APEM journal
Advances in Production Engineering & Management
Volume 16 | Number 1 | March 2021 | pp 37–46
https://doi.org/10.14743/apem2021.1.383
ISSN 1854-6250
Journal home: apem-journal.org
Original scientific paper

Multi-objective automated guided vehicle scheduling based on MapReduce framework

Shi, W.[a,b], **Tang, D.B.**[a,*], **Zou, P.**[b]

[a]College of Mechanical and Electrical Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, P.R. China
[b]Beijing Aerospace Smart Manufacturing Technology of Development Co., Ltd., Beijing, P.R. China

ABSTRACT

During material handling processes, automated guided vehicles (AGVs) pose a path conflict problem. To solve this problem, we proposed a multi-objective scheduling model based on total driving distance and waiting time, and used the A* path planning algorithm to search the shortest path of AGV. By using a speed control strategy, we were able to detect the overlap path and the conflict time. Additionally, we adopted an efficient MapReduce framework to improve the speed control strategy execution efficiency. At last, a material handling system of smart electrical connectors workshop was discussed to verify the scheduling model and the speed control strategy combined with the MapReduce framework is feasible and effective to reduce the AGV path conflict probability. The material handling system could be applied in workshop to replace manual handling and to improve production efficiency.

ARTICLE INFO

Keywords:
Automated-guided vehicle(AGV);
Scheduling;
AGV scheduling;
MapReduce;
Path planning;
A* search algorithm

**Corresponding author:*
d.tang@nuaa.edu.cn
(Tang, D.B.)

Article history:
Received 25 January 2021
Revised 2 March 2021
Accepted 8 March 2021

1. Introduction

Automated guided vehicles (AGVs) are automated intelligent unmanned vehicles that rely on electromagnetic, laser, and visual navigation equipment to complete a series of transportation and assembly tasks under the control of a scheduling system along a guided path between workstations. In the actual production environment, through cooperative work, multiple AGVs are able to quickly and efficiently complete tasks, such as material handling, warehousing, and transportation. The scheduling system of the AGVs should pursue the global optimal solution, and determine a reasonable allocation plan to achieve the best match between the tasks and the AGVs.

As a key part of future automated logistics, AGV systems have broad development prospects, and the research of scheduling technology, which is the core of the AGV automatic transportation system, should be highlighted. As a concurrent system, simultaneous operation of multiple AGVs inevitably causes conflicts and deadlocks, which severely negatively affects the efficiency and reliability of the system operation. The traditional control system generally adopts a centralized scheduling strategy, in which a path planning algorithm and the task scheduling process of multiple AGVs are carried out by a central controller. Thus, the increasing number of AGVs will

make the system structure and path planning algorithm exceedingly intricate. In terms of AGV system scheduling, Fazlollahtabar *et al*. provided an overview of AGV scheduling and path planning [1, 2] and compared relevant methods. Common algorithms include the Dijkstra algorithm [3, 4], A*algorithm [5-10], ant colony algorithm [11-13], and fuzzy and path planning algorithm [14-17]. Liu *et al*. proposed a scheduling algorithm [18] that considered the process route based on a genetic algorithm. Li *et al*. considered the priority of various tasks to minimize the total path length [19, 20]. They studied the application of queuing theory in AGV scheduling with the storage and transportation system as the research object. Zheng *et al*. studied AGV scheduling technology based on an endocrine hormone regulation mechanism [21, 22]. Most scheduling strategies, however, are based on a *push strategy* and cannot accommodate the needs of production models.

From the perspective of fulfilling the actual application requirements of production systems, this study focused on AGV scheduling system modeling and AGV path planning to research a multi-objective compound task scheduling model and AGV path search algorithm. We adapted the task handling requirements in complex production scenarios, thereby raising the overall operating efficiency of the system.

2. Proposed multi-objective compound task scheduling model

2.1 Multi-objective scheduling model

The material handling system of workshop is shown in the Fig. 1. *A, B*, and *C* are the three loading and unloading manipulators in the production workshop; *a, b,* and *c* are shelving areas in the warehouse, where *a* is the shelf area of the incoming materials, and *b* and *c* are the shelf areas of the outgoing materials. In the storage task, the AGV is required to load the pallet from the designated manipulator and carry it to the designated material storage area. In the outgoing task, the AGV is required to load the pallet from the designated outbound material area and carry it to the designated manipulator buffer area. The *A, B,* and *C* manipulators handle the task of loading and unloading materials, but only the *A* manipulator handles the task of loading and unloading.

