On the Shortest and Conflict-Free Path Planning of Multi-AGV System Based on Dijkstra Algorithm and the Dynamic Time-Window Method

Tianjian Chen¹,a, Yuan Sun¹, Wei Dai², Wen Tao³ and Shuai Liu³

¹School of Mechanical Science and Engineering, Huazhong Univ. of Sci. and Tech., China
²Department of Control Science and Engineering, Huazhong Univ. of Sci. and Tech., China
³School of Material Science and Engineering, Huazhong Univ. of Sci. and Tech., China

Keywords: The Dynamic Time-Window Method, Dijkstra Algorithm, AGV, Path Planning

Abstract: This paper presents a dynamic routing method for shortest path planning and conflict prevention of AGV system, and then illustrates the principle and implementation of this method. It gives a way to schedule a shortest and conflict-free route by utilizing Dijkstra algorithm and the Dynamic Time-Window Method, and comes to the conclusion by experiments that the combination of these algorithms will solve the routing problem of small-sized multi-AGV system to a large extent.

Introduction

AGV is short for Automated Guided Vehicle, which is a kind of automated material transporting equipment. Nowadays, AGV systems, with many advantages such as highly automaticity, flexibility, and convenience for construction, are gradually applied in automobile industry, medicine industry, textile industry and so on. Undoubtedly, AGV is playing role of great importance to improve the logistic automaticity and manufacturing efficiency.

For AGV systems, path planning is a very important subject. There are some common method including global planning method such as graph algorithm and Genetic Algorithm, and partial planning method such as Virtual-Force Method. [1] Furthermore, the multi-vehicle collaboration and conflict prevention is another research focus, with many theories such as Prediction Method and Reaction Method. [2]

The path planning problem discussed in this paper consists of two parts: firstly, the shortest path planning of a single vehicle regardless of others; secondly, the collision avoidance of many vehicles when they are running simultaneously. Actually, these are two parts of one problem and interact with each other, thus they cannot be separated.

To do our research, we built a small AGV system. The AGVs are optically guided by lines, and their moving distances are recorded by encoders. Also, RFID technology is used for identifying the passing nodes (shown in Fig.3). Therefore, the vehicles can be located accurately. Moreover, the vehicles are connected by a Zigbee wireless star network. The system structure is shown as Fig.1.
Algorithm Overview

**Dijkstra Algorithm.** Dijkstra Algorithm is a typical single-source shortest path searching algorithm, whose feature is the level-by-level searching from the starting point. The flow chart of Dijkstra algorithm is shown as Fig.2. [3] This algorithm is very common in computer-related industrial applications.

The **Classic Time-Window Method.** The classic Time-Window Method is a kind of predictive conflict prevention method. The time window of a vehicle numbered \( u \) in the next \( v \) arc is defined as

\[
W_{uv} = (N_{uv}^{in}, N_{uv}^{out}, t_{uv}^{in}, t_{uv}^{out})
\]

where \( N_{uv}^{in} \) and \( N_{uv}^{out} \) are the entry node and exit node of a certain arc, and \( t_{uv}^{in} \) and \( t_{uv}^{out} \) is the entry time and exit time of an arc. [1,4]

After the vehicle has determined the candidate route, the time window can be calculated based on the speed and distance information, and then the time window overlapping check could be carried out.

The **Dynamic Time-Window Method.** The classic or static Time-Window Method is conducted before the task begins, which will lead to huge accumulated error as the task goes on. To solve this problem, by combining the predictive and reactive strategies, the Dynamic Time-Window Method comes into being. When the vehicle is running on the candidate shortest path, we adopt the look-ahead strategy for the time window overlapping check, calculating and checking the time windows of following few arcs every second, and pushing forward the time window as the vehicle runs.

Computerized Information Expression

**Map Modeling.** The map information consists of two aspects: the weighed graph which expresses the topology and then is used for shortest path searching, and orientation information which expresses location and then is used for vehicle navigation.

