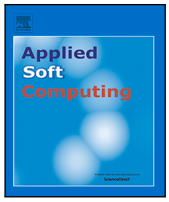
Applied Soft Computing Journal 161 (2024) 111725

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc



Deadlock prevention and multi agent path finding algorithm considering physical constraint for a massive fleet AGV system ☆

Chang Hyun Chung [a], Young Jae Jang [a,b,*]

[a] Korea Advanced Institute of Science and Technology (KAIST), 291, Daehak-ro, Yuseong-gu, Daejeon, 34141, South Korea
[b] DAIM Research Corp., Duwon Building (2nd Floor), 20 Banpo-daero 28-gil, Seocho-gu, Seoul, 06648, South Korea

ARTICLE INFO

Keywords:
Deadlock
Path planning
Automotive industry
Fulfillment system
Simulation

ABSTRACT

This study introduces deadlock avoidance algorithms for Automated Guided Vehicles (AGVs) within warehouse fulfillment systems. Numerous AGVs are tasked with transporting requests to deliver shelves to workers stationed at various locations. The pathfinding challenge within such systems is identified as the Multi-Agent Pickup and Delivery (problem) and has been extensively explored in computer science and AI. Nevertheless, most prior research was conducted in environments that are discretized and allow circular movements of AGVs, potentially leading to deadlocks in real-world applications. Our proposed algorithm leverages a dynamic path block method and path reservation. Specifically, it limits movement to certain edges when AGVs navigate along reserved paths, while the remaining AGVs plan their routes using unrestricted edges. We show that our algorithm not only effectively prevents deadlocks but also scales well in environments with a high number of AGVs.

1. Introduction

The AGV system is the most commonly utilized automated material handling system (AMHS) in warehouse automation for logistics and manufacturing, as noted by Wurman et al. [1]. AGVs are tasked with fulfilling transport requests by delivering items to machines or workers within the warehouse. The necessity to maintain high throughput and avoid idleness among workers or machines prompts the operation of multiple AGVs simultaneously in confined spaces. Consequently, AGV-based warehouses often feature numerous narrow passages where AGVs lack the capability to bypass one another, presenting a significant challenge for efficient space utilization and system flow.

AGV systems are categorized into unidirectional and bidirectional layouts. The unidirectional layout, while advantageous for controlling multiple AGVs, can lead to inefficient movements in certain scenarios. On the other hand, the bidirectional layout facilitates allow more efficient AGV movements but introduces additional risks for collisions and deadlocks. Given that the movements in a bidirectional layout cover all possible movements achievable in a unidirectional layout, it is evident that a bidirectional system, with appropriate control mechanisms in place, is generally deemed more efficient than its unidirectional counterpart.

The most significant challenge in managing multiple AGVs within a bidirectional layout is the risk of deadlock. Deadlock is defined as a situation where two or more processes are indefinitely delayed because each is waiting for a resource held by another. In warehouse operations, AGVs are analogous to processes, and the navigable points within the warehouse serve as the contested resources. Deadlocks in bidirectional AGV systems fall into two main categories: heading-on deadlock and loop deadlock (Fig. 1). Heading-on deadlock occurs when AGVs attempt to navigate the same path from opposite directions, effectively blocking each other. Loop deadlock emerges when three or more AGVs each attempt to occupy the positions of the others, creating a circular blockage that prevents any movement.

Numerous studies have tackled the challenge of operating multiple AGVs without deadlock in a bidirectional layout by framing it as the Multi-Agent Path Finding (MAPF) problem. The objective of MAPF is to identify collision-free routes that enable a set of agents to achieve their pre-specified destinations. This problem is recognized as NP-hard [2,3], and a variety of algorithms have been introduced to address it. One conventional method is the centralized planner, which devises a global plan through integer programming [4,5]. However, this method struggles with scalability and is hindered by the curse of dimensionality as the agent count rises. In recent years, AI

☆ This work was supported by the Ministry of Education of the Republic of Korea and the National Research Foundation of Korea (NRF-2021R1A2C3008172). The proposed approaches and required data and information are supported by DAIM Research.

* Corresponding author at: Korea Advanced Institute of Science and Technology (KAIST), 291, Daehak-ro, Yuseong-gu, Daejeon, 34141, South Korea.
E-mail addresses: cch0720@kaist.ac.kr (C.H. Chung), yjang@kaist.ac.kr (Y.J. Jang).

https://doi.org/10.1016/j.asoc.2024.111725
Received 9 July 2023; Received in revised form 20 April 2024; Accepted 30 April 2024
Available online 16 May 2024

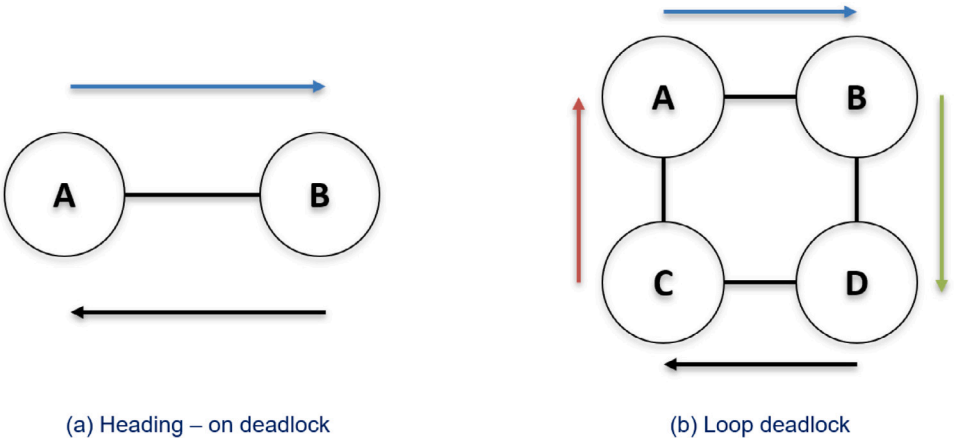


Fig. 1. Heading-on deadlock and loop deadlock.

researchers have advanced various algorithms to discover optimal solutions, including search-based algorithms like conflict-based search and its modifications [6–9], decentralized and distributed approaches such as prioritized planning [10–12], and learning-based techniques [13–15].