Multi-AGV scheduling systems are established on the basis of indexes, including waiting time, queue length, system throughput, running distance, and vehicle utilization percentage. Considering that the efficiency of the handling task is reflected primarily in vehicle running distance, task waiting time, and task priority in actual production, we proposed a multi-AGV task scheduling model based on total driving distance, waiting time, and the priorities of handling tasks. The objective function is shown as follows:

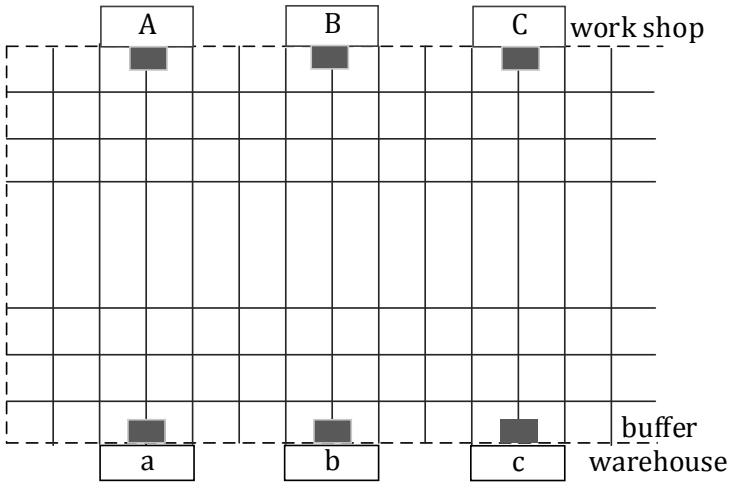

Fig. 1 The layout of the material handling system of workshop

$$\begin{cases} F = min\sum_{i=1}^{m}\sum_{j=1}^{k}(u_1 t_i + u_2 p_i + u_3)x_{ij}d_{xj} \\ s.t.\sum_{i=1}^{m} x_{ij} = 1,\ j = 1,2,\cdots,k \\ \sum_{j=1}^{k} x_{ij} = 1, i = 1,2,\cdots,m \\ x_{ij} = \{0,1\} \\ i = \{1,2,\cdots,m\}, j = \{1,2,\cdots,k\} \end{cases} \quad (1)$$

Where m is the current number of tasks; k is the number of currently available AGV vehicles; $x_{ij} = 1$ indicates that the i-th task is executed by the j-th AGV, or $x_{ij} = 0$; d_{ij} indicates the path length of the i-th task being executed by the j-th AGV; t_j is the waiting time of the i-th task at the current time; p_j is the priority of the i-th task at the current time, and u_1, u_2, u_3 is the weight coefficient.

Without a clear allocation theory for the weight coefficient, it is difficult for the current composite scheduling model to quantitatively analyze whether the weighting coefficient distribution method is reasonable. Therefore, we determined the weight coefficient according to the empirical method and dynamically adjusted the weight ratio according to the system performance.

2.2 Description of path feature

The three usual types of conflicts in the AGV path planning [23] are position conflict, opposite conflict, and same direction conflict. A position conflict occurs when multiple AGVs arrive at the same location at the same time. An opposite conflicts occurs when two AGVs meet in a path that is available for only one AGV without giving way to the other. A same direction conflict occurs when multiple AGVs run in a direction on the same path, and the latter AGV operating at a higher speed collides with the AGV ahead. Therefore, the setting of path should be restricted to avoid these three conflicts. To improve the adaptability and reliability of the AGV system, the AGV path characteristics are described as follows:

(1) Each AGV can receive only one task once, which can be changed before the AGV enters the pickup section. After it, the task cannot be changed, and the current task must be completed before accepting the next.
(2) The driving speed of each AGV is set the same when the linear and turning speeds alternate according to the actual running conditions.
(3) The deceleration and acceleration of each AGV are set the same at approaching or leaving the target point.
(4) Each section is a one-way path, with the fixed driving direction.
(5) Multiple AGVs are allowed to run simultaneously on each section.
(6) A safe distance is set between vehicles to avoid rear-end collision.
(7) A sufficient driving distance is set between any two parallel road sections to avoid road scraping.
(8) Each path intersection zone is a bicycle-only area, which allows one AGV at one time.

2.3 Path planning algorithm

To solve the shortest path when the map environment is known, scholars globally have studied many algorithms, including the genetic algorithm, neural network algorithm, Dijkstra's algorithm, and A* algorithm. The genetic algorithm randomly selects the initial population by coding through a simulation of the genetic and variability of chromosomes to calculate the adaptive function and the selection rate. Thus, the algorithm can simulate the crossover and mutation behavior to determine whether the next generation satisfies the demand. The solution to the

optimal path requires multiple iterations and requires computational intricacy. The Dijkstra's algorithm is characterized by adopting a step-by-step extended search strategy similar to the equipotential line and searches only partially connected nodes. As a heuristic search algorithm, A* algorithm adds some qualification, which can effectively discard some path points and strengthen the search efficiency. The A* algorithm combines the ideas of best-first search and Dijkstra's algorithm, and adopts heuristic search to ensure the optimal path. The A* algorithm uses an evaluation function to determine the search direction and expands from the starting point to the surrounding area. A surrogate value of each surrounding node is calculated by the evaluation function. The minimum cost node is selected as the next expansion node, and the process is repeated. In the process of searching, every node on the path has the latest cost.

