Recent Developments on Information and Communication Technology (ICT) Engineering- Meen, Yang & Zhao
ISBN: 978-981-14-2136-5

Conflict-Free Path Planning of AGV based on Improved A-star Algorithm

Yuanyuan Cui[1], DanPeng Ma[2], Yang Fang[2], Zhibin Lei[1]

[1] Hong Kong Applied Science and Technology Research Institute Company Limited, [2] Anji Technology Company Limited
[1] Hong Kong, China, [2] Shanghai, China
conicacui@astri.org, madanpeng@anji-tec.com, fangyang@anji-tec.com, lei@astri.org

Abstract

This paper proposes an improved A* algorithm based on time-window to solve the conflict-free path planning problem for Automated Guided Vehicles (AGVs). In particular, a more precise estimate of the turning time is introduced so that a more accurate prediction of node conflict can be achieved during the path planning stage. Furthermore, a more precise estimation of heuristic value in the A* evaluation function is introduced in which it can further speed up the path searching process as some sub-optimal path candidates will not be checked during the process.

Key words: A* algorithm; Path Planning; Time Window; Conflict-Free; AGV

Introduction

With the automation of manufacturing and logistics operations, unmanned Automated Guided Vehicles (AGVs) are now widely used in the warehouse for efficient transfer of goods or materials among different workstations by following the pre-planned path. An efficient AGV path planning method not only can find the shortest path between the source and destination workstations, but also can avoid the possible collision and deadlocks with another AGV while the designated AGV is moving along the pre-planned path.

Various conflict-free routing algorithms have been proposed in the literature a decade ago [1] [2]. Research on conflict-free path planning is still on-going and time-window based method is commonly used for multi-AGV conflict-free path planning [3] – [10]. In [4], Zhang and Yuan proposed to find all the equidistance shortest paths in a rectangular environment map by the improved Dijkstra algorithm (as proposed in [3]) and then the path which does not conflict with other planned AGV paths will be selected. In [5], Yan et al. proposed some rules and online control strategies to avoid AGV collision and deadlock with both unidirectional and bidirectional paths. In [6] [7], Dijkstra algorithm and time- window principle are combined to achieve conflict-free path planning. In [8], Smolic-Rocak et al. proposed a dynamic routing method which uses time windows in a vector form and the predefined candidate paths are checked if they are feasible or not. In [9], Wang et al. proposed to use an improved A* algorithm in path-time model to avoid conflict with other AVG paths. In [10], Jia et al. proposed to use an improved A* algorithm including turning factor for avoiding conflict with other AVG path more accurately.

In this paper, we aim at solving the multi-AGV conflict-free path planning problem in which the AGVs are allowed to move to and from different pickup / loading workstations along the pre-defined grid lines. This mimics the actual situation in the warehouse. With our proposed method, it not only can improve the accuracy of predicted node collision during the route planning stage, but also can improve the efficiency in path planning.

In the following, some basic assumptions and the environment model of the warehouse are described in Section II. Next, the proposed conflict-free path planning algorithm is described in Section III. Some examples to show the validity of the proposed algorithm are given in Section IV. Finally, a conclusion is given in Section V.

Environment Model

In this paper, the environment map of the warehouse is described using topological method. The workstations and moving paths of AGV are regarded as node - arc structure and the topological map describing the work environment of the warehouse system is shown in Fig.1

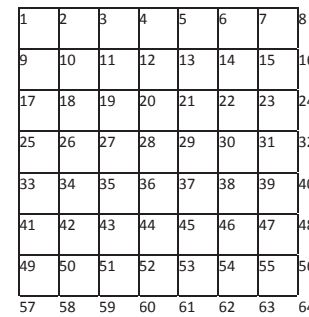


Fig. 1 Environment map of the warehouse.

In this map, the node represents the workstation, which can be a pickup station or an unload station, etc., numbered as 1, 2, 3 ... n. A series of arcs that starts from a node and terminates at the adjacent node represents the travel path of the AGV. The length of the arc represents the distance between two workstations.

Here, AGV is only allowed to move along the grid lines and is allowed to change direction at the nodes. When AGV approaches a node and needs to make a turn, AGV needs to decelerate from normal running speed, *v*, to stationary, then rotate itself to new direction, accelerate from stationary to normal running speed, *v*, again and then move along the grid line in new direction.

doi: 10.35745/icice2018v2.008

Suppose some AGVs have been scheduled already (say AGV1 has been scheduled) When planning the route of AGV2, the schedule and the route of AGV1 remain unchanged, whereas the schedule and/or the route of AGV2 will be chosen so as to avoid the conflict with the route of AGV1.