Algorithms designed to address the Multi-Agent Path Finding (MAPF) problem focus on creating comprehensive routes that enable multiple agents to reach their designated destinations. This approach is commonly described as "one shot" and "offline". However, in real-world applications, AGV systems frequently encounter dynamic tasks and uncertain environments, necessitating a "lifelong" and "online" approach. The practical version of the MAPF problem, known as the Multi-Agent Pickup and Delivery (MAPD) problem [16], captures more realistic scenarios encountered in automated warehouses. To adapt to these dynamic and uncertain characteristics, research on MAPD problems has been conducted [17,18], aiming to develop solutions that are more aligned with the complexities of real-life AGV system operations.

Most research in this area has been conducted within a discrete time, grid-based environment, frequently neglecting the physical characteristics and constraints of AGVs. Notably, loop deadlock, which results from the size of AGVs and the layout of the system, is not considered in many previous studies. As a result, the paths derived from such research are often not feasible for actual hardware implementation. While the Dynamic Reservations for Robustness (DRR) method [19] does take into account loop deadlock and offers deadlock-free paths for agents, it still imposes excessive constraints on AGV movement to avert loop deadlock scenarios. Furthermore, existing studies commonly assume that AGV movements are perfectly executed over time, an assumption that does not hold in practical applications, where variability and unpredictability are common.

This study focuses on the Multi-Agent Path Finding (MAPF) problem, taking into account the physical constraints of hardware in a bidirectional layout. We propose a Dynamic Path Block (DPB) algorithm designed to prevent deadlock, enabling dozens to hundreds of AGVs to navigate without issues in actual hardware systems. The proposed algorithm mirrors the control method of real-world AGVs; when an AGV receives a driving command, it reserves the necessary points and limits the movements of other AGVs. Consequently, other AGVs calculate the shortest path among unrestricted paths in real-time. This algorithm is compatible with any path planning algorithm, offering broad applicability. We validated the proposed algorithm using the simulation software Applied Materials AutoMod™ (version 14.0), which accurately incorporates the physical constraints of AGVs.

Table 1 offers a comprehensive overview of the existing literature on Multi-Agent Path Finding (MAPF) and the Multi-Agent Pickup and Delivery (MAPD) problem. The table is organized into several columns

Table 1
Overview: MAPF and MAPD research.

	Con.	Lifelong	Opt.	Loop.	Exec.	Obj.
CBS [Sharon *et al.*2015]	×	×	✓	×	time	soc
MAPF-POST [Ma *et al.*2019]	×	×	✓	✓	event	soc
MCCBS [Li *et al.*2019]	×	×	✓	×	time	soc
CCBS [Andreychuk*et al.* 2019]	✓	×	✓	×	time	soc
TP and TPTS [Ma *et al.*2019]	×	×	×	×	time	adt
RHCR [Li *et al.*2021]	×	✓	×	×	time	adt
DRR [Zhao *et al.*2019]	×	×	×	✓	event	soc
DPB (Proposed)	✓	✓	×	✓	event	adt

to classify the studies based on different criteria: 'Con.' denotes studies conducted in a continuous-time environment; 'Lifelong' refers to an online setting where AGVs are perpetually tasked with new assignments; 'Opt.' indicates the optimality of the provided solution; 'Loop.' highlights whether the research considers loop deadlock scenarios; 'Exec.' distinguishes the execution of plans into time-based and event-based categories; and 'Obj.' is divided into the sum of costs (soc) and average delivery time (adt), representing the primary objectives analyzed in each study.

The contributions of this study are as follows: Firstly, we propose a deadlock avoidance algorithm that effectively prevents both loop deadlock and heading-on deadlock, which occur in actual hardware. This algorithm introduces a framework for the efficient control of a large number of AGVs. Secondly, the method proposed in this study is distinct from the path planning algorithm, ensuring compatibility with any path planning algorithm. This separation represents a significant advantage for future performance enhancements.

2. Problem definition

This study concentrates on the Multi-Agent Pickup and Delivery (MAPD) problem within a bi-directional layout of a warehouse fulfillment system, characterized as a lifelong environment with continuous task assignments [16]. An MAPD instance comprises n AGVs $A = \{a_1, a_2, \ldots, a_n\}$ and a directed graph $G = (N, E)$. Nodes N signify the basic unit of AGV movement, while Edges E denote the connectivity between nodes, permitting AGVs to traverse from one node to another along these edges. Moreover, this research aims to tackle the challenge of loop deadlock, a significant obstacle observed in physical hardware systems.

2.1. Working environment description

Fig. 2 illustrates the warehouse fulfillment system targeted in this study. Shelves, which are the basic units of transportation requests, are

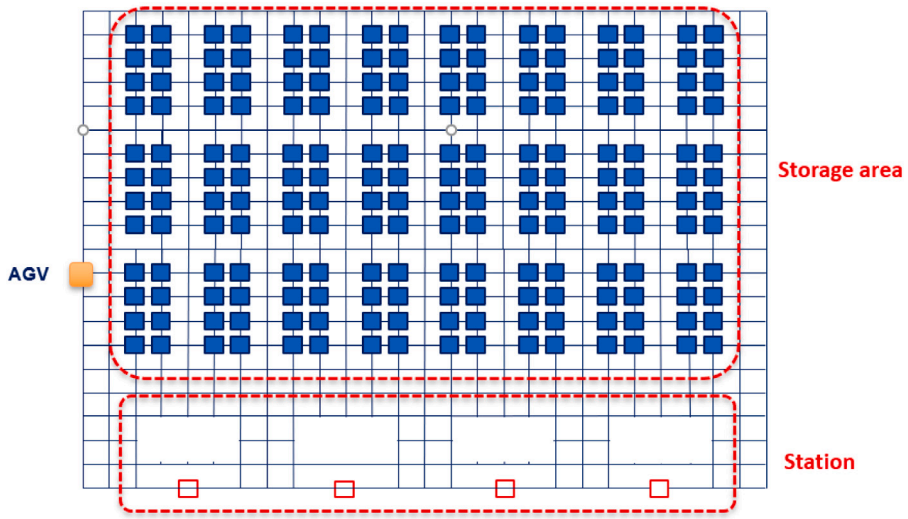


Fig. 2. Components of a fulfillment warehouse.