To comprehensively consider the path search efficiency, we selected A* to search for the shortest path. The A* algorithm can be described by Eq. 2:

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

Where, $f(n)$ is an estimated distance from the starting point to the target point though the path point n; $g(n)$ is the actual distance from the starting point to the path point n; $h(n)$ is the distance estimation function from the path point n to the target point. The Euclidean distance from the path point n to the target point is selected as an estimation function, that is, $h(n) = \sqrt{(x_d - x_n) + (y_d - y_n)}$, where x_d, y_d are the coordinates of the target point and x_d, y_d are the coordinates of the path point n. The algorithm process is shown in Fig. 2.

As shown in Fig. 2, we set the AGV starting position point as the starting point k, searched for the path point connected to the starting point k, and calculated the distance between k and all the path points. We selected the path point with the smallest distance as the new search starting point and continued this process until the target point was found.

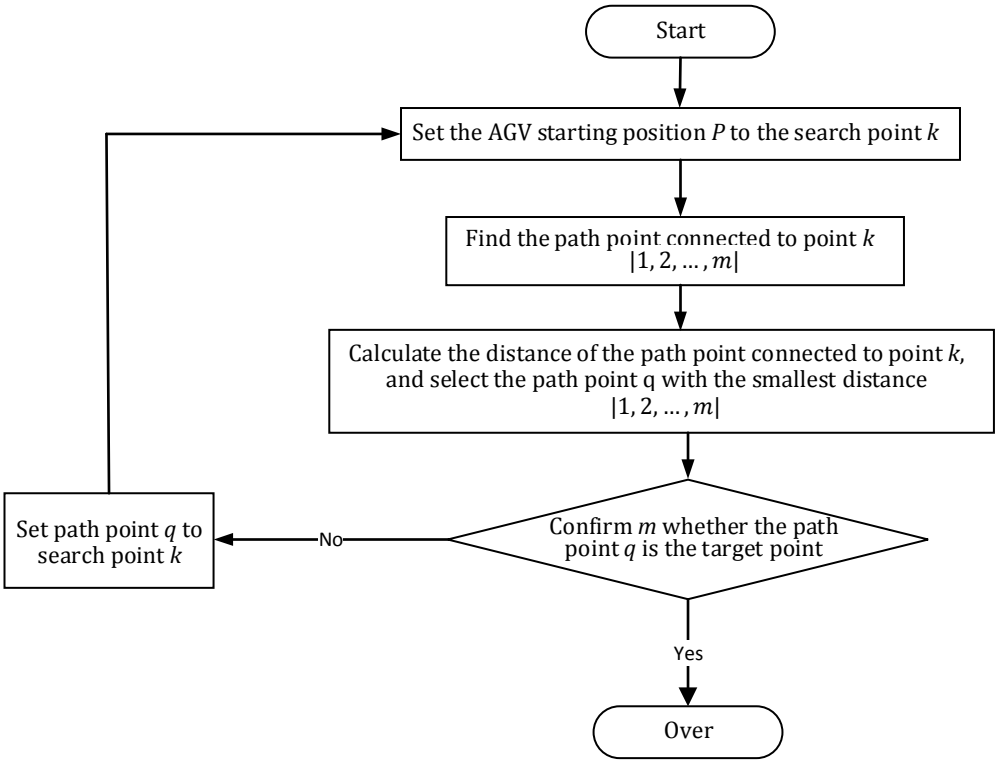

Fig. 2 The flow chart of the A* algorithm

2.4 Path conflict detection strategy

According to the materials handling system as shown in the Fig. 1, we used the graph theory to realize the mathematical representation of the map. The graph is denoted as $G = (N, W)$, where N is a set of nodes, W is a set of edges. $W_{k(i,j)} = (n_i, n_j)$ represents the edge of path k from node i to node j, where $i \neq j$ and $i, j \neq 1,2,\cdots,n$, and n is the number of nodes. The notation used in this section can be summarized as shown in Table 1.

Table 1 Key notations

Symbols	Description
L	The length of AGV
L_s	The safe length of two AGVs on the same path
P	The set of all paths
k	The index of paths, $k = 1, 2, 3, \ldots$
P_k	Path number k
K	The set of index of paths, $k \in K$
AGV_{km}	AGV number m of P_k
D_k	The distance of path k, $k \in K$
N_k	The set of points of path k, $k \in K$
n_{ki}	The point number i of path k, $k \in K$
$C_{(k_a,k_b)}$	The set of conflict points of path k_a, k_b , $k_a \in K$, $k_b \in K$
C_{kr}	The conflict point number i of path k, $k \in K$
s	The start position of path k, $k \in K$
e	The end position of path k, $k \in K$
t_{kms}	The start time of AGV m of path k, $k \in K$
t_{kmc_r}	The spending time that AGV m arrival to the conflicts node r of path k, $k \in K$
t_{kme}	The end time of AGV m of path k, $k \in K$
tc_k	The set of the time that AGVs arrival to the conflict nodes of path k, $k \in K$
TC	The set of the time that AGVs arrival to the conflict nodes
T	The total time of all conflict time

We identified an AGV path conflict if the two AGVs arrived at the same node at the same time. One should run away from the other AGV to avoid the path conflicts. Therefore, we gave the minimum safety length L_s of the two AGVs on the same path.

