Proceedings of the 2022 Winter Simulation Conference
B. Feng, G. Pedrielli, Y. Peng, S. Shashaani, E. Song, C.G. Corlu, L.H. Lee, E.P. Chew, T. Roeder, and P. Lendermann, eds.

DEADLOCK AVOIDANCE DYNAMIC ROUTING ALGORITHM FOR A MASSIVE BIDIRECTIONAL AUTOMATED GUIDED VEHICLE SYSTEM

Kang Min Kim
Chang Hyun Chung
Young Jae Jang

Department of Industrial and Systems Engineering
Korea Advanced Institute of Science and Technology
291, Daehak-ro, Yuseong-gu,
Daejeon, 34141, REPUBLIC OF KOREA

ABSTRACT

In a bidirectional automated guided vehicle (AGV) system, it is essential to allocate routes to AGVs to avoid deadlocks. However, avoiding deadlocks is more challenging in dynamic environments where AGVs are continuously assigned tasks and thus operate simultaneously. This paper proposes a deadlock prevention method for dynamic environments. The proposed method combines the active path reservation method, which restricts the directions of the paths of some moving AGVs, and the dynamic Dijkstra algorithm, which finds the shortest route according to the path reservation status. Apart from the Dijkstra algorithm, the proposed method is compatible with other routing algorithms, such as the Q-learning-based route algorithm; therefore, the proposed method enables the development of more efficient transport systems that account for AGV congestion. The efficiency and scalability of the proposed method were verified using Applied Materials AutoMod (version 14.0) simulation software.

1 INTRODUCTION

Automated guided vehicles (AGVs) have been widely used in plants and warehouses to increase throughput and reduce operating costs (Zhao et al. 2020). In addition, warehouses have changed their modes of fulfillment to meet the increased demand arising from E-commerce. A large number of products are stored in limited warehouse spaces; therefore, the spaces available for AGV transport have become smaller and narrower. In warehouses, the bidirectional AGV system, in which the path used by the AGV is not fixed, can be more efficient than the unidirectional AGV system; however, preventing deadlock in the bidirectional system is more complicated.

As shown in Figure1, two types of deadlocks may occur in the bidirectional AGV system: head-on deadlock and loop cycle. A head-on deadlock occurs when two AGVs facing each other wait for each other because their paths are interlocked. A loop cycle occurs when the routes of four or more AGVs form a cyclical loop.

AGV routing methods for preventing deadlock and conflict in the bidirectional AGV system have been studied for decades. The prevention methods can be classified as static routing and dynamic routing, depending on whether the AGV can change its route while moving. The static routing method, in which the route determined at the departure time does not change until the vehicle reaches its destination, involves either of the following processes: (1) Before the AGV departs, a deadlock-free route is determined using time-window based methods (Kim and Tanchoco 1991; Jia et al. 2017; Zhang et al. 2018; Triwiyatno and Riyadi 2020; Maza and Castagna 2001) or Petri net (Wu and Zhou 2001; Wu and Zhou 2004; Wu and Zhou 2007; Fanti 2002); (2) the shortest path from source to destination is first determined, and then

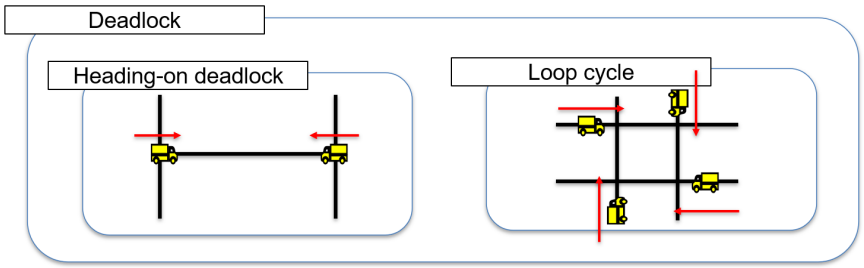

Figure 1: Two types of deadlock.

deadlock is prevented using another algorithm such as the dynamic resource reservation (DRR) method, established by Zhao et al. (2020), or the chain of reservation (COR) method, established by Małopolski (2018).

Time-window-based methods calculate and adjust the time for AGVs to reach their destination following a deadlock-free route, while Petri net models check whether the transition on the Petri net will cause a deadlock. However, these methods are not adequate for extensive systems with many AGVs because as the number of moving AGVs increases, the computational cost to find a deadlock-free route exponentially increases.

The DRR and COR methods are applicable to large systems because the number of vehicles and the computational cost of the algorithm is not strongly related. In the DRR method, if the routes of two vehicles considerably overlap, the AGV with a later start time waits until the other AGV leaves the overlapping route. In the COR method, once the order of AGVs passing through a node is set, it cannot be changed. Therefore, lower-priority vehicles must wait even if they can pass first. While the work in (Sharon et al. 2015) deals with multi-agent path finding (MAPF). It presents a conflict-based formalization for MAPF and a corresponding an algorithm called Conflict Based Search (CBS).

