Int j simul model 19 (2020) 1, 134-145
ISSN 1726-4529 Original scientific paper

DESIGN AND IMPLEMENTATION OF A MULTIPLE AGV SCHEDULING ALGORITHM FOR A JOB-SHOP

Zhao, X. F.; Liu, H. Z.[#]; Lin, S. X. & Chen, Y. K.
College of Electrical Engineering & Intelligentization, Dongguan University of Technology, Dongguan 523808, China
E-Mail: liuhz@dgut.edu.cn ([#] Corresponding author)

Abstract
Considering the actual needs of the job-shop, this paper establishes an experimental platform for automated guided vehicles (AGVs) with a six-wheel dual-drive mechanical structure, and designs a multi-AGV scheduling system for unmanned factories. The scheduling system works in the following steps: Firstly, the deviation of each AGV from the magnetic strip is calculated based on the data of the magnetic sensor, and the speeds of left and right drive wheels are adjusted based on the deviation, keeping the AGV moving stably on the magnetic strip. Next, the A* algorithm is called to plan a collision-free and efficient path. Finally, the conflict points and AGV priorities are determined by comparing the paths of different AGVs, and the paths are planned again if head-on collision may happen on the conflict points. In this way, multiple AGVs can operate coordinately on the same map. The proposed multi-AGV scheduling system was proved feasible through simulation and experiments. Our system can be applied widely in manufacturing factories to replace traditional manual handling and conveyor belt transmission, reduce labour cost, and improve production efficiency.
(Received in September 2019, accepted in January 2020. This paper was with the authors 1 month for 1 revision.)

Key Words: Job-Shop, Automated Guided Vehicles (AGVs), Scheduling Algorithm, Path Planning

1. INTRODUCTION

The manufacturing boom in China has intensified the competition among enterprises. To stand out in the competitive market, many enterprises face the challenge to reduce the cost and enhance the efficiency of production. In fact, material processing only takes up 5-10 % of the production cycle. In most of the time, materials are being stored, loaded, unloaded, and transported, or waiting for processing [1-3]. For a modern enterprise, one of the main ways to reduce production cost is to improve the performance of logistics system and shorten the non-processing time.

The automated guided vehicles (AGVs) have greatly promoted the production efficiency of the manufacturing industry. These vehicles are immensely popular in the market, providing a replacement for porters. Currently, the AGVs have been increasingly applied widely in many industries, ranging from tobacco, medicine, food to chemical.

There are many navigation methods for the AGVs, each with its own merits and applicable scopes. The typical navigation methods include laser guidance, inertial guidance, visual guidance and magnetic guidance. Among them, magnetic guidance is more accurate, cost-effective and stable than other navigation methods. It enjoys great advantages in the relatively fixed path in job-shop. As a result, magnetic guidance is the most popular navigation method in line production of factories.

The coordination control of multiple AGVs has been a hot topic and difficulty in AGV systems. Many scholars have explored the scheduling of multiple AGVs, and presented a number of algorithms, such as traffic control method [4], time window-based path planning [5], and genetic algorithm (GA)-based online scheduling [6, 7]. However, the existing multi-AGV coordinated scheduling strategies are not mature enough. The algorithm design should be further integrated with practical applications.

This paper mainly develops a scheduling system for multiple magnetically-guided AGVs. The system consists of two parts: motion control hardware of the lower computer, and task scheduling software of the upper computer. The hardware was developed in C language on Keil μVision 5, while the software was programmed in C# on Visual Studio (VS) 2017. The software integrates such functions as path planning algorithm, multi-AGV scheduling algorithm, AGV state display and task allocation. The proposed multi-AGV scheduling system was introduced in four aspects (system design, navigation algorithm, path planning algorithm and scheduling algorithm), and proved feasible through simulation and experiment.

2. SYSTEM DESIGN

As shown in Fig. 1, the multi-AGV scheduling system consists of a lower computer and an upper computer. The lower computer encompasses an STM32 controller, a magnetic sensor, a radio frequency identification (RFID) sensor, an infrared obstacle avoidance sensor, a power detection module, a motor drive module (brushless direct current (DC) motor), a wireless serial port and a 24 V battery. The upper computer mainly contains the task scheduling software, and communicates with the lower computer via the wireless serial port.