We describe the weighed graph using the adjacency matrix. The matrix, denoted as \( D_{N \times N} \), represents the graph of N nodes, and its element \( D_{ij} \) shows the distance between node i and node j (the value \( \infty \) illustrates that these nodes are not directly connected by arcs). [3,4]

Moreover, it is necessary to know the turning direction after the candidate path is determined, so in addition to topology, the locational information is needed in order to match the relative coordinates on vehicle and absolute coordinates of the map. Similarly, we use the orientation matrix \( O_{N \times N} \) to describe the directional relationship. Shown as Fig.3, the exits are numbered 1, 2, 3, 4 from the positive direction of x-axis by rotating counter-clockwisely, and the element \( O_{ij} \) shows which exit should be chosen from node i to node j.
Expression of Vehicle Locating Information. In order to calculate the time window, it is needed to know not only the isolated position on nodes, but also continuous position information between nodes. The locating information of a certain vehicle contains 4 parts:

\[ L = (F, N_{prev}, E_{prev}, D) \]  

Where \( F \) is a flag bit showing whether the vehicle is on a node or on an arc, which can be inferred by the past 2 RFID tag No., \( N_{prev} \) is the previous node No., \( E_{prev} \) is the previous exit No., and \( D \) shows the distance traveled from previous node. Then the locating data can be shared by the wireless network in such format.

Presentation of Routes and Time Windows. As for the route information, by the backtracking of Dijkstra algorithm, all nodes in candidate route can be found. Due to our strategy of looking ahead on the following 2 arcs, it is only needed to know the following 3 nodes. So the route can be stored in a vector of 3 elements where \( N \) is the node No.:

\[ R = \{N_a, N_b, N_c\} \]  

As for the time window \( W_{uv} = (N_{uv}^{in}, N_{uv}^{out}, t_{uv}^{in}, t_{uv}^{out}) \), all the elements in it can be calculated by candidate route, current position, and current speed. Altogether, the time window of \( n \) vehicles in the following 2 arcs can be denoted as time window matrix \( W_{n \times 2} \), the elements of which are \( W_{uv} \) above.

Conflict Prediction Method

Before the conflict prediction, in order to assure the system is running in order, a priority is allocated for every vehicle when the task is assigned, and the priority would get higher and higher as the time goes. Different vehicle has different priority, and the linear list of priority is being refreshed all the time.

For AGV systems, all conflicts can be categorized into 3 types: crossing conflicts, catching-up conflicts, and head-on conflicts. [1, 2] According to actual industrial situation, we assume that all vehicles run at a same speed, and their turning time periods are also equal. Therefore, the catching-up conflicts can be excluded from the problem, and here we only discuss the prediction of crossing conflicts and head-on conflicts.

Prediction of Crossing Conflicts. The crossing conflicts refer to the case that two vehicles arrive at the same crossing at the same time, in which the time window must be very close or even overlap with each other on the conflicting node.

By checking the time window matrix, the crossing conflicts can be found. The time on a certain node of the high-priority vehicle \( t_{uv}^{in} \) and \( t_{uv}^{out} \), generally denoted as \( t_{uv}^* \), can be extracted from the time window matrix. Then, with a proper margin \( \Delta t \), an unsafe time interval \([t_{uv}^* - \Delta t, t_{uv}^* + \Delta t]\) can be created. After that we can check whether the time for other vehicles with lower priority to reach this node, denoted as \( t_{uv'}^* \), is in this time interval. If so, we can identify the crossing conflict. By doing that repeatedly until all elements in time window matrix has been searched, all crossing conflicts can be found. The crossing conflict and its time window graph are shown as Fig.4.
Prediction of Head-on Conflicts. The head-on conflicts refer to the case that two vehicles entered the same arc in opposite direction. In this situation, two conditions must be satisfied: firstly their time window must overlap in this arc at this time, secondly the entry nodes must be different. The overlapping of time window can be judged by the terminal-check method, which checks if $t_{in}^i \in [t_{in}^j, t_{out}^j]$ and if $t_{in}^j \in [t_{in}^i, t_{out}^i]$. Besides, the direction check can be done with the time window matrix. The head-on conflict and its time window graph are shown as Fig.5.

Conflict Prevention Method

After the conflicts are identified, some measures must be taken, and we adopt the following strategies in sequence to prevent collisions.