In contrast to static methods, dynamic routing methods (Sun et al. 2018; Yan et al. 2017; Breton et al. 2006; Zhao et al. 2021; Jager and Nebel 2001; Fransen et al. 2020) follow the procedure of deadlock resolution after deadlock detection, without finding a deadlock-free route in advance. They consider the possibility of a new deadlock occurrence when a moving AGV deviates from its predetermined route. These methods are difficult to apply in large fulfillment centers because deadlock resolution must be performed after deadlock detection, which can result in inefficient AGV transport in narrow spaces.

Research in Fragapane et al. (2021) describes the material handling with autonomous mobile robots (AMR). However, the work describes in this paper deals with AGV moving on a grid network and therefore, the autonomous path guidance used in AMR is not applicable.

In this paper, we focus on the routing problem for a multi-AGV bidirectional environment. We propose a high-performance dynamic routing algorithm that efficiently avoids the deadlock of dozens to hundreds of continuously working AGVs in a bidirectional environment, making it suitable for real-world fulfillment centers. The proposed algorithm limits the directions of some paths using the active path reservation method, and seeks the shortest route using the dynamic Dijkstra algorithm, in which the path weight is changed in real-time to avoid deadlock. The proposed algorithm can accommodate various types of routing algorithms and the Dijkstra algorithm. Finally, we discuss the verification of the algorithm using the simulation software program Applied Materials AutoMod™ (version 14.0).

The contributions of this paper are as follows: First, we propose an algorithm to avoid deadlocks and loop cycles among moving AGVs. Specifically, the deadlocks and loop cycles avoidance approach proposed in this paper is decoupled from the path routing algorithm. As a result, any type of path routing algorithm can be applied on top of the proposed deadlocks and loop cycles avoidance method. This decoupling provides considerable efficiency in designing AGV based material handling system. Second, we propose

an algorithm for operating large-scale AGVs in a bidirectional AGV environment in which AGVs are continuously assigned tasks.

2 DEADLOCK AVOIDANCE ROUTING ALGORITHM

2.1 Working Environment Description

Figure 2 shows the layout of the fulfillment warehouse targeted in this study. The mobile storage unit (MSU) is a shelf on which various items are stored. According to their roles, the warehouse environment can be divided into a storage area, aisle area, and station. The MSU is in the storage area. The AGV transports the MSU from the storage area to the station and back to its original location. After unloading the MSU, the AGV waits in the storage area for the next task. The aisle area is the path through which the AGV transports the MSU to the station. In the aisle areas, AGVs can move in both directions. The aisle area near the storage area forms a single bidirectional lane path within the limited warehouse space. At the station, the operator takes the goods from the MSU delivered by the AGV and performs the necessary tasks, such as packaging.

Moreover, according to the shape, the warehouse environment can be divided into the grid section and the module section. The grid section is the area between the station and the module section. The module section comprises several uniformly connected unit modules. In the module section, congestion may occur among AGVs entering a unit module to retrieve the MSU and those exiting a unit module to deliver the MSU to the station.

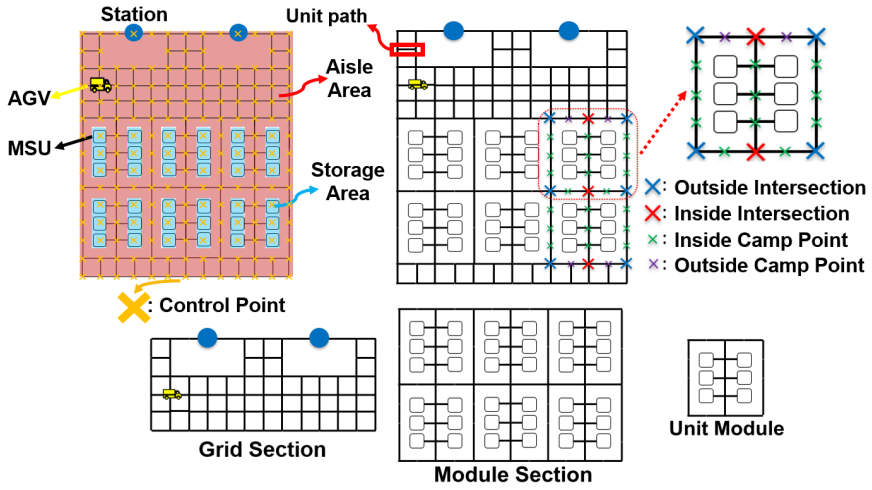

Figure 2: Components of a fulfillment warehouse environment.

The unit module includes two intersections and two types of camp points: outside intersection (OI), inside intersection (II), inside camp point (ICP), and outside camp point (OCP). The OI is a point where two unit modules meet. The II is the area in the unit module through which the AGV must pass when entering or exiting the storage area. The OCP is the point where the grid section and the module section come in contact, while the other points in the module section are the ICPs. The number of MSUs in the unit module does not always have to be six (Figure 2).

2.2 Definitions

Definition 1 A control point (CP) refers to a node responding to the intersection of all unit paths, including paths in the grid and module sections.