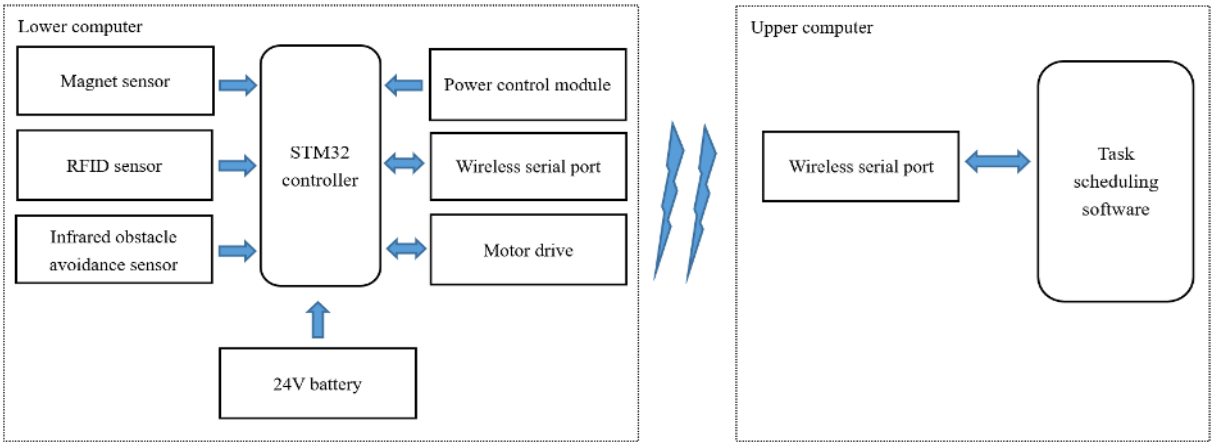

Figure 1: The overall design of the multi-AGV scheduling system.

The STM32 controller receives commands from the upper computer, analyses the data from each sensor, and controls all the actions of the AGVs.

The magnetic sensor monitors the strength of the magnetic field around the magnetic strip of the planned path, converts the deviation of each AGV from the magnetic strip into an 8-bit uniformly distributed data, and transmits the data to the controller, which completes magnetic guidance by the control algorithm.

The RFID sensor reads the radio frequency tags on the map when the AGVs are running, and positions the AGVs in real time.

The infrared obstacle avoidance sensor scans and detects the obstacles in the fan-shaped area right ahead of each AGV, and triggers the interrupt routine of the controller once detecting an obstacle, aiming to prevent the AGVs from collision and protect the machines in the job-shop.

The power detection module displays the power of each AGV, and issues an alarm when the power is below a threshold, alerting the corresponding AGV to be charged.

The wireless serial port links up the AGVs, and connects each AGV with the upper computer.

As shown in Figs. 2 and 3, the self-designed AGV adopts a six-wheel dual-drive mechanical structure. Four universal wheels are installed at the front and rear positions, and two drive wheels are placed at the middle, driven by a brushless DC motor. This simple and

stable structure can be modelled easily. With such a structure, an AGV is enabled to carry heavy objects, and perform various manoeuvres (e.g. going forward, going backward, turning left, turning right and spinning around). Considering the actual application and sensor positions, the self-designed AGV is only allowed to go forward, rather than go backward, under the guidance of the magnetic strip. Besides, spinning is forbidden to avoid damages to the magnetic strip. Thus, the AGV could only turn left or right along the magnetic strip, by making the left and right drive wheels to move at different speeds.

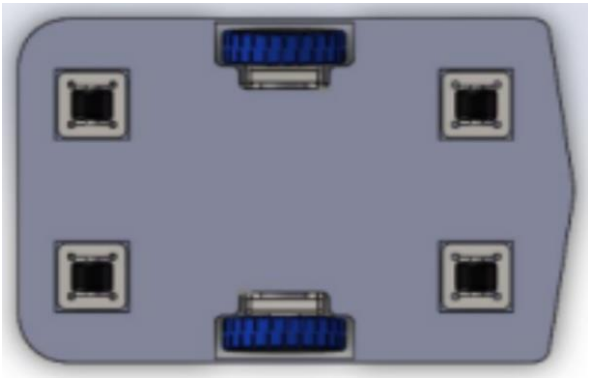

Figure 2: The six-wheel dual-drive structure.

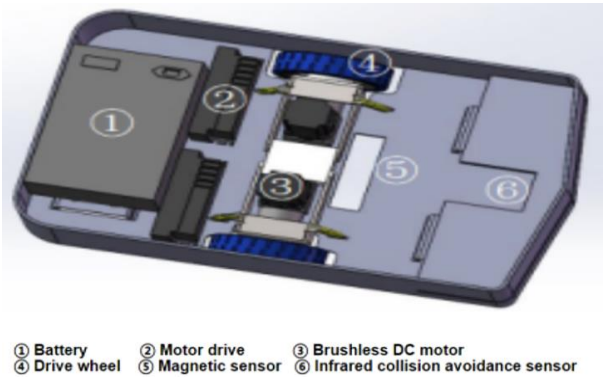

Figure 3: The layout of the self-designed AGV.

