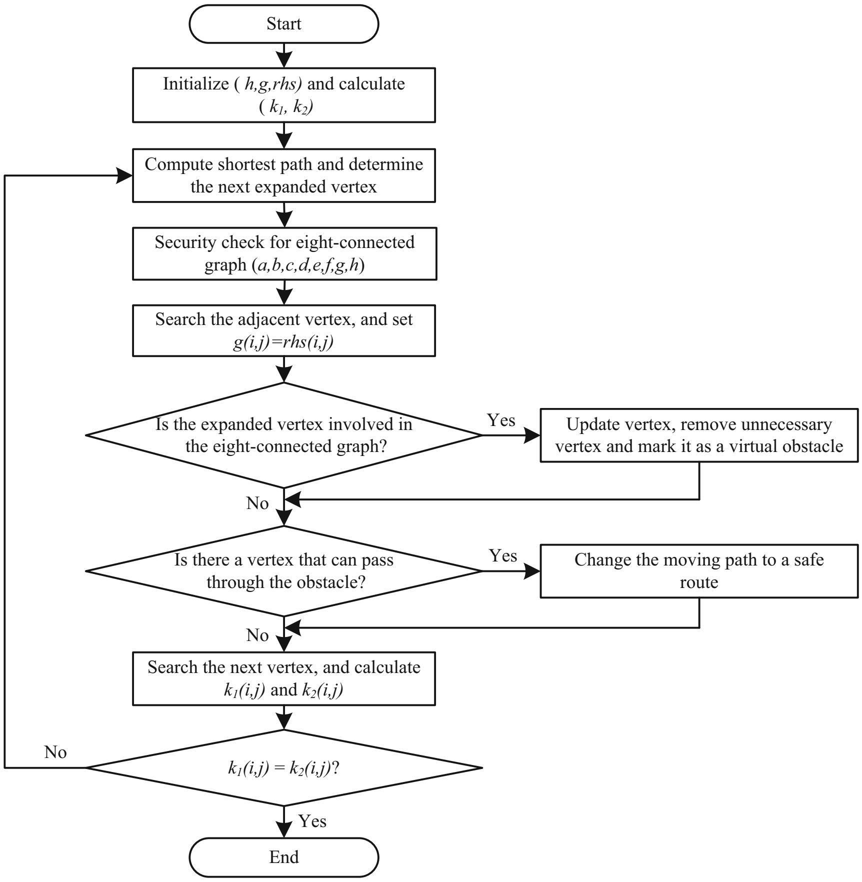
Start
Initialize (h,g,rhs) and calculate (k_1, k_2)
Compute shortest path and determine the next expanded vertex
Security check for eight-connected graph (a,b,c,d,e,f,g,h)
Search the adjacent vertex, and set $g(i,j)=rhs(i,j)$
Is the expanded vertex involved in the eight-connected graph?
Yes
Update vertex, remove unnecessary vertex and mark it as a virtual obstacle
No
Is there a vertex that can pass through the obstacle?
Yes
Change the moving path to a safe route
No
Search the next vertex, and calculate $k_1(i,j)$ and $k_2(i,j)$
No
$k_1(i,j) = k_2(i,j)$?
Yes
End

Fig. 4. Solution process of improved D*lite algorithm

3 Experiments

3.1 Case Description

A logistics warehouse has a length of 11 m and a width of 8 m. There are seven pick-up locations (N1–N7), but only one AGV can be utilized to handle materials with 15 m/min in the logistics warehouse. S1 is the starting area of AGV, and E1 is the ending location (see Fig. 5).