While on a CP, AGVs make decisions to find their next CP. There are 189 CPs in the environment of Figure 2.

Definition 2 A Path information ($P_{a,i}$) indicates the direction in which an AGV can move from a CP.

All AGVs share the same path information (PI) value, which is expressed as

$$P_{a,i} = \begin{cases} 1, & \text{if AGV can move } i \text{ direction from CP } a \\ 0, & \text{otherwise} \end{cases},$$

where $a \in \{\text{CPs}\}$, $i \in \{\text{right, left, up, down}\}$.

As shown in Figure 3, $P_{A,right}$ and $P_{A,left}$ are equal to 1, because the AGV can move leftward and rightward from the CP A. The PI values, used in the dynamic Dijkstra algorithm to limit path direction, for the other directions are 0.

$P_{\text{A, right}} = 1$, $P_{\text{A, left}} = 1$, $P_{\text{A, up}} = 0$, $P_{\text{A, down}} = 0$

Figure 3: Path information example for CP A.

2.3 Assumptions

The proposed algorithms are based on nine assumptions.

- An AGV can only transport one MSU at a time.
- An AGV receives only one task at a time.
- An AGV does not exchange an MSU with other AGVs.
- An AGV waiting for the next task in the storage area does not block the paths of other AGVs.
- An AGV in the storage area does not block the paths of other AGVs.
- An AGV can move upward, downward, leftward, and rightward if there is a unit path.
- AGVs can only move along the unit path, with or without an MSU
- The unit module is uniformly repeated in the environment.
- The number of AGVs is smaller than the number of MSUs in the layout.

2.4 AGV Movement Procedure

Figure 4 shows a basic example of the AGV movement procedure. The "claim" procedure involves an AGV securing ownership of a CP. Before an AGV reaches a CP, the claim procedure for the next CP starts. If multiple AGVs have been waiting for a claim against a CP, the AGV that has waited the longest will proceed first with the claim process. An AGV that has ownership of a CP returns the ownership when it passes the CP. All AGVs repeat this movement process until they reach their destinations. Two AGVs cannot simultaneously claim one CP, which helps to prevent conflict.

2.5 Head-on Deadlock Avoidance

The proposed algorithm simultaneously uses the dynamic Dijkstra algorithm and the active path reservation method to avoid a head-on deadlock. This section introduces the head-on deadlock prevention process and details the application of active path reservation for limiting path direction and preventing deadlock. Finally, the application of the dynamic Dijkstra algorithm for finding the next CP is described.

To prevent a head-on deadlock, a specific direction in a path is reserved for a short period so that the other AGVs cannot use the path in the opposite direction. For instance, as shown in Figure 5, when AGV K moves rightward from CP G1, and another AGV enters from the opposite direction through CP G2, a

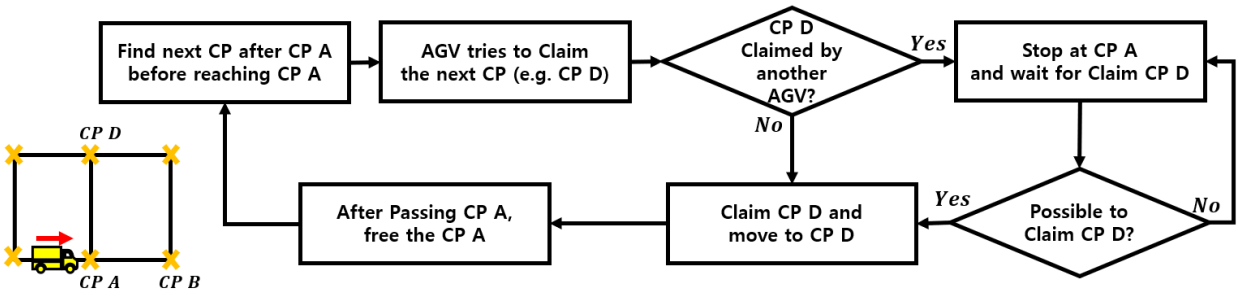

Figure 4: Movement procedure of an AGV.

head-on deadlock will occur. Therefore, all paths from CP G1 to CP G2 is reserved for AGV K; here, the next CP of AGV K is determined as CP A. Reserving the corresponding path implies changing $P_{A,left}$, $P_{B,left}$, $P_{C,left}$ and $P_{G2,left}$ to 0. The *PI* value of the reserved path is one so that the other AGVs can avoid a head-on deadlock on that route, as guided by the dynamic Dijkstra algorithm. The paths reserved by AGV K are initialized; the *PI* value is changed to 1 when AGV K passes CP G2. The reserved path is not initialized when AGV K arrives at CP G2, but at the moment, it passes CP G2. Moreover, if AGV L follows AGV K along the same direction, the time the reserved path is initialized is not when AGV K passes CP G2 but when AGV L passes CP G2. All AGVs continuously reserve and initialize the path while moving to prevent a head-on deadlock. This method is called active path reservation.