The upper computer communicates with all online AGVs via the wireless serial port. The path planning and multi-AGV scheduling are realized by the algorithms in the upper computer. Each AGV only needs to return its working state to the upper computer, and receive the command from the latter.

In actual production, operators can view the working state of each AGV on the multi-AGV scheduling system, and issue a command to the idle AGV closest to the destination, according to the field conditions. Upon receiving the command, the idle AGV will automatically move along the path planned by algorithm, and complete the designated task [8]. Meanwhile, the AGV will feed back its working state in real time to the scheduling system via wireless communication, such that operators could monitor and scheduling the AGVs easily.

3. NAVIGATION ALGORITHM

In this paper, the AGVs are navigated by magnetic guidance, using a D-MNSV6-X8 magnetic sensor. The sensor is connected to the scheduling system via RS232 serial port, and its parameters (e.g. sensitivity) are configurable on the software. The data of each measuring point are outputted through NPN open-drain. The output data can be directly connected to the input/output (I/O) of the STM32 controller, or to the controller via RS232 serial port.

The magnetic guidance system for the AGVs is mainly made up of the magnetic strips on the ground and the magnetic sensor on each AGV. During the production, the magnetic sensor detects the deviation of the AGV from the magnetic strip, and feeds back the signal to the scheduling system. The system will release the control signal to adjust the position of the AGV in time [9, 10].

On the upside, this guidance method is highly flexible: the magnetic strip is easy to deploy, and the path is not difficult to change or expand. On the downside, the magnetic strip is sensitive to metallic materials, prone to interference, and easy to be damaged, calling for regular check and replacement.

Considering the layout of magnetic strip, the navigation algorithm has two modes: single-channel operation and fork-junction operation.

3.1 Single-channel operation

In single-channel operation (Fig. 4), the strength of the magnetic field measured by the magnetic sensor reflects the deviation of the AGV from the magnetic strip, without considering interference.

Figure 4: Single-channel operation.

The controller determines the AGV position relative to the magnetic strip by analysing the data from the magnetic sensor. If the magnetic strip is right below the on-board magnetic sensor, the controller will output a signal, directing the AGV to go straight forward. If the AGV drifts to the left (right) of the strip, the controller will adjust the speeds of the left and right drive wheels according to the degree of deviation, making the AGV turn right (left). The speeds of the two wheels are adjusted continuously, such that the AGV operation converges to the ideal state.

3.2 Fork-junction operation

In actual application, it is inevitable for magnetically-guided AGVs to pass through fork-junctions (Fig. 5). When an AGV operates in a fork-junction, the on-board magnetic sensor could sense the magnetic fields of multiple magnetic strips at the same time. Then, the relative position of the AGV to each strip cannot be reflected correctly by the data of the magnetic sensor. To solve the problem, a common strategy is to close the magnetic navigation function upon entering the fork-junction; the two drive wheels move at different speeds, making the AGV to turn around the fork-junction; then, the magnetic navigation function is turned on again. However, this strategy often causes the AGV to derail, and needs to reset the turning parameters according to the radius of each curve.

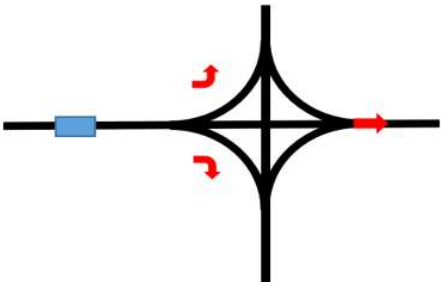

Figure 5: A typical fork-junction.

This paper adds a turning tag (identified by the RFID sensor) at the fork-junction. Take the left turn for example (the turning direction is determined by the path planning algorithm). Upon detecting the turning tag, an AGV will turn left by adjusting the speed difference between the two drive wheels, according to the data of the magnetic sensor. However, the AGV only modifies the turning radius, without changing the turning direction, until the left drift from the magnetic strip is about enough to cause derailment. In other words, the AGV does not pass the fork-junction, until when the magnetic fields are only detectable by the right part of the magnetic sensor, and almost undetectable by the left part. Next, the AGV resumes

the single-channel operation, completing the left turn at the fork-junction. Experimental results show that the proposed strategy works stably, applies to most curves at fork-junctions, reduces the risk of derailment, and ensures the smoothness of operation.

4. PATH PLANNING ALGORITHM

Path planning aims to find a collision-free optimal (or suboptimal) path from the origin to the destination, according to performance indices like shortest path length, shortest travel time and minimum energy consumption [11, 12]. Path planning problems fall into global path planning (environmental information is completely known) and sensor-based local path planning (environmental information is unknown or not completely known). Global path planning is usually solved by A* algorithm [13], Floyd-Warshall algorithm [14] and Dijkstra's algorithm [15], while local path planning is mainly solved by artificial potential field (AFP) method [16], fuzzy logic method [17], genetic algorithm (GA), and artificial neural network (ANN) [18, 19].