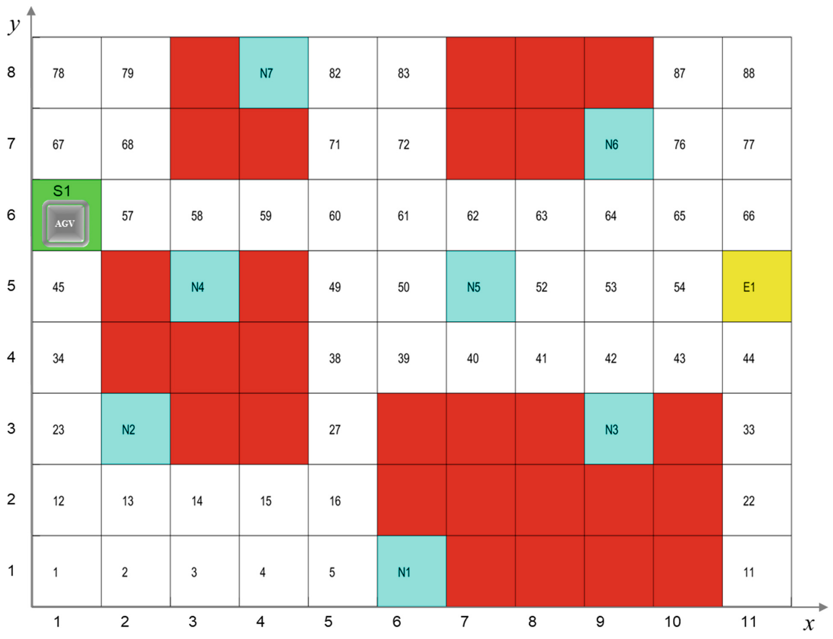

Fig. 5. Grid numbering graph of warehouse

In Fig. 5, each space is a $1 \times 1\ m^2$, and the green grid is the starting area of AGV, red grids are shelves or obstacles, blue grids are pickup locations, yellow grid is the ending position, and white areas are free grids without obstacles.

3.2 Pre-planning for AGV Routing

The topological map of AGV routing is generated with grid graph (see Fig. 6). And the local shortest path from start vertex to the goal vertex can be computed by using Dijkstra algorithm (Red line is the shortest path).

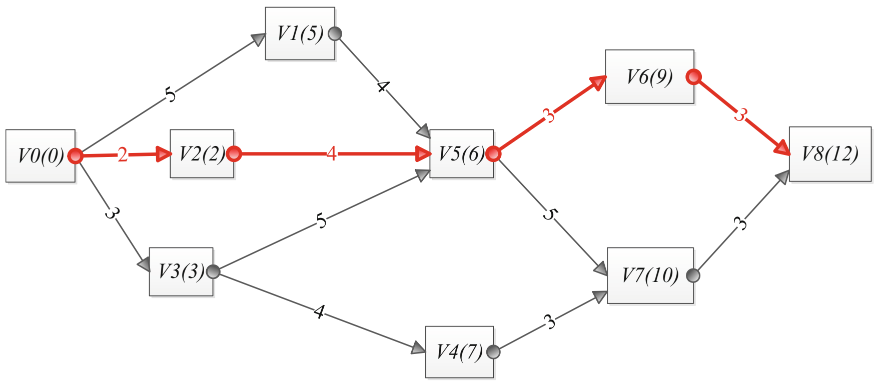

Fig. 6. Topological map of AGV routing

Figure 6 shows that, the adjacent matrix D is established according to the direct distance among vertices.

$$D = \begin{bmatrix} \infty & 5 & 2 & 3 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & 3 & \infty & \infty & \infty \\ \infty & \infty & \infty & 4 & 4 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 2 & \infty \\ \infty & \infty & \infty & \infty & \infty & 3 & 3 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & 3 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & 3 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

Next, the weighted adjacent matrix is inserted into Matlab software, and the simulation results are as follows.

$$path\_all = \begin{bmatrix} 0\,0\,0\,0\,0\,0\,0\,1 \\ 0\,0\,0\,0\,0\,0\,1\,2 \\ 0\,0\,0\,0\,0\,0\,1\,3 \\ 0\,0\,0\,0\,0\,0\,1\,4 \\ 0\,0\,0\,0\,0\,1\,3\,5 \\ 0\,0\,0\,0\,1\,3\,5\,6 \\ 0\,0\,0\,0\,0\,1\,4\,7 \\ 0\,0\,0\,0\,1\,4\,7\,8 \end{bmatrix}$$

Finally, the local shortest path is: 0-1-1-1-3-5-4-7.

3.3 Real-Time Planning for AGV Routing

In the real-time planning stage, a simulation model is built to compute the total shortest path with Matlab software based on the improved D*lite algorithm. In simulation, *S* is the finite set of vertices (*s* = *1, 2, …, 7*). After running the simulation, the total shortest path is marked in blue, and its length is 10 m (see Fig. 7).