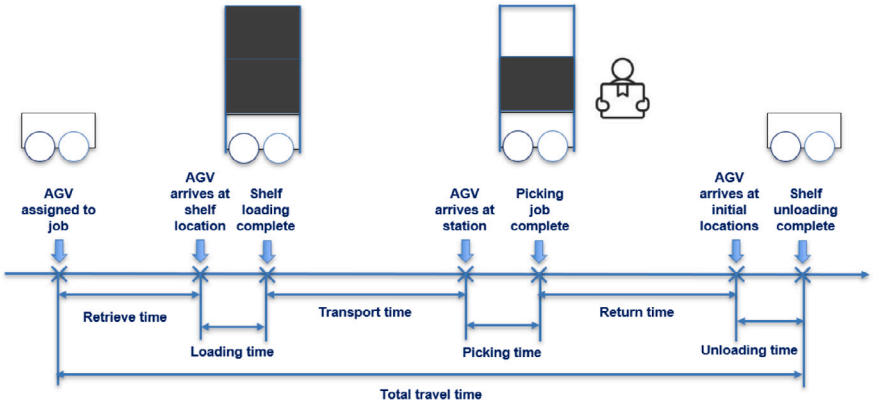


Fig. 3. Job transportation timeline from the perspective of AGV.

located in the storage area, and AGVs perform the role of delivering shelves to the station where the worker carries out the picking. The transportation request for a shelf is denoted as *job*.

Fig. 3 illustrates the steps an AGV takes to perform a job across several events. When a job is initiated, an AGV is assigned to the corresponding job and moves to retrieve the shelf. This process is referred to as *retrieve travel*, with the duration defined as *retrieve time*. The AGV then loads the shelf and travels to the working station; this process and its duration are termed *transport travel* and *transport time*, respectively. Following the completion of the pickup process by the worker, the AGV returns the shelf to its initial position; this process and its duration are referred to as *return travel* and *return time*. *Blocking time* is defined as the duration during which AGVs are halted due to the path planning of other AGVs. *Total travel time* represents the time required to complete a single task from start to finish.

Heading-on and loop deadlocks can occur at any location within a bidirectional system. However, AGVs in the storage area are particularly susceptible to head-on deadlocks due to the warehouse layout structure. AGVs transporting shelves are unable to pass through the shelves (indicated by the blue box) within the storage area, thus requiring navigation through narrow corridors. Stations represent another critical area where deadlocks frequently occur, due to the high density of AGVs. Numerous AGVs visit these areas and must wait until a worker or machine completes the tasks. This high density of AGVs often leads to situations where AGVs attempt to move to a node already occupied by another AGV.

The aim of this study is to propose a deadlock avoidance algorithm that functions effectively within a bidirectional AGV system. Consequently, this study does not cover decision-making processes such as assigning jobs to AGVs or determining the return locations for shelves. For job assignment, a first-come, first-served principle is employed, and the return location for a shelf is designated as its original position. The performance of the algorithms is evaluated based on *average total travel time*, the number of tasks completed, and *blocking time*, rather than *makespan*, due to the continuous generation of jobs.

3. Dynamic path block algorithm

In this section, we introduce methods designed to prevent deadlock in bi-directional AGV systems. Initially, we categorize the AGV's path into two types: a reserved path and a desired path. Utilizing the reserved path information, we then implement movement restrictions on other AGVs that could potentially lead to deadlock. This involves the application of pathblocks and pointblocks to individually address and prevent heading-on deadlock and loop deadlock. Ultimately, AGVs calculate their paths taking into account the restrictions imposed.

3.1. Driving structure and path configuration of AGVs

In actual AGV systems, driving commands are issued for relatively short segments of the path and are continuously updated, as opposed to assigning commands for the entire path to the destination. To

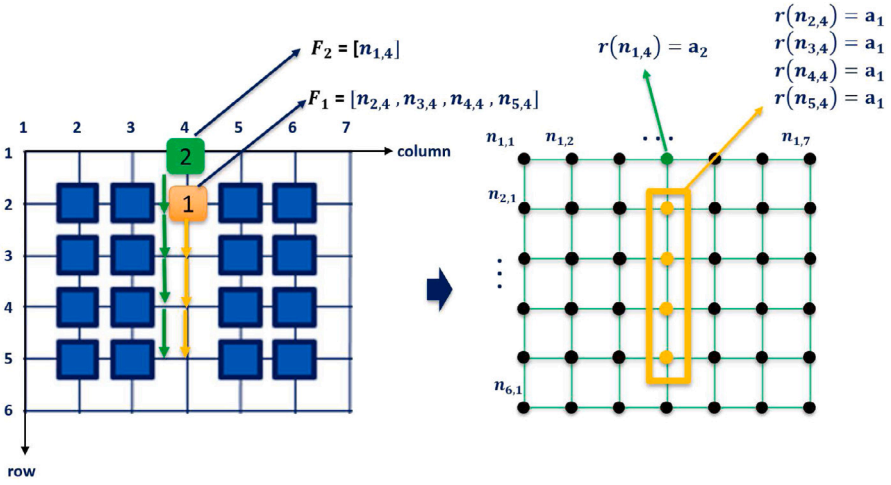


Fig. 4. Path reservation of AGV.

accommodate this operational procedure, we propose differentiating AGV paths into reserved paths and desired paths: the desired path $P(a)$, which denotes the path an AGV intends to travel, and the reserved path $F(a)$, which it can actually reserve and traverse.

The desired path, $P(a)$, for an AGV is determined by a path planning algorithm. The AGV then checks if it can reserve the nodes along this desired path. Reserving a node means that the AGV claims and moves to that specific node. Each node has a reservation state, $r(n)$, indicating which AGV has reserved it. A node can be occupied by only one AGV at any given time.