In our research, the AGVs operate in a 2D workspace, and the map information is known and fixed throughout the operations. Therefore, the actual map modelled by the grid method, and the path of each AGV was planned by A* algorithm.

The grid method meshes the workspace of the AGV into grids, and stores map information in a 2D array, where 0 means the current grid has no obstacle (the black blocks in Fig. 6) and 1 means the current grid has an obstacle. The grid size directly bears on the accuracy of the map. If the grid is too large, the map stores only a little information, and faces limited resolution and accuracy; if the grid is too small, the map stores lots of information, and enjoys a high accuracy, making path planning more time-consuming.

Figure 6: The map modelled by grid method.

4.1 Overview of A* algorithm

This paper relies on the A* algorithm to plan the path of each AGV. The algorithm is extended from the Dijkstra's algorithm and best first search (BFS). As a heuristic search algorithm, the A* algorithm uses a heuristic function to estimate the distance cost from the origin to the destination on the plane [20]. In heuristic search, every reachable position in the state space is searched and evaluated, and then the new search starts from the best position; this process is repeated until the global best position is found [21]. This search strategy avoids many useless paths, making the algorithm more efficient. The key to the A* algorithm is the heuristic function. The A* parameter of the current node n can be expressed as:

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

where, $g(n)$ is the movement cost to move from the origin to the current node n; $h(n)$ is the estimated movement cost to move from the current node n to the destination. Without considering the obstacles on the path, the h value was estimated by Manhattan distance, i.e. the sum of the traversal and horizontal distances from the current node to the destination:

$$h(n)=\left|X_n\text{-}X_{end}\right|+\left|Y_n\text{-}Y_{end}\right| \tag{2}$$

The greatest merit of the A* algorithm comes from the heuristic function. Thanks to the function, the path search becomes directional (i.e. the search direction intelligently converges to the destination) and rapid (for only a few nodes need to be searched). The specific steps of the A* algorithm are as follows:

Step 1. Initialization: Enter the origin, destination and map information; create an OPEN list and a CLOSE list; place the origin in the OPEN list.

Step 2. Check if the OPEN list is empty. If not, enter the loop; if yes, the path search fails; terminate the program.

Step 3. Traverse the OPEN list, and find the node with the smallest *f* value. Taking the node as the current node, delete it from the OPEN list, and add it to the CLOSE list. If the current node is the destination, the path search completes; exit the loop (jump to Step 6).

Step 4. Search for any neighbouring node that can be reached from the current node from the top, bottom, left and right. Check if the node is in the CLOSE list.

If yes, ignore the node. If no, check if the node is in the OPEN list.

If no, take the current node as its parent node. Calculate the *f*, *g* and *h* of the node: the *f* value is the sum of *g* and *h*; the *g* value is the sum of the *g* value of the parent node and the movement cost of one grid; the *h* value is estimated by the Manhattan distance. If yes, check if the *g* value is smaller when the node is reached via the new path. If no, the new path is not suitable; go to the next step. If yes, the new path is better; change the parent node into the current node, and recalculate the *g* and *f*.

Step 5. If the path search is not complete, return to Step 2.

Step 6. Link up the destination with the origin via the parent node on each level, producing the complete path.

4.2 Simulation of path planning

As mentioned before, the self-designed AGV cannot move backward. Thus, the current direction of the AGV must be considered in path planning. The direction can be manually initialized after the AGV is powered on, or calculated in AGV operations based on the tags captured by the RFID sensor. After introducing the direction parameter, the path planning algorithm needs to be adjusted as follows: a neighbouring node of the origin can be selected if it is reachable in front of the AGV.

Here, the path planning for an AGV is simulated based on the map in the upper computer. The direction, origin and destination of the AGV were initialized, and the path was solved by the A* algorithm. The simulation results are shown in Fig. 7.

Figure 7: The simulation results of the A* algorithm.

The planned paths for different initial directions are compared in Fig. 8. It can be seen that the A* algorithm outputs the shortest path from the origin to the destination, and the planned path between two nodes changes with the initial directions.

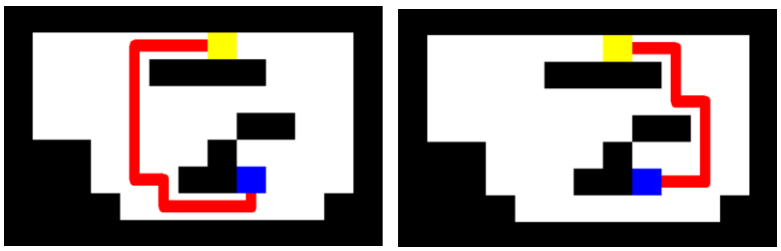

Figure 8: The simulation results for different initial directions.