To detect AGV path conflicts, we considered both the path overlap and the time conflicts that pass the node. Between the conflict nodes, there were non-conflict paths in which the AGV normally could pass. The speed of AGV was v_0. Eq. 3 calculates the conflict detection of the path node. Eq. 4 calculates the conflict detection of the pass time and L/v_0 represents the total time from arrival to pass the conflict node. To guarantee the drive safety of the two AGVs, the absolute value of the time difference between the two AGVs arrival to the conflict path should be no less than $(L + L_s)/v_0$, or else we call it AGV conflicts happen.

$$C(k_1, k_{11}) = N_K \cap N_K \tag{3}$$

$$\mathrm{TC} = |\mathrm{tc_k} - \mathrm{tc_k}| < (\mathrm{L} + \mathrm{L_s})/\mathrm{v_0} \tag{4}$$

When we detected the AGV path conflict, the priority of the AGVs should be sequenced. The lower priority AGV should wait until the higher priority AGV passed the conflict path for the multiple AGVs approaching the conflict path. The longer the time of arrival at the conflict section, the later the arrival would be; therefore, priority was lower, and the AGV that arrived first would move forward.

The solution for path conflicts is the speed control strategy. For the speed control strategy, we kept the higher priority AGV drive speed uniform, but changed the lower priority AGV drive speed. We gave acceleration a for the speed change. In Eq. 5, l_s represents the path length of AGV decelerating from v_0 to v_1 or accelerating from v_1 to v_0. Eq. 6 represents the AGV decelerates to v_1, and then AGV passes through the conflict node at constant speed v_1. Eq. 8 and Eq. 9 represent the spending time before the AGV arrival to the conflicts node the speed change. Eq. 10 and Eq. 11 represent the spending time that AGV arrival to the conflicts node 1, 2, and r accordingly. Eq. 12 represents the total time for the AGVs arrival to the terminal.

$$l_s = (v_0^2 - v_1^2)/2a \tag{5}$$

$$(L + L_s)/v_0 = (v_0 - v_1)/a + (L + L_s)/v_1 \tag{6}$$

$$t_1 = (L + L_s)/v_0 \tag{7}$$

$$t_2 = (v_0 - v_1)/a \tag{8}$$

$$t_{k1mc1} = t_{kms} + t_1 + \left(W_{k(s,uk1)} - L_S\right)/v_0 \tag{9}$$

$$t_{k1mc2} = t_{k1m1} + \left(W_{k1(s,uk1)} - L_S\right)/v_0 + t_2 \tag{10}$$

$$t_{k1mc2} = t_{k1m1} + \sum_{i=2}^{ukr}\sum_{j=2}^{ukr} W_{k1(i,j)}\Big/v_0 = t_{k1mc2} + \sum_{i=2}^{ukr}\sum_{j=2}^{ukr}\left(n_{k1i}, n_{k1j}\right)\Big/v_0 \tag{11}$$

$$min\ T_{k1m} = t_{k1m1} - (L_s - l_S)/v_0 + t_1 + t_2 \tag{12}$$

2.5 Efficient MapReduce framework

MapReduce [24, 25] is a parallel computing framework for cluster systems. Phoenix++ [26] is a C++ programming implementation of the MapReduce framework based on shared memory, which can be used to program multi-core chips and shared memory multiprocessors. MapReduce framework optimization provides the opportunity to reuse hash table storage space between multiple Map tasks and reuse the hash table between multiple Reduce tasks. Accordingly, we designed a high-efficiency reusable hash table (HRHT). Considering that the unordered map used by Phoenix in the Reduce stage used a linked list to store data, we had to apply space for each pair of *keys/values* to create a node, which caused a large number of memory applications. As a result, the HRHT designed in this study eliminated the use of linked lists. Fig. 3 displays the data structure and data access process of HRHT, which improved the *unordered_map* as follows:

Step 1: A valid flag added to each bucket indicated that the data in the corresponding bucket were invalid, and the bucket could be used to store new data. The data in the corresponding bucket were valid, that is, the current bucket was already occupied. When the hash table needed to be reused, it had to reset all valid flags to 0, and thus invalidated all the data in the bucket. The HRHT could be quickly reset by the valid flag to ensure high efficiency reuse of the hash table.