Fig. 4 illustrates the node reservation process for AGVs. In this scenario, both the green and yellow AGVs plan to navigate through the same corridor. The yellow AGV successfully reserves the required nodes and proceeds along its path, while the green AGV is forced to wait until the nodes are released by the yellow AGV. If an AGV intends to move through nodes already occupied by another AGV, it enters a blocking state, where it waits for the node to become available for reservation. Once an AGV vacates a node, the reservation for that node is released, allowing any AGV in a blocking state to reserve and travel through that node.

The maximum number of reserved nodes, m, is a user-specified parameter. As the value of m increases, the AGV conducting the reserved path can receive practical driving commands over longer distances, allowing for faster driving without the need for deceleration. However, this also imposes greater constraints on the movements of other AGVs. Conversely, as the value of m decreases, the AGV can only receive practical driving commands over shorter distances, which may necessitate repeated acceleration and deceleration but reduces the constraints on the movements of other AGVs. Therefore, m represents a trade-off between driving efficiency and the constraints imposed on other AGVs.

Definition 1 (*Reserve State*)**.** If a node n is reserved by an AGV a_i, the value of the reserve state $r(n)$ is a_i. If no AGV has reserved node n, then the reserve state $r(n) = 0$.

The nodes reserved by an AGV form the AGV's reserved path, which is a subset of the desired path and remains fixed once determined. However, nodes included in the desired path but not in the reserved path may change based on the situation when re-planning is required.

3.2. Preventing deadlocks

The reserve state and reserved path provide information about the nodes that an AGV will definitely traverse. This information is utilized to limit the movements of other AGVs that could potentially lead to collisions or deadlocks. As outlined in prior research, we conceptualize the bi-directional AGV system as a directed graph $G = (N, E)$. The status of each edge signifies the availability of the corresponding edge.

Definition 2 (*Edge Status*)**.** The state of an edge $s(e(v_a, v_b))$ can be either available ($s(e(v_a, v_b)) = 1$) or unavailable ($s(e(v_a, v_b)) = 0$).

Once an AGV reserves nodes, the pathblock function changes the edge status to 0 for the opposite direction of the edge that the AGV is driving on. The pathblock function aims to prevent AGVs from planning paths that could lead to head-on deadlock. However, it is ineffective against loop deadlock. Since the pathblock function only limits AGVs from moving in the opposite direction of the reserved path, it does not restrict loop-shaped path planning. Loop deadlock occurs when an AGV stops due to the absence of available nodes for reservation. Loop deadlock ensues when other AGVs attempt to move to the stopped AGV's current position and form a cycle. To avert loop deadlock, the pointblock function restricts the edge status of edges leading to the node where the AGV has stopped, except for the edge along which the AGV moved into the stopped state.

Fig. 5 illustrates the dynamics of the DPB algorithm. Once an AGV reserves nodes, movement in the opposite direction on specific edges is prohibited. The pathblock function only restricts movement in the opposite direction, allowing AGVs to still plan paths to traverse the same node.

Theorem 1. *In an environment where AGVs navigate based on reserved paths, the set of nodes restricted by the pointblock function does not form a cycle and remains unreachable from all other nodes.*

Proof. We prove by contradiction. Suppose there exists a set of nodes N' that forms a strongly connected component due to the pointblock function. The set of all nodes not belonging to N' is denoted as N''. The pointblock function restricts only one available edge for a reserved path with a length of 1, which implies that every node n_i in N' with an incoming edge from node n_j in N'' must be restricted by the pointblock function. Consequently, all AGVs that stop at nodes in N' must have originated from adjacent nodes within N'', creating a loop transition.

However, in an environment where AGVs travel based on reserved paths, loop transitions cannot exist. Therefore, the scenario where the pointblock function forms strongly connected components while multiple AGVs are stopped does not occur. □

Since Theorem 1 guarantees that pointblock function does not forms strongly connected component, AGVs can dynamically find loop-deadlock free path to its destination.

Since pathblock function and pointblock function only operate on nodes in the AGV's desired path, they cannot capture all information

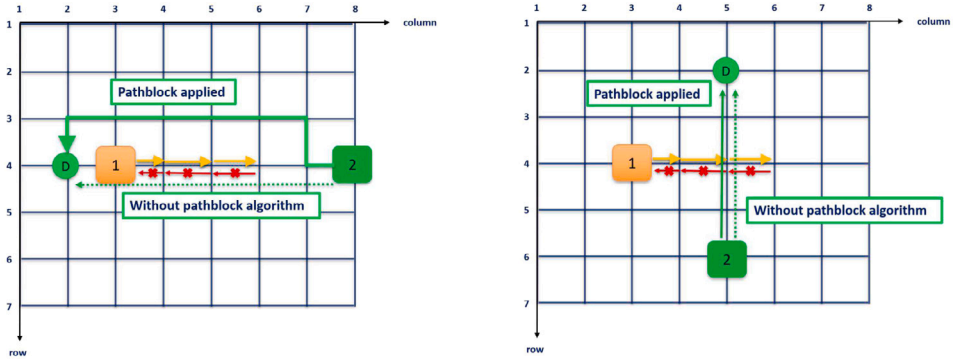


Fig. 5. Pathblock and pointblock.

about AGVs movements. In cases where an AGV does not stop, sufficient constraints may not be applied to the last node on the reserve path that the AGV has reserved. This can lead to deadlock or detour which decrease system efficiency. The points where detours or deadlock occur are typically nodes with an outdegree of two or less. In this study, the problem of AGV path selection in an aisle layout is defined using a directed graph. In such graph, when an AGV enters a node with an outdegree of 2 or less, there is only one edge that AGV move along to prevent detours.

Extend path function is triggered when an AGV tries to reserve nodes for its desired path, and the last available node for reservation has an outdegree of 2 or less. When the last reservable node on an AGV's desired path has an outdegree of 2 or less, the function checks the reachability of all nodes up to *target node*. *Target node* denotes a destination node of AGV or a node with an outdegree of 3 or more. The reserved path is extended only if all nodes in desired path until *target node* are reachable. Algorithm 2 demonstrate detailed procedure of extend path function.

An AGV communicates with AGV control system (ACS) and updates its information. When an AGV arrives at certain nodes and proceedes to the next node, it sends message to ACS and releases the reservation of the node it has passed through (n_p). The extend path function check whether AGV can reserve additional nodes. Since the reserved path of AGV is unchangeable, the extend path function begins extension of the reserved path from the n_s which is the last reserved point by the AGV (represented as $F(a_i)[-1]$).