Conflict-free Path Planning of AGV Based on Improved A-Star Algorithm

A. Background

We are now looking for the conflict-free in time domain, which is different from the traditional routing algorithms in which they are just looking for the shortest distance path from starting node to target node. So, here we need to convert the traditional path model in which the cost function is in distance unit into a path-time model in which the cost function is in time unit. Such path-time model is briefly described as follows.

At each node, sets of parameters are registered. Each parameter set consists of four elements {*atime*, *dtime*, *tNode*, *no*} in which *atime* represents the arrival time at the node, *dtime* represents the departure time at the node, *tNode* represents whether AGV needs to make a turn at this node or not (true or false), and *no* represents AGV identity number. Then, the AGV path planning is equivalent to plan the shortest time line from the starting node to the target node based on the existing time information in the available period.

A* algorithm is a heuristic search algorithm in which it uses a heuristic function to estimate the cost from the transit node to the destination node. A* algorithm evaluation function can be expressed as follows.

$$f(n) = g^*(n) + h^*(n) \tag{1}$$

Wherein, $g^*(n)$ represents the shortest path from starting node n_{start} to the current node n. $h^*(n)$ indicates heuristic value of the shortest path from current node n to the target node n_{target}.

Now, we can change the cost function in travel time as follows:

$$f(n) = g(n) + h(n) \tag{2}$$

Wherein, $g(n)$ indicates the earliest time that the AGV can enter the node n while $h(n)$ represents the time it takes for AGV to move from node n to target node n_{target}. A* algorithm will evaluate different paths from starting node to target node so as to find the path with minimum evaluation cost value.

B. Our Algorithm

In our improved A* algorithm, we propose some ways to obtain $g(n)$ and $h(n)$ as accurate as possible by including AGV deceleration, rotation, and acceleration in making the turn at the node and by considering the actual topology of the warehouse. With more accurate $g(n)$, i.e. we can estimate *atime* of the AGV at this node, we can detect the possible node conflict more accurately during path finding while with more accurate $h(n)$, we can speed up the path finding process.

Before detailing our proposed algorithm, the following terminology is described first.

At first, let us consider $g(n)$, i.e. the earliest time at which the AGV enters node n. By considering different scenarios in which AVG needs to make a turn at this current node n and its parent node n', the time T it takes for the AVG to move from node n' to node n can be obtained as follows.

TABLE I
DESCRIPTION OF NOTATIONS

Notation	Description
n_{start}	start node, x y co-ordinates is (x_{start} , y_{start})
n_{target}	target node, x y co-ordinates is (x_{target} , y_{target})
n	current node, x y co-ordinates is (x_n , y_n)
n'	parent node of n
tNode(n), tNode(n')	flag indicates whether AGV needs to make a turn at node n and n'
t_r	AGV 90-degree rotation time
v	AGV normal running speed
a	AGV acceleration after making a turn
d	AGV deceleration before making a turn
L_a	Distance for AGV to accelerate from stationary to velocity v
L_d	Distance for AGV to decelerate from velocity v to stationary
L	Distance between two neighboring nodes (assumes $L > (L_a + L_d)$)

If AGV does not need to make a turn at both node n' and node n, then T is given as follows.

$$T = \frac{L}{v} \tag{3}$$

If AGV does not need to make a turn at node n' but needs to make a turn at node n, then T is given as follows.

$$T = \frac{L}{v} + \frac{v}{2d} \tag{4}$$

As AGV needs to decelerate from velocity v to stationary when approaching node n.

If AGV needs to make a turn at node n' but does not need to make a turn at node n, then T is given as follows.

$$T = \frac{L}{v} + \frac{v}{2a} + t_r \tag{5}$$

As AGV needs to make 90-degree turn at node n' and then need to accelerate from stationary to velocity v when leaving the node n'.

If AGV needs to make a turn at both node n' and node n, then T is given as follows.

$$T = \frac{L}{v} + \frac{v}{2a} + \frac{v}{2d} + t_r \tag{6}$$

As AGV needs to make 90-degree turn at node n' and then need to accelerate from stationary to velocity v when leaving the node n'. When approaching node n, the AGV needs to decelerate from velocity v to stationary.