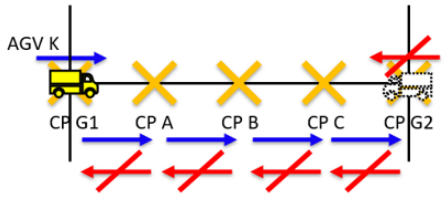

Figure 5: Main principle in head-on deadlock prevention.

The principle in the active path reservation method slightly varies with the location and situation of the AGV in the actual environment layout. AGVs have numerous direction choices within the grid section; thus, in this section, path reservation and initialization occur on a unit path by unit path basis. For example, as shown in Figure 6, the moment AGV K claims CP B, it reserves the path from CP A to CP B, and $P_{B,left}$ changes to 0, and when the AGV passes CP B, $P_{B,left}$ changes to 1.

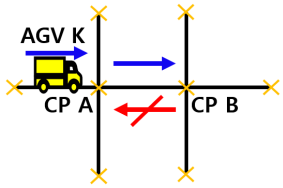

Figure 6: Path reservation on grid section.

In contrast, two or more unit paths are simultaneously reserved or reset to the initial value in the module section. Figure 7 shows the path reservation process for the four AGV scenarios in the module section. For all cases, when the AGV claims the blue CP, all unit paths from the current AGV location to the black CP are simultaneously reserved. Moreover, once the AGV passes the yellow CP, the *PI* value of the path previously passed by the AGV is reset to the initial value so that other AGVs can rapidly use the path. After the AGV arrives at the black CP, the *PI* value for the remaining paths is simultaneously reset to the

initial value. For Case 1, to reduce AGV congestion in II, the path to the black CP is reserved. In Case 2, the path the AGV does not pass is reserved to prevent two AGVs from simultaneously leaving the unit module through two IIs.

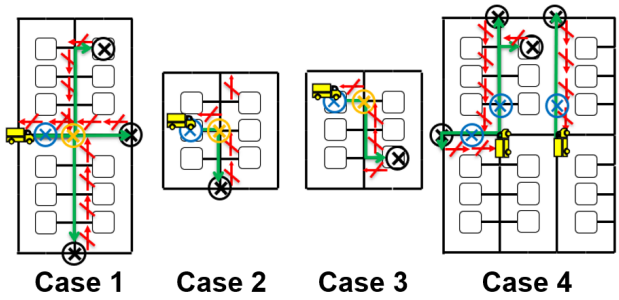

Figure 7: Path reservation on module section.

The dynamic Dijkstra algorithm finds and updates the new route from the current location to the destination, considering the *PI* values updated according to the active path reservation method. The dynamic Dijkstra algorithm may find a route to the destination but often fails. If the algorithm cannot find a route (a state referred to as "Dijkstra fail"), the next CP is found according to the AGV location and the situation in the surroundings. The algorithm's process for finding the next CP is detailed in Appendix 1.

3 LOOP CYCLE AVOIDANCE

Even if the *PI* value is dynamically modified through active path reservation, the loop cycle problem can still occur. Because the loop cycle is a situation where the routes of four or more AGVs are rotationally oriented instead of approaching face to face, it cannot be prevented through *PI* value adjustment. Consequently, loop cycles often occur in grid sections. The proposed algorithm simultaneously applies the loop cycle direction avoidance and stopped vehicle avoidance methods to prevent the loop cycle problem.

3.1 Loop Cycle Direction Avoidance

The loop cycle direction avoidance method ensures that the last AGV that can form a loop cycle avoids the direction that would result in a loop cycle. For instance, in the left panel of Figure 8, if the yellow AGV moves upward, a loop cycle would occur. The algorithm detects and limits the loop cycle direction and directs the AGV to move rightward instead, where a loop cycle cannot form.

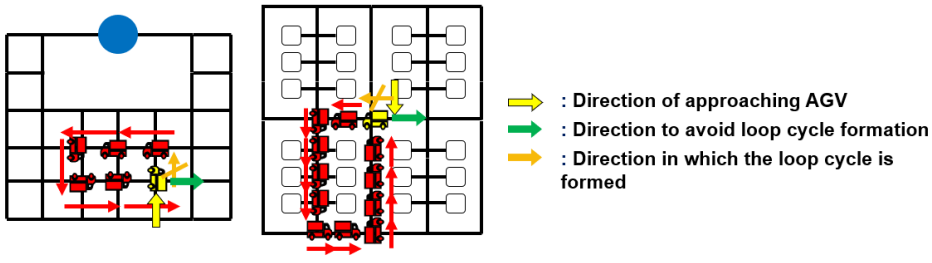

Figure 8: Examples of Loop cycle direction avoidance in grid section and module section.