The fundamental idea behind the pathblock and pointblock functions is to temporarily deactivate edges in a directed graph. In some cases (e.g., existence of articulation node or strong bridge), deactivating edges may invoke the case that AGVs cannot find feasible paths. However, in this study, we focus MAPD problem based on warehouse fulfillment system in which articulation nodes or bridge does not exists. Furthermore, AGVs are able to wait indefinitely under a shelf after completing their assigned task and extend path function prevents AGVs from stopping at nodes with an outdegree of 2 or less. We experimentally demonstrate that our algorithm prevents deadlocks efficiently in next chapter.

Algorithm 1 DPB

```
while system is operating do
    read message sent from AGV a_i
    if message type = node claim then
        set n_p to F(a_i)[0]
        call ReleasePoint(n_p)
    end if
    if message type = path extension then
        set n_d to destination location of a_i
        set n_s to F(a_i)[−1]
        set P(a_i) to A*(n_s,n_d)
        set F(a_i) to ExtendPath(a_i, P(a_i))
        if length(F(a_i)) = 1 then
            set n_b to F(a_i)[0]
            call BlockPoint(n_b)
        else
            call BlockPath(F(a_i))
        end if
    end if
end while

procedure ReleasePoint(n)
    set reserve state r(n) = 0
    for all n_n satisfying e(n_n, n) ∈ E do
        set edge status s(e(n_n, n)) to 0
    end for
end procedure

procedure BlockPoint(n_b)
    for ∀n_n s.t e(n_e, n_b) ∈ E do
        set edge status s(e(n_e, n_b)) to 1
    end for
end procedure

procedure BlockPath(n_b)
    for i = 0...len(F(a_i)) − 2 do
        set edge status s(e(F(a_i)[i + 1], F(a_i)[i]) to 1
    end for
end procedure
```

4. Numerical experiments

In this section, the performances of the proposed algorithms are evaluated through numerical experiments. Typically, the verification of AGV path planning algorithms is conducted in a virtual grid-based environment with discrete transitions. However, given that this study addresses physical challenges that may arise in real hardware systems, verification was conducted using simulation software that incorporates physical constraints. Two metrics were employed as performance measures: the number of completed tasks, which represents the total tasks completed over the time periods, and the average traveling time per job.

4.1. Simulation environment

Most previous research has been conducted in virtual environments, focusing on the visualization of Automated Guided Vehicle (AGV) movements. However, to demonstrate the applicability of our proposed algorithm, we utilized the simulation program Applied Materials' AutoMod™ 14.0, which accounts for the physical constraints of the hardware. In the AutoMod simulation, the movement of AGVs is implemented with precision, closely mirroring that of actual hardware. Fig. 7 illustrates AGVs performing the rotation and transportation of shelves. The acceleration speed, maximum velocity, and rotation speed

Algorithm 2 Extend path function

```
function ExtendPath(a,P(a))
    extensionlist = [ ]
    Extensionflag = True
    for i = 1...len(P(a)) do
        if reserve state r(P(a)[i]) = 0 then
            if Extensionflag = True then
                extensionlist.append(P(a)[i])
                if len(extensionlist) > w and (d(P(a)[i] > 2) or d(P(a)[i]) is destination of AGV a then
                    break
                end if
            else
                if len(extensionlist) > w and (d(P(a)[i] > 2) or d(P(a)[i]) is destination of AGV a then
                    break
                end if
            end if
        else
            Extensionflag = False
            reserving AGV a_r = r(P(a)[i])
            n_x = next desired node of a_r
            if n_x = r(P(a)[i − 1]) then
                while d(extensionlist[−1]) > 2 do
                    extensionlist.pop
                end while
                break
            end if
        end if
    end for
    for j = 1...len(extensionlist) do
        F(a).append(extensionlist(j))
        r(extensionlist(j)) = a
    end for
end function
```

Table 2
AGV specification used in simulation test.

Parameters	Maximum velocity	Acceleration	Deceleration	Rotation speed	Stop distance	AGV size
Value	1 m/s	0.5 m/s^2	0.5 m/s^2	60°/s	0.3 m	1 m × 1 m

of an AGV are set at 0.5 m/s^2, 1 m/s, and 60°/s, respectively. The AGV specification used in the simulation is presented in Table 2. The simulation results indicate that the proposed algorithm can be directly applied to actual hardware.

The experiments were carried out in an environment modeled after a fulfillment center layout measuring 20 × 21, featuring 108 shelves and 3 stations. In this study, the completion of tasks associated with each shelf immediately triggers the generation of new tasks, aiming to test the efficiency of AGV path planning in preventing deadlock within a dynamic environment where AGVs are continuously operational. For job assignment, the first-come, first-served (FCFS) rule was applied. Fig. 6 displays the layout used in the numerical experiment. The AGV density varied from 2.5% to 10%, which exceeds the typical density range of 7%–8% used in prior studies on bi-directional fulfillment layouts. The density was calculated as follows.

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

4.2. Benchmark algorithm

We compared the proposed multi-agent planner with the DRR algorithm [19], which accounts for loop deadlock. However, the DRR algorithm may induce excessive delays when overlapping points exist between AGV paths to prevent deadlocks. Additionally, the lack of re-planning capability in DRR makes it challenging to directly compare it with the algorithm proposed in this study. For a fair comparison, we relaxed the driving conditions of the DRR to minimize unnecessary waiting times.