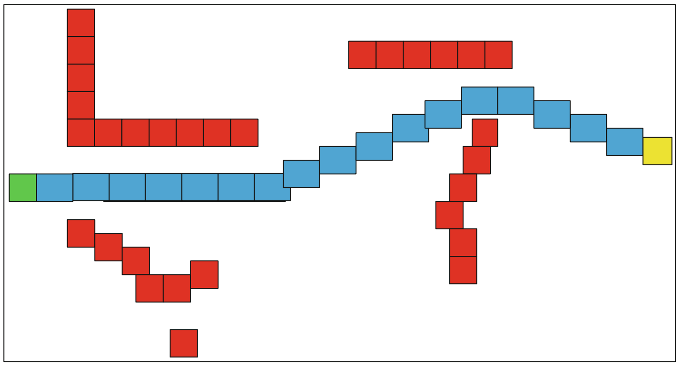

Fig. 7. The total shortest path of AGV

3.4 Experimental Results

The simulation results for two algorithms are drawn in Fig. 8 and Fig. 9.

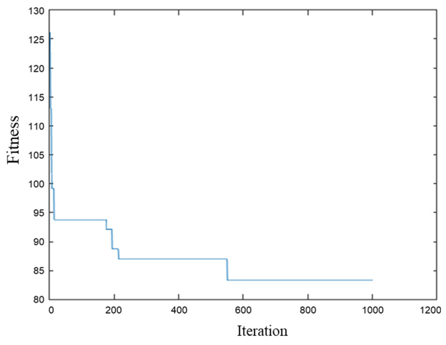

Fig. 8. Dijkstra algorithm

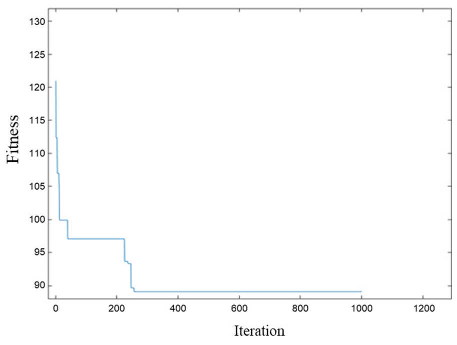

Fig. 9. Improved D*lite algorithm

Figure 8 shows that, at 540 iterations, the maximum distance of AGV is 12 m for the Dijkstra algorithm.

Figure 9 shows that, at 270 iterations, the maximum distance of AGV is 10 m for the improved D*lite algorithm.

Table 1. Comparison of Dijkstra and Improved D*lite algorithm

No	Items	Dijkstra algorithm	Improved D*lite algorithm	Effect
1	Routing (m)	12	10	16.7%↓
2	Duration (s)	48	40	16.7%↓
3	Iterations	540	270	50%↑

From Table 1, it can be seen that the improved D*lite algorithm has achieved several optimizations, such as the convergence speed is 50% faster, the shortest path and duration are reduced by 16.7%.

4 Conclusion

AGV is playing a key role in the area of distribution logistics because of its efficient, accurate and flexible for material handling. However, the traditional AGV path planning algorithm has slow convergence speed and is easy to fall into local optimal solution. Therefore, an improved D*lite algorithm is developed by combining Dijkstra algorithm and D*lite algorithm to solve the problem. In the improved D*lite algorithm, the total shortest path in AGV delivery task can be computed through backward searching. And the experimental results show that the AGV path distance and convergence speed are improved by using the improved D*lite algorithm.

In the following steps, we are going to study the path planning and task scheduling for multiple AGVs with the improved D*lite algorithm.

Acknowledgments. This work is funded by the education and teaching reform project of Beijing Institute of Technology, Zhuhai (No. 2021012JXGG).