3.2 Stopped Vehicle Avoidance Method

However, in an environment where a dozen or more AGVs are moving in real-time, an AGV may be unable to deviate to avoid a loop cycle, as the loop-free route may be occupied by another approaching

AGV (Figure 9). As shown in the grid section in Figure 9, the yellow AGV can only return to its origin. Moreover, in the module section, there are situations in which a deadlock cannot be avoided only by loop cycle direction avoidance. If multiple AGVs stop simultaneously, the probability of a loop cycle occurrence increases (Figure 9). Therefore, the AGV needs to avoid the other stopped AGVs. The stopped vehicle avoidance method works differently in the grid and module sections. First, in the grid section, the strong path information reservation method directs AGVs to avoid a CP that another AGV is currently waiting to claim. The method is as follows:

1. Define the *Strong Path Information value* (SPI) to be same as the *PI value*.
2. Change the *SPI* value according to the modification of the *PI* value.
3. If an AGV is waiting to claim the next CP (CP N), change the *SPI values* in all directions from the adjacent CP of the current AGV location (CP C) to 0.
4. Reset the *SPI* value to 1 when an AGV starts moving after the claim process.
5. First find the routes of all AGVs using the dynamic Dijkstra algorithm and only the direction of the path with an *SPI value* of 1.
6. If the dynamic Dijkstra algorithm fails to find a route using the *SPI value*, then repeat the process using the *PI value* instead.

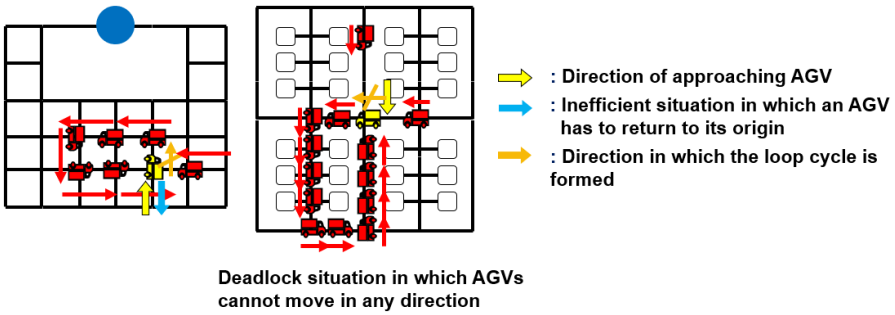

Figure 9: Examples of inefficient and inevitable scenarios in loop cycle formation.

However, because the strong path information reservation method is naïve and does not consider securing the routes of other AGVs, the dynamic Dijkstra algorithm often cannot find the route of the AGV of interest. Given that the dynamic Dijkstra algorithm using the *SPI value* can avoid the CP in which the AGV stops only by finding a route, layout tuning is required to minimize route determination failure.

The strong path information value is not modified in the module section even if an AGV stops and waits. Because the unit module section is in the form of a single road, when the strong path information value is changed to 0, Dijkstra fail always occurs. In the CP within the black circle in Figure 10, the strong path information value is not modified to avoid Dijkstra fail. To prevent congestion between the station entrance and the exits of other stations, some paths near the station are made unidirectional. Unidirectionalization means that although AGVs can move in both directions, modification of the dynamic Dijkstra algorithm cost causes the AGVs to move in the desired direction in as many instances as possible.

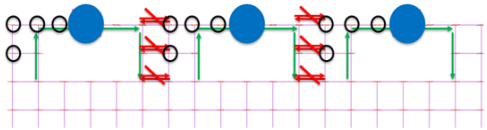

Figure 10: Layout tuning in the strong path information reservation method.

Furthermore, to direct the AGV to move outside a highly congested path when multiple consecutive AGVs are waiting for more than a certain number of AGVs in the module section, increase the dynamic Dijkstra algorithm cost of the congested path. After the AGV departs, the dynamic Dijkstra algorithm cost of the corresponding path should be reset to the initial value. The stopped vehicle avoidance method can solve the AGV congestion problem, thereby preventing a loop cycle and minimizing the number of instances in which the AGV has to return to its origin.

4 RESULTS AND DISCUSSION

4.1 Simulation Assumptions

The assumptions used in the simulation of the proposed algorithm are as follows: First, an AGV can immediately start and stop. Second, the processing time of the workers at the station is 8 seconds. Third, the time it takes for the AGV to pass through one unit path is 2.5 seconds. Forth, the MSU loading time is 2 seconds. Fifth, an AGV's speed is constant regardless of the presence or absence of an MSU. The dynamic Dijkstra algorithm finds a route that maximally reduces direction deviation by modifying the path cost.

4.2 Performance Test

The DRR method (Zhao et al. 2020) has the main disadvantage: the AGV has to wait until no AGV remains on an overlapping node set between each shortest route. Therefore, we compared the performance of the proposed algorithm with those of three methods: the modified DRR method, the bidirectional infeasible lower-bound (ghost mode) method, and the unidirectional infeasible lower-bound (ghost mode) method.

The modified DRR method is based on the Dijkstra algorithm, which considers the waiting time in the cost, making it possible to choose between waiting until another AGV passes and detouring. Moreover, the modified DRR method is a static routing algorithm that directs the AGV to move from origin to destination through a route determined at the origin.

In the infeasible lower-bound method, AGVs can pass each other without physical collision, and only the station and MSU capacity constraints exist. Therefore, an AGV only moves along the shortest route. The infeasible lower-bound method is implemented in the bidirectional and unidirectional layouts.