5. SCHEDULING ALGORITHM

5.1 Overview of AGV scheduling problem

Scheduling is a common problem in the real world. Typical scheduling problems aim to allocate tasks reasonably (with completion sequence sometimes), execute tasks that interfere with each other, and allocate resources in a rational manner. For a logistics system of several AGVs and loading/unloading stations, the AGV scheduling should work efficiently and establish rational connections between the transport tasks and the AGVs [22], such that each AGV can execute tasks without collision or congestion.

If there is only one AGV, the scheduling system only needs to solve the shortest path, because the AGV operation is not interfered by any other AGV. If there are multiple AGVs, the scheduling problem becomes very difficult. The difficulty increases with the number of tasks, the number of AGVs and the complexity of the map. Hence, the scheduling system is the key technique of the AGV industry.

The scheduling system must give overall consideration to the operation of all AGVs. The system should pursue the global optimal solution, rather than the best solutions for several AGVs. In most cases, the global optimal solution does exist, but is difficult to solve under the changing operation conditions. The existing scheduling algorithms are designed based on one or several indices, such as the total travel distance, waiting time and utilization rate of the AGVs, and the number of simultaneously operating AGVs.

Considering the application scenarios, this paper breaks down AGV scheduling into two sub-problems: task allocation (which AGV should be selected to execute a task, in order to maximize system efficiency) and multi-AGV coordination (how to avoid path conflict, deadlock and collision).

5.2 Task allocation

In actual production, the system efficiency can be improved through the rational distribution of tasks. The task allocation covers two parts: task scheduling and AGV selection.

The first step of task scheduling is to set up a task list. The priority of each task was set up manually. Then, the tasks were organized as a waiting queue based on their priorities. The tasks with the same low priority (general tasks) were sorted by their time of issuance. The tasks with high priorities (urgent tasks) were executed with priority.

In essence, a task is to move from the given origin (loading point) to the destination (unloading point). Once the path between the origin and destination is determined, the same effect can be achieved by any AGV. Therefore, the AGV to execute the current task should be selected based on the current position of the AGV and the loading point of the task. In this paper, all idle AGVs are traversed based on the shortest travel distance, and the shortest path

is solved by the A* algorithm according to the current position the AGV and the loading point of the task. The paths of all idle AGVs were compared to find the AGV with the shortest travel distance.

5.3 Multi-AGV coordination

The multi-AGV coordination is a difficult point in AGV scheduling, and the focus of this research. To coordinate the operations of multiple AGVs on the same map, the key lies in solving vehicle collision and path conflict. The main approaches to prevent AGVs from collision include path laying, regional control, predictive collision avoidance, and reactive collision avoidance [23-25].

Path laying divides the path into main and side lanes, and sets up traffic rules for each lane. The path laying algorithm is simple and easy to implement, but not universal.

Regional control avoids conflicts by splitting the workspace into non-overlapping regions. In each region, only one AGV is allowed to complete all the tasks. However, intermediate processing is needed between different regions, suppressing the transport efficiency.

Predictive collision avoidance plans the optimal path for each AGV, and makes offline prediction of conflicts between AGVs based on their paths. If a conflict is possible, the paths of the relevant AGVs will be adjusted to avoid the collision. Predictive collision avoidance offers a promising way to detect and eliminate conflicts, and optimize the entire transport system [26].

Reactive collision avoidance does not plan the path of each AGV in advance. Instead, the AGV path is determined in segments through dynamic planning, in the light of the current state of the system. Reactive collision avoidance responds quickly to changes in system environment. But the algorithm is very complex to design.

In this paper, a multi-AGV scheduling algorithm is developed based on predictive collision avoidance. The AGV collisions were prevented by classifying and marking the conflict nodes. According to the job-shop environment and AGV operations, there are three possible collisions: rear collision, intersection collision and head-on collision.

1) Rear collision

As shown in Fig. 9, if the rear AGV moves at a faster speed than the front AGV, rear collision will occur should no countermeasure be taken.

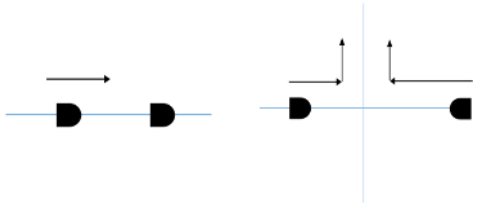

Figure 9: Sketch map of rear collision.

2) Intersection collision

As shown in Fig. 10, if two AGVs arrive at the same node and drives in different directions after passing the node, intersection collision will occur should no countermeasure be taken.