4.3. Experiment results

Before comparing the proposed algorithm with DRR and the uni-directional layout, we conducted a performance comparison of the proposed algorithm at different values of the maximum reserved length m, which influences the algorithm's effectiveness. Various values of m were tested to assess the system's performance in relation to the maximum reserved length. The detailed statistics of the completed number of tasks and blocking time, as well as the number of tasks completed, are summarized in Table 1. Fig. 8 illustrates the average total transport time per single job for each reserved length. We refer to the proposed algorithm with parameter m as DPB-m. It was observed that as the maximum reserve length decreases, the restriction on the path planning of other AGVs is diminished, leading to a reduction in blocking time. However, when m is set to 2, the actual moving

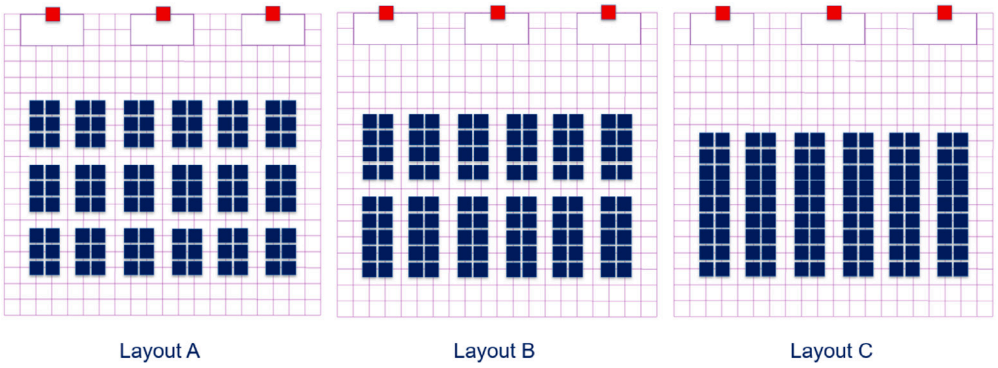


Fig. 6. Simulated layouts.

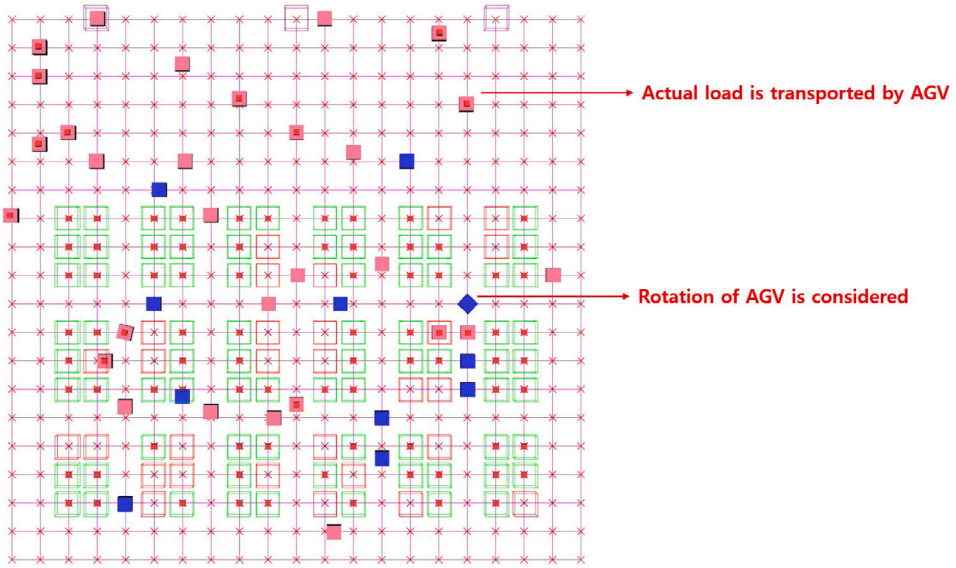


Fig. 7. Simulation of the system — Movie clip of the simulation (https://youtu.be/1e9uxOaXtsk).

commands received by AGVs become too short, causing repeated deceleration and acceleration. This results in an increase in total travel time and a decrease in the number of tasks completed. The optimal parameter m may depend on the AGV's acceleration, velocity, and decision-making cycle, but in the experimental environment of this study, DPB-3 exhibited the best performance across all layouts when m was set to 3. Consequently, we compare DPB-3 with DRR and the unidirectional layout.

Fig. 9 presents the outcomes of implementing the proposed algorithm in both unidirectional and bidirectional layouts, alongside the results of applying the DRR algorithm to bidirectional layouts. The proposed algorithm outperforms the DRR algorithm in all tested layouts, exhibiting more than a 25% improvement over the DRR algorithm. This performance gap widens with an increase in the number of AGVs, with the DRR algorithm even performing worse than the proposed algorithm when applied to a unidirectional layout. Furthermore, when comparing unidirectional to bidirectional layouts, the proposed algorithm demonstrates approximately a 15% improvement in bidirectional layouts (see Table 3).

4.4. Discussion

Based on the numerical results, we conclude that DPB algorithm efficiently prevents deadlocks while ensuring efficient operation of the AGV system. The DRR algorithm imposes excessive constraints on AGV movement to prevent deadlocks, resulting in inefficient movement of AGVs. As number of AGVs increases, DRR algorithm exhibits extremely inefficient movement. However, proposed algorithm minimizes blocking time of AGVs while preventing the total traveling time from becoming excessively large.

The DRR algorithm gives priority to the AGV that planned its path first. This approach is effective and prevents deadlocks quite efficiently when the number of AGVs is small. However, it imposes excessive constraints on AGVs that plan their paths later. AGVs that planned earlier (with a higher priority for reservation) do not suffer from the blocking phenomenon. In contrast, AGVs that planned their paths later experience significant blocking times, resulting in an average blocking time that is much higher than that of the proposed algorithm (Fig. 9).

As the number of AGVs increases, the interaction between AGVs increases, leading to an increase in AGV blocking time. To be more precise, as the number of AGVs increases in an environment, there is a higher density of AGVs, resulting in increased interaction between them, which is not influenced by an absolute number. Nevertheless, the findings of this study suggest that considering the interaction between AGVs is crucial for the efficient operation of bidirectional AGV systems. This becomes increasingly important as AGV density grows, and in the future, an algorithm capable of operating hundreds of AGVs in a brief period will be necessary.