Step 2: The linked list was no longer used, and the *key/value* pairs were saved directly to buckets. When there were data to be added, they were mapped to a position in the buckets according to the key. We then checked the buckets backward from the position, found a location with a valid flag of 0 from the buckets, and then saved the new data. Thus, we no longer had to apply for space temporarily.

In this study, we introduced HRHT into the design of the hash container and designed it as an HRHT-based hash container. Then we replaced the unordered map of Reduce stage with HRHT to avoid the application of a large number of small blocks of memory. In addition, HRHT also guaranteed that the bucket size was a power of 2 when being designed. After the hash value was calculated by $Hash()$, when positioned in the bucket, we used (*hash_value&*(*buckets_size-1*)) to replace the previous modulo calculation (*hash_value%buckets_size*) and thus improved retrieval efficiency.

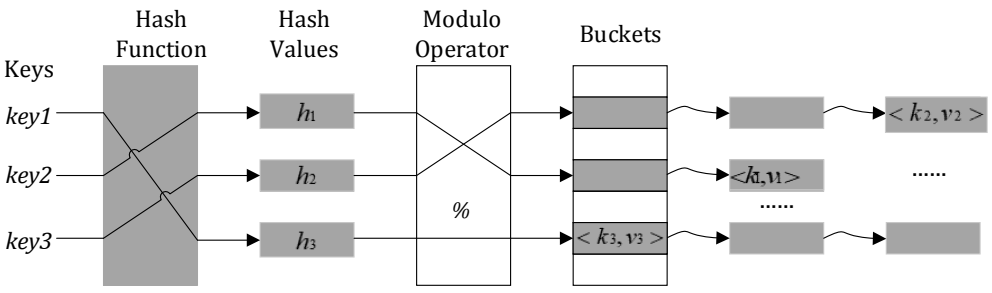

Fig. 3 The data structure of HRHT

3. Experimental analysis: Results and discussion

In this study, we took the logistics of an electrical connector workshop as the research background. *A, B*, and *C* were the three loading and unloading manipulators in the production workshop and *a, b*, and *c* where the shelving areas in the warehouse. Symbol *a* represents is the shelf area of incoming materials, *b* and *c* are the shelf areas of outgoing materials. In the storage task, the AGV was required to load the pallet from the designated manipulator and had to carry it to the designated material storage area. In the outgoing task, the AGV was required to load the pallet from the designated outbound material area and carry it to the designated manipulator buffer area. The *A, B*, and *C* manipulators handled the task of loading and unloading materials, but only the *A* manipulator handled the task of loading and unloading.

In this analysis, the experimental system is based on Intel Xeon E5645 platform (6 cores per CPU, 12 CPUs in total), which can be used to run MapReduce framework Phoenix++, and is used CentOS 6.5 as the operating system and GCC 4.7 as the compiler. We set work for three AGVs, and then set basic parameters: $L = 1.2$ m, $L_s = 1$ m, $v_0 = 1.2$ m/s, $v_1 = 0.8$ m/s, and $a = 0.4$ m/s^2. We used the A* algorithm to plan the AGV operation path. The path of the AGV is shown in Figure 4, and the overlap of each path is given in Table 2.

We compared the impact of speed control strategy combined with efficient MapReduce framework on the AGV total travel time. Compared with the normal arrival time, the 0 time difference indicated that the AGV driving state in the path remained unchanged; a positive time difference indicated that the AGV waited in the path. As shown in Fig. 5 and Fig. 6, the efficient MapReduce framework improved the execution efficiency of the speed control strategy and shortened the spending time of the entire loading and unloading process.

With the increase of the number of tasks and AGVs, the possibility of conflict path segments between different paths increased, which led to an increase in the probability of conflict between AGVs. We predicted the probability of conflict between AGVs, as shown in Fig. 7. It can be seen that the efficient MapReduce framework improved the execution efficiency of the algorithm and reduced the probability of conflict between AGVs.

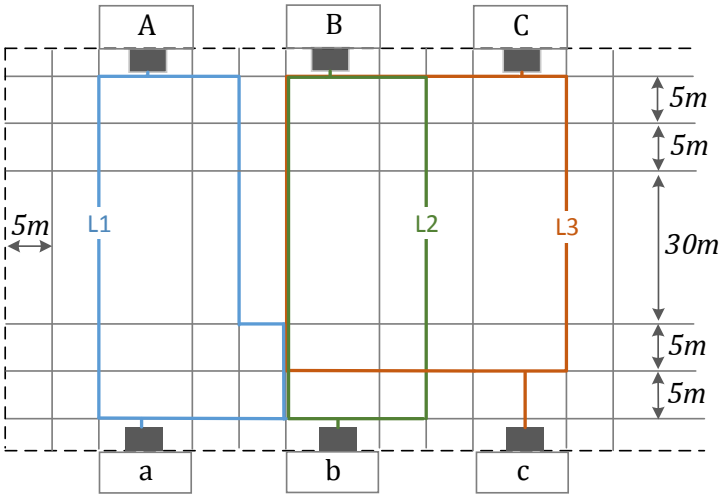

Fig. 4 The map of AGVs

Table 2 Overlapping paths

No.	Path number	Path length	Number of overlapping paths
1	L1	140 m	L2, L3
2	L2	130 m	L1, L3
3	L3	155 m	L1, L2