1. Waiting. If the potential conflicts were crossing conflicts, then the waiting strategy should be adopted, which commands the vehicle with lower priority to stop before the conflicting node and wait for the prior vehicle. At the same time, time window would be modified. However, if the potential conflicts were head-on conflicts, this strategy should not be used. [1]

2. Re-scheduling the path. If strategy 1 could not work, the system would modify the adjacent matrix of the vehicle with lower priority. [1] For crossing conflicts not solved by strategy 1, the weight of the previous arc of the conflicting node shall be set to $\infty$, which means this arc is not available. For head-on conflicts, the conflicting arc should be set to be unavailable, too. After the modification, Dijkstra algorithm should be re-run to search for new solution, and the prevention strategy should be re-run successively from strategy 1.

3. Switching evading rule. If no feasible solution were found after the system re-scheduled the path several times, the evading rule should be switched, which commands the high-priority vehicle to evade low-priority vehicle, and then re-schedule the route as strategy 2 does. However, this switch is only valid at this time, and it would be canceled just after one vehicle passed the potential conflicting area. Also, it is only valid for these two potentially conflicting vehicles, and avoidance with other vehicles shall be coped using the original priority.

4. Stopping and backing off. If the deadlock (mainly caused by head-on conflicts) occurred in severe circumstances, the alarm would ring, and the low-priority vehicle would back off, waiting for the prior vehicle get out of the conflicting area. Then the task will be re-scheduled under human’s remote intervention on the host computer.
Experimental Verification

We built a small-sized AGV system to verify the theories above. The monitoring software is shown as Fig.6. The photo of part of the testing area is shown as Fig.7.

![Fig. 6 GUI of Monitoring Software](image1)

![Fig. 7 A Corner of the Testing Area](image2)

To test the validity of the proposed method, we have carried out several stress tests. During the tests, the host computer assigned the destinations and priorities of the vehicles randomly, and then assigned new tasks ceaselessly after the vehicle reached the destination. Besides, all vehicles are commanded to run at their highest speed (approximately 1.1m/s).

We use the collision/deadlock rate as the evaluating index to show the reliability of the proposed method. [5]

\[
\text{Collision/Deadlock Rate} = \frac{\text{Total Stop Time by Fault}}{\text{Total Running Time}}
\]

The experiment results are listed in Table 1.

<table>
<thead>
<tr>
<th></th>
<th>1st Test</th>
<th>2nd Test</th>
<th>3rd Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Test Time[min]</td>
<td>60</td>
<td>60</td>
<td>60</td>
</tr>
<tr>
<td>Number of Vehicles</td>
<td>3(Appropriate)</td>
<td>4(Appropriate)</td>
<td>5(Excessive)</td>
</tr>
<tr>
<td>Crossing Conflicts Count</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Total Relief Time from Crossing Conflicts [min]</td>
<td>0</td>
<td>0.25</td>
<td>0</td>
</tr>
<tr>
<td>Head-on Conflicts Count</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Total Relief Time from Head-on Conflicts [min]</td>
<td>1.20</td>
<td>1.05</td>
<td>1.75</td>
</tr>
<tr>
<td>Collision/Deadlock Rate</td>
<td>2.00%</td>
<td>2.17%</td>
<td>2.92%</td>
</tr>
</tbody>
</table>

It can be seen from the table that the collision/deadlock rate (especially the crossing conflicts rate) remains relatively low in the 60-minute test, and it goes up as number of vehicles rises. Therefore, the data above indicates that proposed method is effective for conflict prevention (especially for crossing conflicts) of small-sized AGV system.

Conclusion

We adopt Dijkstra algorithm and the Dynamic Time-Window Method to solve the path planning problem for AGV system, with the former to search for shortest path and the latter to avoid conflicts. The overload experiments have verified that the proposed method is effective for a small-sized AGV system. Nevertheless, further study and experiments should be done to ensure the validity and effectiveness of the proposed method for complex and long-time running AGV systems. After further verification and modification, this method would be of great significance to improve the reliability and efficiency of the multi-AGV system for industrial applications.

References

On the Shortest and Conflict-Free Path Planning of Multi-AGV System Based on Dijkstra Algorithm and the Dynamic Time-Window Method

DOI References
http://dx.doi.org/10.1109/TASE.2009.2016350