The proposed algorithm shows good performance in high-density AGV environments without using the entire path information of other

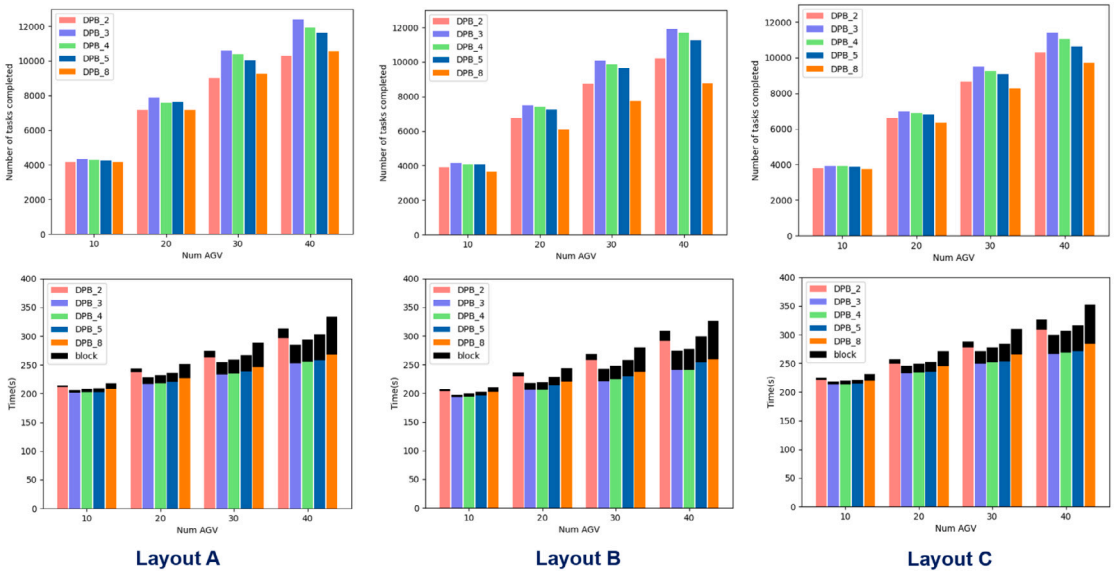


Fig. 8. Number of completed tasks(up) and average travel time(down) varying with m.

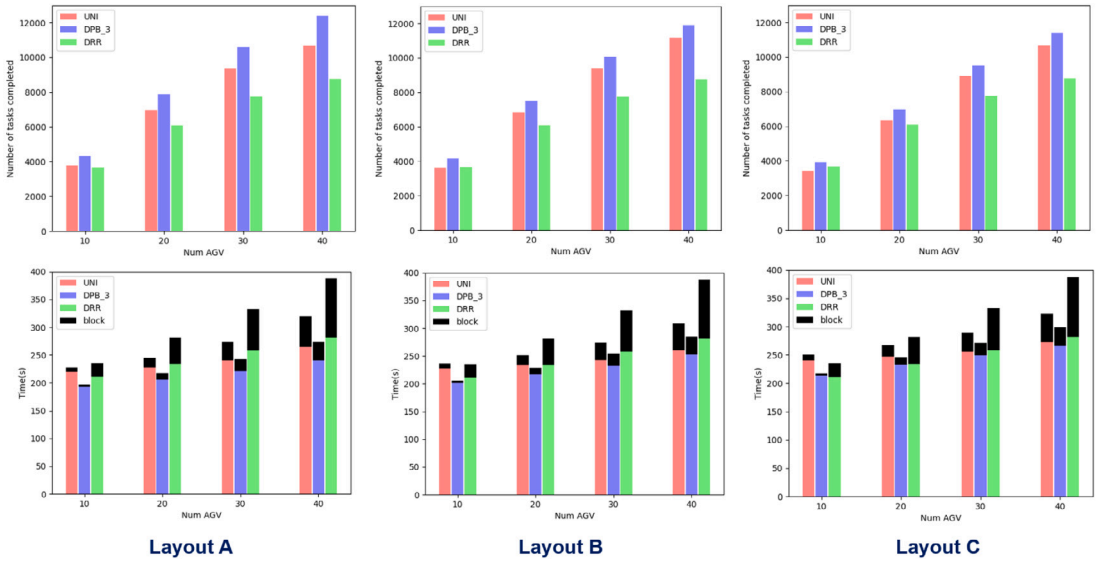


Fig. 9. Number of completed tasks(up) and average travel time(down) of benchmarks.

AGVs. As the density of AGVs increases, narrow passages between shelves are occupied for longer periods of time by AGVs, resulting in a relatively lower degree of freedom in path selection. Proposed algorithm efficiently explores better paths according to the situation, resulting in superior performance compared to unidirectional layouts. This has a significant advantage in that it can make efficient decisions using partial information, even when the size of the layout and the number of AGVs increase.

This research is ongoing and requires further exploration. The DPB algorithm is not constrained to grid layouts and is adaptable to various types of layouts, as it takes into account the driving mechanisms of AGVs. However, using the outdegree as the sole criterion may not be adequate for determining the reserved path of AGVs. General layouts feature edges of varying lengths, and the outdegree of a node is not as limited as in grid-based models. Therefore, the conditions for extending a reserved path must be adaptively determined, taking into account the topology of the general layout.

5. Conclusion

In this paper, we introduce a path-planning framework for bidirectional AGV systems that is directly applicable to actual systems. While numerous studies have addressed the AGV path planning problem, most overlook the loop deadlock phenomenon caused by the physical characteristics of AGVs. Those studies that do consider loop deadlock often impose excessive restrictions on AGV movement, leading to decreased system efficiency. We propose a practical, deadlock-free path planning algorithm compatible with various path planning methods. The algorithm's performance was analyzed in simulation environments that incorporate the physical aspects of real hardware, showing that our proposed algorithms surpass conventional approaches in addressing loop deadlock. For future work, we aim to develop a path planning algorithm that accounts for the operating principles of the hardware and the AGV's driving method. Additionally, we plan to generalize the algorithm to effectively prevent deadlocks in various layouts containing multiple types of AGVs.