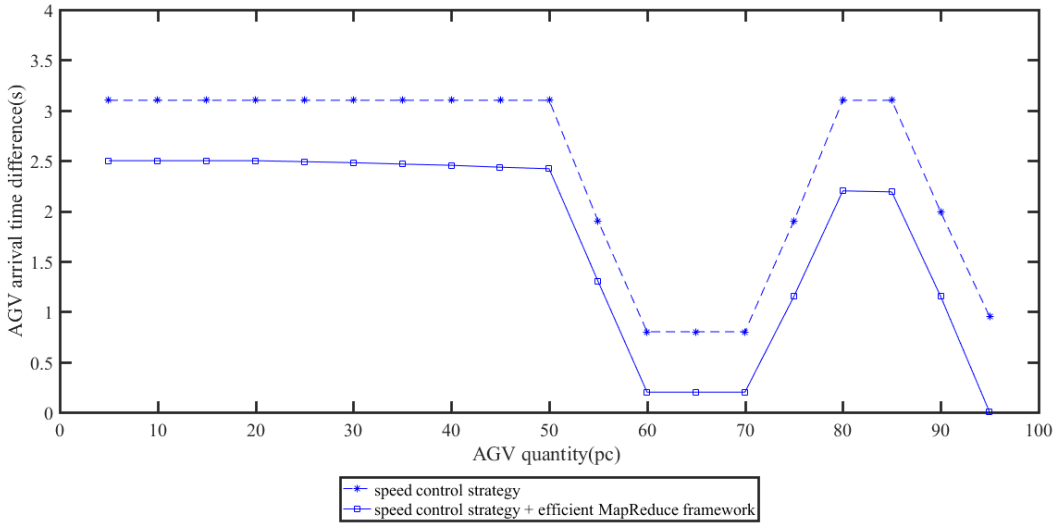

Fig. 5 Changes in the time of the AGV arrival time

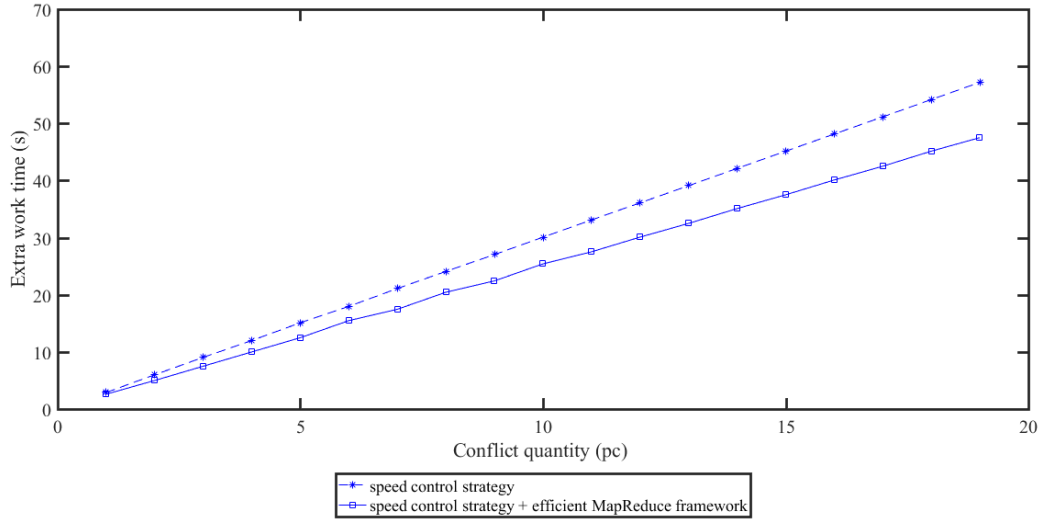

Fig. 6 The impact of speed control strategy combined with efficient MapReduce framework