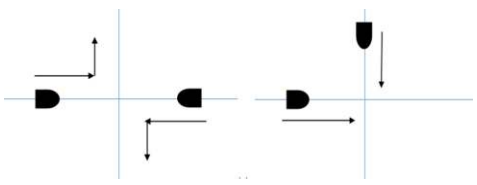

Figure 10: Sketch map of intersection collision.

3) Head-on collision

As shown in Fig. 11, if two AGVs move in opposite directions, head-on collision will occur as only one AGV is allowed to pass through the same path, creating a deadlock.

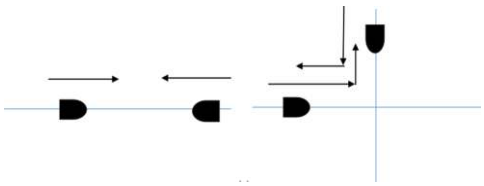

Figure 11: Sketch map of head-on collision.

To prevent rear collision, the scheduling system can order the rear AGV to stop until the front AGV moves out of the safe distance. To prevent intersection collision, the AGVs should be assigned different priorities. Before collision, the low-priority AGV should stop and wait for the high-priority AGV to pass through the intersection. To prevent head-on collision, manual intervention is required instead of ordering an AGV to wait, because the deadlock will cause road congestion. Before operation, the paths of all AGVs must be predicted, the head-on collision nodes be identified, and the paths be planned again to prevent deadlock. Considering the actual operation environment, the following hypotheses were put forward:

Hypothesis 1. All AGVs move at the same speed. In actual operation, some AGVs may need to stop and wait for others. Therefore, the rear collision nodes are not to be ignored.

Hypothesis 2. Before receiving a task, an AGV must check if it has enough power. If the power is low, the AGV should automatically move to the charging spot.

Hypothesis 3. The AGVs do not occupy any node on the map in the idle state, the loading/unloading state and charging state, i.e. special parking spaces are available for these vehicles.

Based on the greedy algorithm, this paper analyses the conflict nodes and coordinates the paths of multiple AGVs in a one-to-many manner. The basic idea of the greedy algorithm is to divide the problem into several sub-problems, find the local optimal solution for each sub-problem, and then combine all local optimal solutions into one solution of the original problem.

For the multi-AGV scheduling problem, the shortest path of an AGV is a local optimal solution. To prevent the collisions between multiple AGVs, the shortest path of each AGV was compared with that of any other AGV to see if there are conflict nodes. The conflict nodes of two paths were saved into a list in sequence. If the neighbouring points are not continuous, the potential conflict is an intersection collision; if the neighbouring points are continuous and the two AGVs move in the same direction, the potential conflict is a rear collision; if the neighbouring points are continuous and the two AGVs move in opposite directions, the potential conflict is a head-on collision. The intersection collision and rear collision were solved by waiting, and the head-on collision was solved by re-planning the paths.

To prevent intersection collision, it is very inefficient to determine which AGV should pass the intersection with priority. Thus, the priorities of the AGVs were classified in advance to reduce the total waiting time and enhance system efficiency:

Step 1. Calculate the conflict nodes between each AGV and all the other AGVs, and classify these conflict nodes.

Step 2. Compare the number of conflict nodes between two AGVs, and assign a high priority to the one with fewer conflict nodes.

Step 3. If the two AGVs have the same number of conflict nodes, compare the number of head-on collision nodes, and assign a high priority to the one with fewer head-on collision nodes.

Through the above steps, the AGV with a relatively smooth path can complete its tasks first.

5.4 Workflow of multi-AGV scheduling algorithm

The operator allocates several tasks on the upper computer, sorts the tasks by priority, and schedules the tasks based on priority.

Firstly, the optimal idle AGV is selected to execute the task with the highest priority. The A* algorithm is called to plan the shortest path, and create a path list of path information and node time. The path list is updated whenever the AGV passes through a path node (monitored by the RFID sensor).

Next, the task with the second highest priority is scheduled by checking if any AGV is idle. If no, the waiting state is initiated. If yes, the optimal AGV is selected to execute the task. The A* algorithm is called to plan the shortest path. The path is compared with the existing path in the map to find the conflict nodes and predict the conflict time. If there is no conflict in time, the conflict nodes are ignored. If no conflict node exists, the path planning is complete. Otherwise, different types of conflicts are solved by the corresponding strategies.

If head-on collision is possible, the conflict section is marked as unreachable in the path planning algorithm. Then, the paths are planned again, and the above process is repeated until no head-on collision exists. The AGV paths are updated constantly in operation. With the elapse of time, the multi-AGV scheduling system will eventually find paths with no head-on collision. Therefore, a timer is set up for the scheduling process. Once the time reaches the deadline, the system will start to schedule the next task. After the paths without head-on collision are solved, the AGVs will enter into operation, the paths will be added to the path list, and the path information will be updated in real time.