Table 3
Experiment results.

|A|	Algorithm	Layout A		Layout B		Layout C	
		Number of tasks	Average block time (s)	Number of tasks	Average block time (s)	Number of tasks	Average block time (s)
10	DPB-2	4140	2.671955	3900.7	3.201368	3821.7	3.575513
	DPB-3	4332.7	4.415568	4152	5.106156	3923.3	5.212478
	DPB-4	4296	4.930286	4104.7	5.686877	3895.3	6.016213
	DPB-5	4261.3	5.637218	4088.3	6.355067	3874	6.762368
	DPB-8	4142.3	8.114273	3939.7	9.387925	3718	10.28584
	UNI	3850.7	20.44491	3650.3	23.23844	3447	26.8287
	DRR	3766	7.323545	3634	9.901354	3425.3	9.927029
20	DPB-2	7166	6.452143	6776.3	6.883352	6626	7.549817
	DPB-3	7875.3	11.36026	7490	12.35307	6975.7	13.12718
	DPB-4	7559	12.01085	7385.7	13.81743	6889.7	14.68522
	DPB-5	7628	15.15921	7260.7	15.99893	6801.3	16.65192
	DPB-8	7181.7	23.26068	6833	24.04356	6342.3	25.5407
	UNI	6471	43.39068	6090.7	47.57321	5601.7	55.16241
	DRR	6988.3	18.098	6813.7	16.33509	6345	20.70772
30	DPB-2	9028	10.85229	8736	10.94753	8667	11.7492
	DPB-3	10 606	21.00006	10 083	21.20582	9496.3	21.74337
	DPB-4	10 368	24.29682	9872.3	24.33496	9245.7	25.09258
	DPB-5	10 020	27.4258	9645.7	28.71174	9084	29.87345
	DPB-8	9246	41.61513	8863.7	41.76086	8280.3	45.36684
	UNI	8259.3	69.83084	7750.7	74.41578	7058.7	83.73674
	DRR	9380.3	34.36199	9395.7	23.69088	8902	33.34057
40	DPB-2	10 284	16.52952	10 194	16.28342	10 287	17.10104
	DPB-3	12 376	32.9064	11 893	32.3833	11 389	32.03732
	DPB-4	11 945	36.74972	11 678	38.54504	11 043	37.83483
	DPB-5	11 599	44.99463	11 267	45.54557	10 642	45.20106
	DPB-8	10 576	67.62353	10 183	65.68339	9697.7	67.86735
	UNI	9297	98.78794	8751.5	106.1836	8193.5	117.7325
	DRR	10 663	55.13338	11 144	32.28125	10 655	49.50394

CRediT authorship contribution statement

Chang Hyun Chung: Writing – review & editing, Writing – original draft, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Young Jae Jang:** Writing – review & editing, Project administration, Methodology, Investigation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

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

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.asoc.2024.111725.

References

[1] P.R. Wurman, R. D'Andrea, M. Mountz, Coordinating hundreds of cooperative, autonomous vehicles in warehouses, AI Mag. 29 (1) (2008) 9.
[2] P. Surynek, An optimization variant of multi-robot path planning is intractable, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 24, (1) 2010, pp. 1261–1263.
[3] J. Yu, S. LaValle, Structure and intractability of optimal multi-robot path planning on graphs, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 27, (1) 2013, pp. 1443–1449.
[4] J. Yu, S.M. LaValle, Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics, IEEE Trans. Robot. 32 (5) (2016) 1163–1177.
[5] E. Lam, P. Le Bodic, D. Harabor, P.J. Stuckey, Branch-and-cut-and-price for multi-agent path finding, Comput. Oper. Res. 144 (2022) 105809.
[6] A. Andreychuk, K. Yakovlev, P. Surynek, D. Atzmon, R. Stern, Multi-agent pathfinding with continuous time, Artificial Intelligence 305 (2022) 103662.
[7] G. Sharon, R. Stern, A. Felner, N.R. Sturtevant, Conflict-based search for optimal multi-agent pathfinding, Artificial Intelligence 219 (2015) 40–66.
[8] A. Felner, J. Li, E. Boyarski, H. Ma, L. Cohen, T.S. Kumar, S. Koenig, Adding heuristics to conflict-based search for multi-agent path finding, in: Proceedings of the International Conference on Automated Planning and Scheduling, vol. 28, 2018, pp. 83–87.
[9] J. Li, A. Felner, E. Boyarski, H. Ma, S. Koenig, Improved heuristics for multi-agent path finding with conflict-based search., in: IJCAI Vol. 2019, 2019, pp. 442–449.
[10] D. Silver, Cooperative pathfinding, in: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, vol. 1, (1) 2005, pp. 117–122.
[11] M. Phillips, M. Likhachev, Sipp: Safe interval path planning for dynamic environments, in: 2011 IEEE International Conference on Robotics and Automation, IEEE, 2011, pp. 5628–5635.
[12] H. Ma, D. Harabor, P.J. Stuckey, J. Li, S. Koenig, Searching with consistent prioritization for multi-agent path finding, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, (01) 2019, pp. 7643–7650.
[13] G. Sartoretti, J. Kerr, Y. Shi, G. Wagner, T.S. Kumar, S. Koenig, H. Choset, Primal: Pathfinding via reinforcement and imitation multi-agent learning, IEEE Robot. Autom. Lett. 4 (3) (2019) 2378–2385.
[14] Q. Li, W. Lin, Z. Liu, A. Prorok, Message-aware graph attention networks for large-scale multi-robot path planning, IEEE Robot. Autom. Lett. 6 (3) (2021) 5533–5540.
[15] M. Damani, Z. Luo, E. Wenzel, G. Sartoretti, Primal _2: Pathfinding via reinforcement and imitation multi-agent learning-lifelong, IEEE Robot. Autom. Lett. 6 (2) (2021) 2666–2673.
[16] H. Ma, W. Hönig, T.S. Kumar, N. Ayanian, S. Koenig, Lifelong path planning with kinematic constraints for multi-agent pickup and delivery, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, (01) 2019, pp. 7651–7658.
[17] J. Švancara, M. Vlk, R. Stern, D. Atzmon, R. Barták, Online multi-agent pathfinding, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, (01) 2019, pp. 7732–7739.
[18] J. Li, A. Tinka, S. Kiesel, J.W. Durham, T.S. Kumar, S. Koenig, Lifelong multi-agent path finding in large-scale warehouses, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, (13) 2021, pp. 11272–11281.
[19] Y. Zhao, X. Liu, G. Wang, S. Wu, S. Han, Dynamic resource reservation based collision and deadlock prevention for multi-AGVs, IEEE Access 8 (2020) 82120–82130.