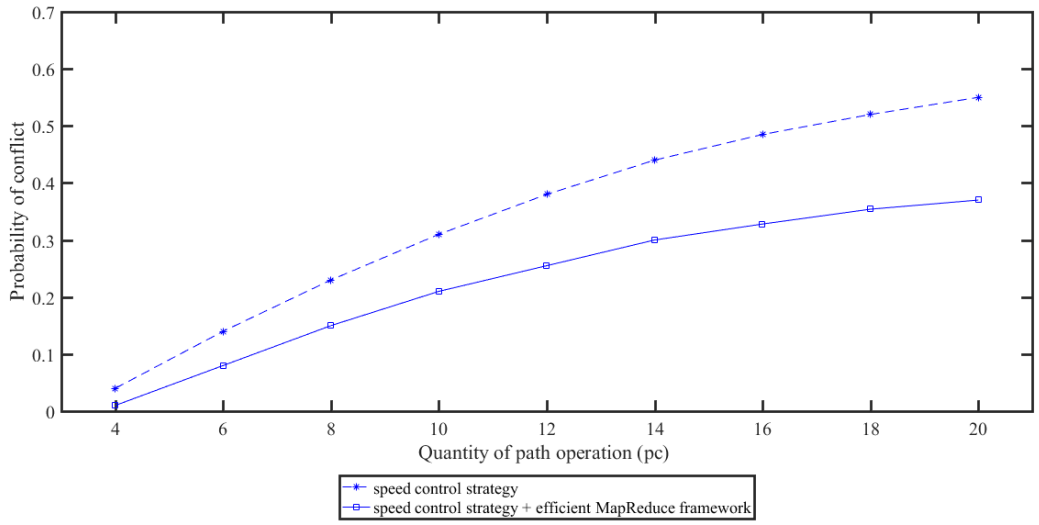

Fig. 7 The prediction of conflict between AGVs

4. Conclusion

In this paper, a multi-objective scheduling model is proposed based on total driving distance and waiting time, the A* algorithm is selected to search the shortest path of AGVs , the speed control strategy is proposed to detect path conflict without changing the driving path and task scheduling, the MapReduce framework is adopted to improve the speed control strategy execution efficiency. The experiment is proven to the scheduling model and the speed control strategy combined with the MapReduce framework is feasible and effective to reduce the probability of conflict in material handling process of AGVs.

In the future work, more researches should be carried on dynamic scheduling model of multi-AGV to solve the path conflict occurred in large-scale material handling process of AGVs.

Acknowledgement

Project supported by Industrial Internet Innovation and Development Project 2018 (INDICS Industrial Internet Platform Project), the National Key R&D Program of China (No. 2018YFB1004000).

References

[1] Fazlollahtabar, H., Saidi-Mehrabad, M. (2015). Methodologies to optimize automated guided vehicle scheduling and routing problems: A review study, *Journal of Intelligent & Robotic Systems*, Vol. 77, No. 3-4, 525-545, doi: 10.1007/s10846-013-0003-8.

[2] Sun, Q., Liu, H., Yang, Q., Yan, W. (2011). On the design for AGVs: Modeling, path planning and localization, In: *Proceedings of the 2011 IEEE International Conference on Mechatronics and Automation,* Beijing, China, 1515-1520, doi: 10.1109/ICMA.2011.5985974.

[3] Yershov, S., LaValle, S.M. (2012). Simplicial Dijkstra and A∗ algorithms: From graphs to continuous spaces, *Advanced Robotics*, Vol. 26, No. 17, 2065-2085, doi: 10.1080/01691864.2012.729559.

[4] Wang, C.L., Li, S.W. (2018). Hybrid fruit fly optimization algorithm for solving multi-compartment vehicle routing problem in intelligent logistics, *Advances in Production Engineering & Management*, Vol. 13, No. 4, 466-478, doi: 10.14743/apem2018.4.304.

[5] Wang, C., Wang, L., Qin, J., Wu, Z., Duan, L., Li, Z., Cao, M., Ou, X., Su, X., Li, W., Lu, Z., Li, M., Wang, Y., Long, J., Huang, M., Li, Y., Wang, Q. (2015). Path planning of automated guided vehicles based on improved A-star algorithm, In: *Proceeding of the 2015 IEEE International Conference on Information and Automation,* Lijiang, China, 2071-2076, doi: 10.1109/ICInfA.2015.7279630.

[6] Lau, H.Y.K., Zhao, Y. (2008). Integrated scheduling of handling equipment at automated container terminals, *Annals of Operations Research,* Vol. 159, 373-394, doi: 10.1007/s10479-007-0262-5.

[7] Liu, Y., Ji, S., Su, Z., Guo, D. (2019). Multi-objective AGV scheduling in an automatic sorting system of an unmanned (intelligent) warehouse by using two adaptive genetic algorithms and a multi-adaptive genetic algorithm, *PLoS ONE*, Vol. 14, No. 12, Article ID e0226161, doi: 10.1371/journal.pone.0226161.

[8] Li, G., Zeng, B., Liao, W., Li, X., Gao, L.(2018). A new AGV scheduling algorithm based on harmony search for material transfer in a real-world manufacturing system, *Advances in Mechanical Engineering*, Vol. 10, No, 3, Article No. 1687814018765560, doi: 10.1177/1687814018765560.

[9] Yang, M.S., Ba, L., Zheng, H.Y., Liu, Y., Wang, X.F., He, J.Z., Li, Y. (2019). An integrated system for scheduling of processing and assembly operations with fuzzy operation time and fuzzy delivery time, *Advances in Production Engineering & Management*, Vol. 14, No. 3, 367-378, doi: 10.14743/apem2019.3.334.