If intersection collision is possible, the node before the conflict node is marked in the path list of the AGV with low priority. The AGV will stop at the marked node, and update the node time continuously, until the AGV with high priority passes through the conflict node.

If rear collision is possible, whether the rear AGV needs to wait is determined based on the time information of the rear collision node.

The scheduling is complete after all conflict nodes are predicted and handled. For each remaining task or newly added task, AGV allocation and path planning are conducted based on the task priority. After path planning, the possible conflict nodes are detected. This process is repeated until the tasks of all AGVs are scheduled.

6. CONCLUSIONS

With the development of science and technology, the manufacturing industry is calling for a higher level of automation. The traditional manual handling can no longer satisfy the production demand of factories. As a result, more and more AGVs have been introduced to production lines. This paper firstly illustrates the design of a self-designed magnetically-guided AGV, and then develops a multi-AGV scheduling system on the upper computer, using the A* algorithm for path planning and scheduling. The proposed scheduling method was proved valid through simulation and experiment.

The path planning and scheduling of multiple AGVs are a very complicated project, which is closely correlated with the application scenario. This research only offers a static scheduling strategy to solve the coordinated paths for multiple AGVs, failing to tackle sudden problems in the production process. The further research will attempt to further improve the scheduling algorithm to enhance system efficiency through dynamic scheduling.

ACKNOWLEDGEMENTS

This work was supported by the Science and Technology Project of Guangdong Province under Grant No. 2017B010132001 and the National Major Project under Grant No.2018YFA0404103.

REFERENCES

[1] Guo, Q.; Zhu, S. J.; Chen, J. (2017). Parameter tuning of linear active disturbance rejection controller based on chaotic quantum behaved particle swarm optimization, *Proceedings of the 2017 29th Chinese Control And Decision Conference*, 7484-7489, doi:10.1109/CCDC.2017.7978539

[2] Hassani, K.; Lee, W.-S. (2016). Multi-objective design of state feedback controllers using reinforced quantum-behaved particle swarm optimization, *Applied Soft Computing*, Vol. 41, 66-76, doi:10.1016/j.asoc.2015.12.024

[3] Fazlollahtabar, H.; Saidi-Mehrabad, M.; Masehian, E. (2015). Mathematical model for deadlock resolution in multiple AGV scheduling and routing network: a case study, *Industrial Robot*, Vol. 42, No. 3, 252-263, doi:10.1108/ir-12-2014-0437

[4] Wu, W.; Zhang, F.; Liu, W.; Lodewijks, G. (2020). Modelling the traffic in a mixed network with autonomous-driving expressways and non-autonomous local streets, *Transportation Research Part E: Logistics and Transportation Review*, Vol. 134, Paper 101855, 22 pages, doi:10.1016/j.tre.2020.101855

[5] Sidoti, D.; Avvari, G. V.; Mishra, M.; Zhang, L.; Nadella, B. K.; Peak, J. E.; Hansen, J. A.; Pattipati, K. R. (2017). A multiobjective path-planning algorithm with time windows for asset routing in a dynamic weather-impacted environment, *IEEE Transactions on Systems, Man and Cybernetics: Systems*, Vol. 47, No. 12, 3256-3271, doi:10.1109/TSMC.2016.2573271

[6] Zhao, P. X.; Gao, W. Q.; Han, X.; Luo, W. H. (2019). Bi-objective collaborative scheduling optimization of airport ferry vehicle and tractor, *International Journal of Simulation Modelling*, Vol. 18, No. 2, 355-365, doi:10.2507/IJSIMM18(2)CO9

[7] Mousavi, M.; Yap, H. J.; Musa, S. N.; Tahriri, F.; Dawal, S. Z. M. (2017). Multi-objective AGV scheduling in an FMS using a hybrid of genetic algorithm and particle swarm optimization, *PloS ONE*, Vol. 12, No. 3, Paper e0169817, 24 pages, doi:10.1371/journal.pone.0169817

[8] Cservenák, Á. (2018). Further development of an AGV control system, Jamai, K.; Bollo, B. (Eds.), *Vehicle and Automotive Engineering 2*, Springer, Cham, 376-384, doi:10.1007/978-3-319-75677-6_32

[9] Wu, W.; Huang, L.; Du, R. (2020). Simultaneous optimization of vehicle arrival time and signal timings within a connected vehicle environment, *Sensors*, Vol. 20, No. 1, Paper 191, 17 pages, doi:10.3390/s20010191

[10] Sheikh, T. A.; Bora, J.; Hussain, A. (2019). Performance analysis of massive multi-input and multi-output with imperfect channel state information, *Traitement du Signal*, Vol. 36, No. 4, 361-368, doi:10.18280/ts.360409