The modified DRR method is an ineffective benchmark for the proposed algorithm because a node on the intended route of an AGV cannot be early passed by another AGV that chooses the route later. Therefore, we experimentally compared the proposed algorithm with the infeasible lower-bound (ghost mode) method.

Figure 11 shows the layouts for the performance test. 20 AGVs were used, corresponding to a 5% density level with respect to the total number of CPs. The density is calculated as follows:

$$Density = \frac{The\ number\ of\ AGVs}{The\ number\ of\ CPs\ in\ the\ layout}.$$

In the modified DRR method experiment, ten sets were used, in which 20000 tasks were randomly stored. The experiment was conducted over a simulation time of 2 days. Moreover, we conducted ten experiments with infeasible lower bounds over a simulation time of 25 hours by increasing the load factor.

Figure 12 compares the performance of the proposed algorithm and the modified DRR algorithm. In all ten runs of the experiment, the AGV system based on the proposed algorithm completes all 20,000 tasks within two days, without deadlock, and processes 46% more tasks than the system based on the modified DRR method. In contrast, the system based on the modified DRR method enters deadlock five times, even under the 5% density level. Figure 13 compares the proposed algorithm and the infeasible lower-bound (ghost mode). The AGV system based on the proposed algorithm completes 9.7% fewer tasks than the system based on the bidirectional infeasible lower-bound method. Moreover, the proposed algorithm completes 10% more tasks than the unidirectional infeasible lower-bound method.

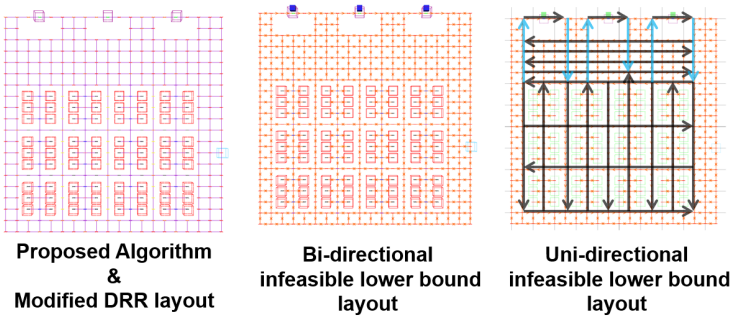

Figure 11: Layout for performance test.

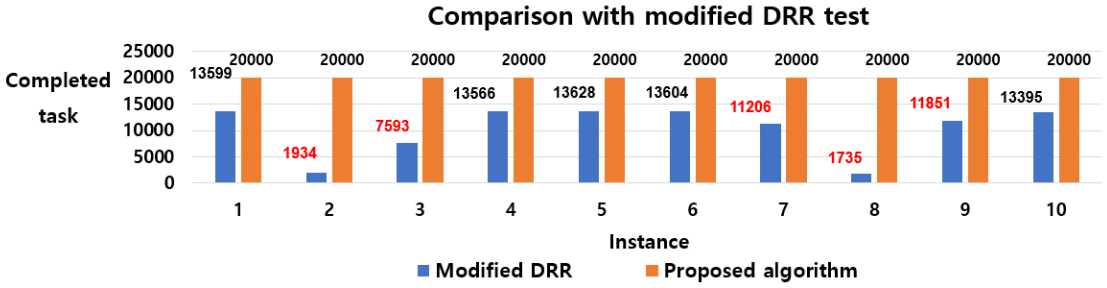

Figure 12: Performance of the proposed algorithm and the modified DRR method.

4.3 Computational Complexity of the Proposed Algorithm

Let N be the total number of CPs in the layout, n the number of CPs in a route from origin to destination, E the total number of unit path in the layout, and v the total number of AGVs; then the computational complexity required for an AGV to process one task according to the proposed algorithm is as follows:

- Path reservation for 1 AGV
  - $O(|n|)$
- Dynamic Dijkstra algorithm for 1 AGV
  - $O(|n||E|+|n||E|log|E|)$
- Loop cycle avoidance for 1 AGV
  - $O(|n||E|+|n||E|log|E|)$
- **Total**
  - $\mathbf{O}(|\mathbf{v}||\mathbf{n}||\mathbf{D}|)$
    where $O(D) = O(|E|+|E|log|E|)$, $O(D)$: Computational complexity of Dijkstra algorithm

The computational complexity of the proposed algorithm is highly dependent on that of the Dijkstra algorithm. The proposed algorithm is not significantly affected by the increase in the number of AGVs.

4.4 Performance Analysis

The reasons for the excellent performance of the algorithm are as follows: First, the algorithm minimizes the traffic congestion caused by the prolonged stop of an AGV on a narrow path. Second, the algorithm effectively avoids deadlock by reserving the minimum path length. Third, the dynamic Dijkstra algorithm continuously updates the efficient routes.