References

1. Han, Z., Wang, D., Liu, F., Zhao, Z.: Multi-AGV path planning with double-path constraints by using an improved genetic algorithm. PLoS One **12**(7), 1–16 (2017). e0181747
2. Vivaldini, K., Rocha, L.F., Martarelli, N.J., Becker, M., Moreira, A.P.: Integrated tasks assignment and routing for the estimation of the optimal number of AGVS. Int. J. Adv. Manuf. Technol. **82**(1–4), 719–736 (2015). https://doi.org/10.1007/s00170-015-7343-4
3. Li, B., Liu, H., Xiao, D., Yu, G., Zhang, Y.: Centralized and optimal motion planning for large-scale AGV systems: a generic approach. Adv. Eng. Softw. **106**, 33–46 (2017)
4. Bocewicz, G., Nielsen, I., Banaszak, Z.: Automated guided vehicles fleet match-up scheduling with production flow constraints. Eng. Appl. Artif. Intell. **30**, 49–62 (2014)
5. Liu, Z., Wu, Y., Lu, C., Wang, J.: Application of Dijkstra algorithm in laser guided AGV scheduling system. Mech. Eng. Autom. **2017**(2), 33–34
6. Li, Q., Li, B., Zhang, R, Jiang, T.: Research on AGV path planning based on improved Dijkstra algorithm. Mech. Eng. Autom. (1), 23–25 (2021)
7. Guo, C., Chen, X., Guo, P., et al.: Multi-AGV non-conflict path planning based on space-time A* algorithm. Comput. Syst. Appl. **31**(4), 360–368 (2022)
8. Zhang, D., Sun, X., Fu, S., Zheng, B.: Cooperative path planning in multi-robots for intelligent warehouse. Comput. Integr. Manuf. Syst. **24**(2), 410–418 (2018)
9. Sun, B., Jiang, P., Zhou, G., Dong, D.: AGV optimal path planning based on improved genetic algorithm. Comput. Eng. Des. **41**(2), 550–556 (2020)
10. Li, T., Ning, P., Niu, P.: Factory AGV safety path planning based on improved genetic algorithm. Modul. Mach. Tool Autom. Manuf. Tech. (3), 160–163 (2020)
11. Hu, C., Jiang, P., Zhou, G.: Application of improved ant colony optimization in AGV path planning. Comput. Eng. Appl. **56**(8), 270–278 (2020)

12. Hsueh, C.F.: A simulation study of a bi-directional load-exchangeable automated guided vehicle system. Comput. Ind. Eng. **58**(4), 594–601 (2010)
13. Xin, J.B., Negenborn, R.R., Odewijks, G.: Trajectory planning for AGVs in automated container terminals using avoidance constraints: a case study. IFAC Proc. Vol. **47**(3), 9828–9833 (2014)
14. Miyamoto, T., Inoue, K.: Local and random searches for dispatch and conflict-free routing problem of capacitated AGV systems. Comput. Ind. Eng. **91**, 1–9 (2016)
15. Smolic-Rocak, N., Bogdan, S., Kovacic, Z., Petrovic, T.: Time windows based dynamic routing in multi-AGV systems. IEEE Trans. Autom. Sci. Eng. **7**(1), 151–155 (2010)
16. Alcaidea, D., Chub, C., Katsc, V., Levnerd, E., Sierksmae, G.: Cyclic multiple-robot scheduling with time-window constraints using a critical path approach. Eur. J. Oper. Res. **177**(1), 147–162 (2007)
17. Draganjac, I., Miklić, D., Kovačić, Z., Vasiljević, G., Bogdan, S.: Decentralized control of multi-AGV systems in autonomous warehousing applications. IEEE Trans. Autom. Sci. Eng. **13**(4), 1433–1446 (2016)
18. Bocewicz, G., Banaszak, Z., Nielsen, I.: Multimodal processes prototyping subject to fuzzy operation time constraints. IFAC-Pap. OnLine **48**(3), 2103–2108 (2015)
19. Saidi-Mehrabad, M., Dehnavi-Arani, S., Evazabadian, F., Mahmoodiana, V.: An Ant Colony Algorithm (ACA) for solving the new integrated model of job shop scheduling and conflict-free routing of AGVs. Comput. Ind. Eng. (86), 2–13 (2015)
20. Sun, M., Kuang, B.: Path planning method for AGV dynamic collision avoidance based on time window. Appl. Res. Comput. **39**(1), 54–58 (2022)
21. Wang, D.: AGV path planning based on improved Q-learning algorithm. Electron. Des. Eng. **29**(4), 7–10 (2021)
22. Sven, K., Maxim, L.: D*Lite. In: 18th National Conference on Artificial Intelligence, pp. 476–483. AAAI Press, Menlo Park (2002)
23. CSDN. https://blog.csdn.net/tjcwt2011. Accessed 07 Mar 2022