[11] Muthukumaran, S.; Sivaramakrishnan, R. (2019). Optimal path planning for an autonomous mobile robot using dragonfly algorithm, *International Journal of Simulation Modelling*, Vol. 18, No. 3, 397-407, doi:10.2507/IJSIMM18(3)474

[12] Kim, D. H.; Hai, N. T.; Joe, W. Y. (2017). A guide to selecting path planning algorithm for automated guided vehicle (AGV), Duy, V.; Dao, T.; Zelinka, I.; Kim, S.; Phuong, T. (Eds.), *AETA 2017 – Recent Advances in Electrical Engineering and Related Sciences: Theory and Application: Lecture Notes in Electrical Engineering*, Vol. 465, Springer, Cham, 587-596, doi:10.1007/978-3-319-69814-4_56

[13] Liu, L. (2018). Occupational therapy in the fourth industrial revolution, *Canadian Journal of Occupational Therapy*, Vol. 85, No. 4, 272-283, doi:10.1177/0008417418815179

[14] Wolfram, M.; Schlegel, S.; Westermann, D. (2017). Closed loop flow detection in power systems based on Floyd-Warshall algorithm, *Proceedings of the 2017 IEEE Manchester PowerTech*, 6 pages, doi:10.1109/PTC.2017.7980880

[15] Khachiyan, L.; Gurvich, V.; Zhao, J. (2006). Extending Dijkstra's algorithm to maximize the shortest path by node-wise limited arc interdiction, Grigoriev, D.; Harrison, J.; Hirsch, E. A.

(Eds.), *Computer Science – Theory and Applications, CSR 2006: Lecture Notes in Computer Science*, Vol. 3967, Springer, Berlin, 221-234, doi:10.1007/11753728_24

[16] Zhang, C. (2018). Path planning for robot based on chaotic artificial potential field method, *Proceedings of the 4th International Conference on Advanced Engineering and Technology*, Paper 012056, 4 pages, doi:10.1088/1757-899X/317/1/012056

[17] Zhang, D. (2017). High-speed train control system big data analysis based on fuzzy RDF model and uncertain reasoning, *International Journal of Computers Communications & Control*, Vol. 12, No. 4, 577-591

[18] Wan, X.; Hu, W.; Zheng, B.; Fang, W. (2015). Robot path planning method based on improved ant colony algorithm and Morphin algorithm, *Science & Technology Review*, Vol. 33, No. 3, 84-89, doi:10.3981/j.issn.1000-7857.2015.03.014

[19] Sánchez-Escalona, A. A.; Góngora-Leyva, E. (2018). Artificial neural network modeling of hydrogen sulphide gas coolers ensuring extrapolation capability, *Mathematical Modelling of Engineering Problems*, Vol. 5, No. 4, 348-356, doi:10.18280/mmep.050411

[20] Fu, B.; Chen, L.; Zhou, Y.; Zheng, D.; Wei, Z.; Dai, J.; Pan, H. (2018). An improved A* algorithm for the industrial robot path planning with high success rate and short length, *Robotics and Autonomous Systems*, Vol. 106, 26-37, doi:10.1016/j.robot.2018.04.007

[21] Croft, D. W. (2004). A* algorithm, Croft, D. W. (Ed.), *Advanced Java Game Programming*, Apress, Berkeley, 347-385, doi:10.1007/978-1-4302-0713-9_8

[22] Jiang, L.; Peng, G.; Xu, B.; Lu, Y.; Wang, W. (2018). Foreign object recognition technology for port transportation channel based on automatic image recognition, *EURASIP Journal on Image and Video Processing*, Vol. 2018, No. 1, Paper 147, 9 pages, doi:10.1186/s13640-018-0390-7

[23] Bogdan, S.; Puncec, M.; Kovacic, Z. (2003). The shortest path determination in AGV systems by using string composition, *Proceedings of the 2003 IEEE International Conference on Industrial Technology*, 984-989, doi:10.1109/ICIT.2003.1290795

[24] Ko, J. P.; Jung, J. W.; Jeon, J. W. (2013). Anti-collision method for AGV using RFID and ZigBee network, *Proceedings of the 13th International Conference on Control, Automation and Systems*, 599-604, doi:10.1109/ICCAS.2013.6703938

[25] Huo, K. G.; Zhang, Y. Q.; Hu, Z. H. (2016). Research on scheduling problem of multi-load AGV at automated container terminal, *Journal of Dalian University of Technology*, Vol. 56, No. 3, 244-251, doi:10.7511/dllgxb201603004

[26] Liu, S.; Sun, D.; Zhu, C. (2014). A dynamic priority based path planning for cooperation of multiple mobile robots in formation forming, *Robotics and Computer-Integrated Manufacturing*, Vol. 30, No. 6, 589-596, doi:10.1016/j.rcim.2014.04.002