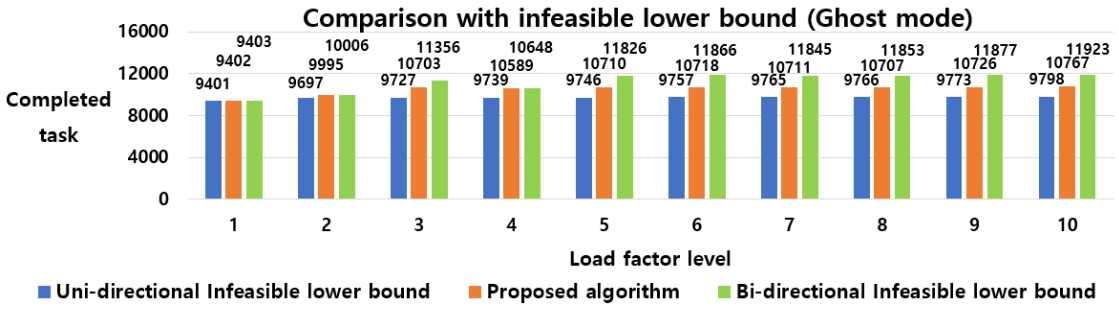

Figure 13: Performance of the proposed algorithm and the infeasible lower-bound (ghost mode) method.

5 CONCLUSION

This paper proposes an algorithm to avoid deadlock in a fulfillment warehouse environment in which dozens to hundreds of AGVs are continuously assigned tasks. The system based on the proposed algorithm completes approximately 9.7% fewer tasks per unit time than the system based on the bidirectional infeasible lower-bound (ghost mode) method. Furthermore, the proposed algorithm can be applied to routing algorithms other than the Dijkstra algorithm. The computational cost can be considerably reduced depending on the applied routing algorithm. Also, a better bidirectional AGV system can be achieved using a routing algorithm that more efficiently considers AGV congestion.

However, the proposed algorithm is not extensible to layouts other than those examined in this study. The study's layout is characterized by a highway that can reduce AGV congestion among the unit modules. An additional consideration is required in situations lacking such a highway. In the future, we will modify the algorithm so that it can be applied to a Kiva robot. A Kiva robot can also use the storage area as a path when an MSU is not being transported. Finally, although the proposed algorithm avoids deadlock as much as possible, deadlocks may nevertheless occur if the number of AGVs is large. The proposed algorithm performs optimally under appropriate numbers of AGVs and density levels in a vast fulfillment center.

ACKNOWLEDGMENTS

This work was supported by the Ministry of Education of the Republic of Korea and the National Research Foundation of Korea (NRF-2021R1A2C3008172).

A Method to find next CP

Algorithm 1 Method to find next CP.

Require: Set H to heading location (moving) or current location (stopped), Set B to previous location, Set D to destination

1: **if** H is ICP **then**
2: **if** AGV has saved route **then**
3: finalize next CP to first element of saved route
4: **else**
5: go to line 7
6: **else**
7: do dynamic Dijkstra algorithm from H to D
8: **if** dynamic Dijkstra algorithm return route **then**
9: set N to first element of route

```
10:     if loop cycle occur if AGV goes to N then
11:       change P_{H,HtoNdirection} to 0 temporary and go to line 7
12:     else
13:       finalize next CP to N and initialize any temporary modified PI value
14:   else
15:     if H is one of storage area then
16:       wait until dynamic Dijkstra algorithm find the route
17:     else if H is OI then
18:       if AGV has saved route then
19:         set next CP N to first element of saved route
20:         if P_{H,HtoNdirection} = 1 then
21:           while N is not Intersection do
22:             if P_{H,HtoNdirection} = 1 then
23:               set H to N, set N to next CP of N on the same direction
24:               go to line 21
25:             else
26:               remove the saved route, change P_{H,HtoNdirection} to 0 temporary
27:               go to line 33
28:           if B is ICP connected to the storage area then
29:             go to line 13
30:           else
31:             go to line 33
32:         else
33:           Set N be a closest CP to the destination among the CPs attached to H.
34:           go to line 21
35:       else if H is II then
36:         wait until Dijkstra algorithm find the route
37:       else if B is ICP connected to the OI then
38:         go to line 21
39:       else
40:         go to line 33
41:     else
42:       go to line 18
```

REFERENCES

Breton, L., S. Maza, and P. Castagna. 2006. "A Multi-Agent based Conflict-Free Routing Approach of Bi-directional Automated Guided Vehicles". In *Proceedings of the 2006 American Control Conference*, edited by E. Misawa, 6. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Fanti, M. P. 2002. "A Deadlock Avoidance Strategy for AGV Systems Modelled by Coloured Petri Nets". In *Proceedings of the Sixth International Workshop on Discrete Event Systems*, edited by M. Silva, A. Giua, and J. M. Colom, 61–66. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Fragapane, G., R. De Koster, F. Sgarbossa, and J. O. Strandhagen. 2021. "Planning and Control of Autonomous Mobile Robots for Intralogistics: Literature Review and Research Agenda". *European Journal of Operational Research* 294(2):405–426.