Taking into account of these different turning scenarios at node n' and node n and assuming we know the earliest entering

time of AGV at node *n'* is *g(n')*, we can obtain a more accurate estimation of *g(n)* as follows:

$$g(n)=\begin{cases} g(n')+\dfrac{L}{v} & \text{if tNode(n') = false and tNode(n) = false} \\ g(n')+\dfrac{L}{v}+\dfrac{v}{2d} & \text{if tNode(n') = false and tNode(n) = true} \\ g(n')+\dfrac{L}{v}+\dfrac{v}{2a}+t_r & \text{if tNode(n') = true and tNode(n) = false} \\ g(n')+\dfrac{L}{v}+\dfrac{v}{2a}+\dfrac{v}{2d}+t_r & \text{if tNode(n') = true and tNode(n) = true} \end{cases} \quad (7)$$

Next, let us consider *h(n)*, i.e. the estimation of the time it takes for AGV to move from current node *n* to target node n_{target}. The simplest way to estimate the time is to assume the AGV can move directly from current node *n* to target node n_{target} in straight line. But in actual warehouse topology, AGV can only move along the designated grid line. So, in case node *n* and target node n_{target} are not at the same vertical and horizontal level, AGV needs to make at least one turn in order to move from current node to target node. By taking this into account, we can obtain a more accurate estimation of *h(n)* as follows.

If current node *n* and target node n_{target} are at the same horizontal level, then *h(n)* is given as follows.

$$h(n)=\frac{\left|x_{target}-x_n\right|}{v} \quad (8)$$

If current node *n* and target node n_{target} are at the same vertical level, then *h(n)* is given as follows.

$$h(n)=\frac{\left|y_{target}-y_n\right|}{v} \quad (9)$$

If current node *n* and target node n_{target} are not at the same horizonal and vertical level, then *h(n)* is given as follows.

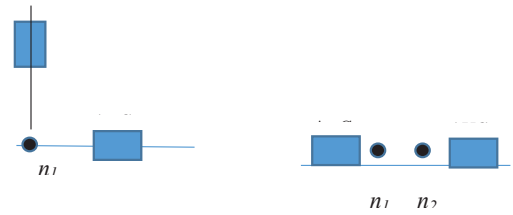
$$h(n)=\frac{\left|x_{target}-x_n\right|}{v}+\frac{\left|y_{target}-y_n\right|}{v}+\frac{v}{2a}+\frac{v}{2d}+t_r \quad (10)$$

Extra turning time, $\dfrac{v}{2a}+\dfrac{v}{2d}+t_r$, is included as AGV needs to make at least one turn at an intermediate node in moving from current node to target node.

In short, a more accurate estimation of *h(n)* is given as follows.

$$h(n)=\begin{cases} \dfrac{\left|x_{target}-x_n\right|}{v} & \text{if } (y_{target}-y_n)=0 \\ \dfrac{\left|y_{target}-y_n\right|}{v} & \text{if } (x_{target}-x_n)=0 \\ \dfrac{\left|x_{target}-x_n\right|}{v}+\dfrac{\left|y_{target}-y_n\right|}{v}+\dfrac{v}{2a}+\dfrac{v}{2d}+t_r & \text{otherwise} \end{cases} \quad (11)$$

C. Conflict Detection



(a) Node conflict can be avoided by waiting (b) Opposite conflict can be avoided by re-planning

Fig. 2 Types of Conflict

By setting *g(n)* appropriately, collision avoidance can be considered during path planning stage as follows.

Consider two types of conflict in multi-AGV system path planning as shown in Fig. 2. The first one is the node conflict as shown in Fig. 2a. It happens when registration time windows of many AGVs (i.e. the time window as specified by *atime*, *dtime* of the AGV) coincides at node n_1. The solution is allowing AGV2 to park and wait, i.e. set $g(n_1)$ of AGV2 as the departure time of AVG1, so that each AGV registration time window does not overlap. The second one is the opposite conflict as shown in Fig. 2b. Several AGVs may also be in a path in the opposite direction. The solution is to re-plan the path for one of the AGVs to avoid the conflict and this can be done by setting $g(n_1)$ of AVG2 as infinity to signify this is an infeasible path.

Algorithm Validation With Examples

As the registration time windows of the AGV play an important role in detecting the AGV collision, so the more accurate is this registration time window, the more accurate will be the collision detection. As the arrival time of the AGV at node *n* can be directly obtained from *g(n)*, by including the turning time in computing this *g(n)* value will also make this registration time window more accurate too.

For *h(n)*, our proposed algorithm would make it closer to the actual value. If we still use the direct straight path from current

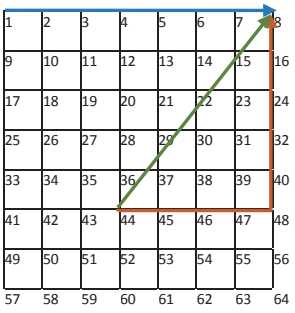


Fig. 3 Example 1 in which the under-estimation of *h(44)* leads to sub-optimal path evaluation.

node to target node in computing this *h(n)* value, such under-estimation of *h(n)* would make the path finding algorithm evaluate more nodes which would not lead to the optimal solution. Examples to illustrate this situation are given

as follows.