[10] Zhao, X.F., Liu, H.Z., Lin, S.X., Chen, Y.K. (2020). Design and implementation of a multiple AGV scheduling algorithm for a job-shop, *International Journal of Simulation Modelling*, Vol. 19, No. 1, 134-145, doi: 10.2507/IJSIMM19-1-CO2.

[11] Wang, J.F., Liu, J.H., Zhong, Y.F. (2005). A novel ant colony algorithm for assembly sequence planning, *The International Journal of Advanced Manufacturing Technology*, Vol. 25, 1137-1143, doi: 10.1007/s00170-003-1952-z.

[12] Reed, M., Yiannakou, A., Evering, R. (2014). An ant colony algorithm for the multi-compartment vehicle routing problem, *Applied Soft Computing,* Vol. 15, 169-176, doi: 10.1016/j.asoc.2013.10.017.

[13] Balseiro, S.R., Loiseau, I., Ramonet, J. (2011). An ant colony algorithm hybridized with insertion heuristics for the time dependent vehicle routing problem with time windows, *Computers & Operations Research*, Vol. 38, No. 6, 954-966, doi: 10.1016/j.cor.2010.10.011.

[14] Mousavi, M., Yap, H.J., Musa, S.N., Dawal, S.Z.M. (2017). A fuzzy hybrid GA-PSO algorithm for multi-objective AGV scheduling in FMS, *International Journal of Simulation Modelling*, Vol. 16, No. 1, 58-71, doi: 10.2507/IJSIMM16(1)5.368.

[15] Sadeghpour, H., Tavakoli, A., Kazemi, M., Pooya, A. (2019). A novel approximate dynamic programming approach for constrained equipment replacement problems: A case study, *Advances in Production Engineering & Management*, Vol. 14, No. 3, 355-366, doi: 10.14743/apem2019.3.333.

[16] Xu, W., Yin, Y. (2018). Functional objectives decision-making of discrete manufacturing system based on integrated ant colony optimization and particle swarm optimization approach, *Advances in Production Engineering & Management*, Vol. 13, No. 4, 389-404, doi: 10.14743/apem2018.4.298.
[17] Li, H.-Y., Xu, W., Cui, Y., Wang, Z., Xiao, M., Sun, Z.-X. (2020). Preventive maintenance decision model of urban transportation system equipment based on multi-control units, *IEEE Access*, Vol. 8, No. 1, 15851-15869, doi: 10.1109/ACCESS.2019.2961433.
[18] Liu, S.N., Ke, Y.L. (2007). An algorithm for job shop scheduling in dual resource constrained with AGV, *China Mechanical Engineering*, Vol. 15, 1810-1813.
[19] Li, X. (2012). Utilization control optimization of AGV based on queuing theory, *Journal of Lanzhou Jiaotong University*, Vol. 31, No. 6, 91-93.
[20] Jiang, C., Xi, J.T. (2019). Dynamic scheduling in the engineer-to-order (ETO) assembly process by the combined immune algorithm and simulated annealing method, *Advances in Production Engineering & Management*, Vol. 14, No. 3, 271-283, doi: 10.14743/apem2019.3.327.
[21] Zheng, K., Tang, D., Giret, A., Gu, W., Wu, X. (2015). Dynamic shop floor re-scheduling approach inspired by a neuroendocrine regulation mechanism, *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, Vol. 229, No, 1, 121-134, doi: 10.1177/0954405414558699.
[22] Zheng, K., Tang, D., Gu, W., Dai, M. (2013). Distributed control of multi-AGV system based on regional control model, *Production Engineering, Research and Development*, Vol. 7, No. 4, 433-441, doi: 10.1007/s11740-013-0456-4.
[23] Rong, J. (2017). Research on collision avoidance strategy of AGV, In: *Proceedings of 2017 International Conference on Manufacturing Construction and Energy Engineering (MCEE 2017),* Hong Kong, 15754-25995, doi: 10.12783/dtetr/mcee2017/15754.
[24] Dean, J., Ghemawat, S. (2008). MapReduce: Simplified data processing on large clusters, *Communications of the ACM*, Vol. 51, No. 1, 107-113, doi: 10.1145/1327452.1327492.
[25] Eken, S., Sayar, A. (2019). A MapReduce-based big spatial data framework for solving the problem of covering a polygon with orthogonal rectangles, *Tehnički Vjesnik – Technical Gazette,* Vol. 26, No. 1, 36-42, doi: 10.17559/TV-20170418094421.
[26] Talbot, J., Yoo, R.M., Kozyrakis, C. (2011). Phoenix++: Modular mapreduce for shared-memory systems, In: *Proceedings of the Second International Workshop on MapReduce and Its Applications,* New York, USA, 9-16, doi: 10.1145/1996092.1996095.