Fransen, K., J. Van Eekelen, A. Pogromsky, M. A. Boon, and I. J. Adan. 2020. "A Dynamic Path Planning Approach for Dense, Large, Grid-Based Automated Guided Vehicle Systems". *Computers & Operations Research* 123:105046.

Jager, M., and B. Nebel. 2001. "Decentralized Collision Avoidance, Deadlock Detection, and Deadlock Resolution for Multiple Mobile Robots". In *Proceedings of 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding*

the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180), Volume 3, 1213–1219. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Jia, F., C. Ren, Y. Chen, and Z. Xu. 2017. "A System Control Strategy of a Conflict-Free Multi-AGV Routing Based on Improved A Algorithm". In *2017 24th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*, edited by F. Alam, 1–6. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Kim, C. W., and J. M. Tanchoco. 1991. "Conflict-Free Shortest-Time Bidirectional AGV Routeing". *The International Journal of Production Research* 29:2377–2391.

Małopolski, W. 2018. "A Sustainable and Conflict-Free Operation of AGVs in a Square Topology". *Computers & Industrial Engineering* 126:472–481.

Maza, S., and P. Castagna. 2001. "Conflict-Free AGV Routing in Bi-directional Network". In *ETFA 2001. 8th International Conference on Emerging Technologies and Factory Automation. Proceedings (Cat. No. 01TH8597)*, edited by N.Komoda, M. Silva, Y.Trinquet, H. Rowlands, and D.Collard, Volume 2, 761–764. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Sharon, G., R. Stern, A. Felner, and N. R. Sturtevant. 2015. "Conflict-Based Search for Optimal Multi-Agent Pathfinding". *Artificial Intelligence* 219:40–66.

Sun, X., Y. Zhao, S. Shen, K. Wang, X. Zheng, and Y. Shi. 2018. "Scheduling Multiple AGVs with Dynamic Time-Windows for Smart Indoor Parking Lot". In *2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design ((CSCWD))*, edited by W. Shen, J. Luo, J. P. Barthès, F. Dong, J. Zhang, and H. Zhu, 864–868. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Triwiyatno, A., and M. A. Riyadi. 2020. "Conflict-Free Dynamic Route Multi-Agv Using Dijkstra Floyd-Warshall Hybrid Algorithm With Time Windows". *International Journal of Electrical and Computer Engineering* 10:3596.

Wu, N., and M. Zhou. 2001. "Resource-Oriented Petri Nets in Deadlock Avoidance of AGV Systems". In *Proceedings of 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, edited by W. H. Kwon, Volume 1, 64–69. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Wu, N., and M. Zhou. 2004. "Modeling and Deadlock Control of Automated Guided Vehicle Systems". *IEEE/ASME Transactions on Mechatronics* 9:50–57.

Wu, N., and M. Zhou. 2007. "Shortest Routing of Bidirectional Automated Guided Vehicles Avoiding Deadlock and Blocking". *IEEE/ASME Transactions on Mechatronics* 12:63–72.

Yan, X., C. Zhang, and M. Qi. 2017. "Multi-AGVs Collision-Avoidance and Deadlock-Control for Item-to-Human Automated Warehouse". In *2017 International Conference on Industrial Engineering, Management Science and Application (ICIMSA)*, edited by X. Huang and N. Joukov, 1–5. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Zhang, Z., Q. Guo, J. Chen, and P. Yuan. 2018. "Collision-Free Route Planning for Multiple AGVs in an Automated Warehouse Based on Collision Classification". *IEEE Access* 6:26022–26035.

Zhao, Y., X. Liu, G. Wang, S. Wu, and S. Han. 2020. "Dynamic Resource Reservation Based Collision and Deadlock Prevention for Multi-AGVs". *IEEE Access* 8:82120–82130.

Zhao, Y., X. Liu, S. Wu, and G. Wang. 2021. "Spare Zone Based Hierarchical Motion Coordination for Multi-AGV Systems". *Simulation Modelling Practice and Theory* 109:102294.

AUTHOR BIOGRAPHIES

KANGMIN KIM received his master's degree in Industrial and Systems Engineering at the KAIST. His research interests include continuous and sustainable traffic control, scheduling, and dispatching of multi-bidirectional AGVs. His email address is kimkangmin49@gmail.com.

CHANGHYUN CHUNG is a Ph.D. candidate in Industrial and Systems Engineering at the KAIST. His research interests include deadlock avoidance, reinforcement learning-based AGV routing, and simulation-based optimization. His email address is cch0720@kaist.ac.kr.

YOUNGJAE JANG received his Ph.D. degree in mechanical engineering from the Massachusetts Institute of Technology (MIT). His current research includes the smart factory and automated material handling systems (AMHS) and manufacturing automation. He is currently the Director of the Synustech-KAIST AI-based Automated Material Handling System Research Center, leading the multi-million dollar industry collaborative research project with the industry partner, Synustech, Inc, a global factory automation system provider. He is currently an associate editor of IEEE Transactions for Automation Science and Engineering and Computers and Industrial Engineering Journal. He is also a founder and CEO of the DAIM Research Co., Ltd., South Korea. His email address is yjang@kaist.ac.kr.