With reference to Fig. 3, suppose the target node is node 8, and assume at node 1 and node 44, their *g(.)* values are the same, i.e. *g(1) = g(44)*. Without loss of generality, we assume the time taken for AGV to move from one node to the neighboring node along the grid line is one second.

In case the direct straight path is used in computing *h(.)* values, then *h(1)* = 7 and *h(44)* = 6.4, so node 44 will be evaluated as *f(1) > f(44)*. But, if we use our proposed method in computing *h(.)* values, these *h(.)* values will become *h(1)* = 7 and *h(44)* = 9. (Here, we do not include turning time which does not affect our illustration.) So, this time, node 1 will be evaluated instead as we know node 44 will not lead to an optimal solution.

Considering another example as shown in Fig. 4, suppose the target node is node 8, and assume at node 30 and node 47, their *g(.)* values are *g(30)* = 2.02 and *g(47)* = 1 respectively.

Again, in case the direct straight path is used in computing *h(.)* values, *h(30)* = 3.6 and *h(47)* = 5.1, then *f(30)* = 5.62 and *f(47)* = 6.1. So, node 30 will be evaluated as *f(47) > f(30)*. But, if we use our proposed method in computing *h(.)* values, these *h(.)* values will become *h(30)* = 5 and *h(47)* = 6 and then *f(30)* = 7.02 and *f(47)* = 7. So, this time, node 47 will be evaluated instead as we know node 30 will not lead to an optimal solution.

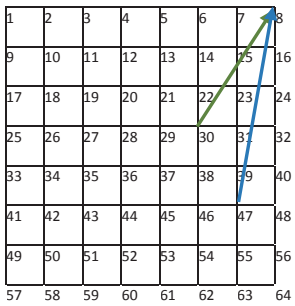


Fig. 4 Example 2 in which the under-estimation of *h(30)* and *h(47)* leads to sub-optimal path evaluation.

Conclusion

In this paper, we proposed a conflict-free path planning of AGV based on improved A* algorithm. This proposed algorithm includes AGV deceleration, rotation and acceleration in making a turn at the node for computing the shortest time path. With more accurate estimation of the earliest arrival time of the AGV at node *n*, i.e. *g(n)*, we can detect the possible node conflict more accurately during path finding. Also, with more accurate estimation of *h(n)*, we can speed up the path finding process since we can avoid the sub-optimal node evaluation as much as possible.

References

[1] S. Maza and P. Castagna, "Conflict-free AGV routing in bi-directional network," *Proceedings of IEEE International Conference on Emerging Technologies and Factory Automation*, pp. 761-764, 2001.

[2] N. Wu and M. Zhou, "AGV routing for conflict resolution in AGV systems," *IEEE International Conference on Robotics and Automation*, pp. 1428-1433, 2003.

[3] G. Qing, Z. Zheng, and X. Yue, "Path-planning of Automated Guided Vehicle based on Improved Dijkstra Algorithm," *29th Chinese Control and Decision Conference*, 2017.

[4] Z. Zhang, Q. Guo, and P. Yuan, "Conflict-free Route Planning of Automated Guided Vehicles Based on Conflict Classification," *IEEE International Conference on Systems, Man, and Cybernetics*, pp. 1459-1464, 2017.

[5] X. Yan, C. Zhang, and M. Qi, "Multi-AGVs Collision-Avoidance and Deadlock-Control for Item-To-Human Automated Warehouse," *International Conference on Industrial Engineering, Management Science and Application*, 2017.

[6] L. He, P. Lou, X. Qian, and R. Liu, "Conflict-free automated guides vehicles routing based on time window," *Computer Integrated Manufacturing Systems*, vol. 16, pp. 2630-2634, 2010.

[7] T. Chen, Y. Sun, W. Dai, W. Tao, and S. Liu, "On the Shortest and Conflict-Free Path Planning of Multi-AGV System Based on Dijkstra Algorithm and the Dynamic Time-Window Method," *Advanced Materials Research*, vol. 645, pp 267-271, 2013.

[8] N. Smolic-Rocak, S. Bogdan, Z. Kovacic, and Tamara Petrovic, "Time Windows Based Dynamic Routing in Multi-AGV Systems," *IEEE Transactions on Automation Science and Engineering*, vol. 7, issue 1, pp. 151-155, 2010.

[9] C. Wang and et al., "Path Planning of Automated Guided Vehicles Based on Improved A-Star Algorithm," *Proceeding of IEEE International Conference on Information and Automation*, pp. 2071-2076, 2015.

[10] F. Jia, C. Ren, Y. Chen, Z. Xu, "A System Control Strategy of a Conflict-free Multi-AGV Routing based on Improved A* Algorithm," *24th International Conference on Mechatronics and Machine Vision in Practice*, 2017.