2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

MARPF: Multi-Agent and Multi-Rack Path Finding

Hiroya Makino[1,*], Yoshihiro Ohama[1], and Seigo Ito[1]

***Abstract*— In environments where many automated guided vehicles (AGVs) operate, planning efficient, collision-free paths is essential. Related research has mainly focused on environments with pre-defined passages, resulting in space inefficiency. We attempt to relax this assumption. In this study, we define multi-agent and multi-rack path finding (MARPF) as the problem of planning paths for AGVs to convey target racks to their designated locations in environments without passages. In such environments, an AGV without a rack can pass under racks, whereas one with a rack cannot pass under racks to avoid collisions. MARPF entails conveying the target racks without collisions, while the obstacle racks are relocated to prevent any interference with the target racks. We formulated MARPF as an integer linear programming problem in a network flow. To distinguish situations in which an AGV is or is not loading a rack, the proposed method introduces two virtual layers into the network. We optimized the AGVs' movements to move obstacle racks and convey the target racks. The formulation and applicability of the algorithm were validated through numerical experiments. The results indicated that the proposed algorithm addressed issues in environments with dense racks.**

I. Introduction

Over the past few decades, introducing automated guided vehicles (AGVs) in warehouses and factories has accelerated efficiency. Numerous efforts have been devoted to developing multi-agent path finding (MAPF) for efficient transportation via AGVs [1]. This problem has been applied in many fields, such as automatic warehouses [2], [3], airport taxiway control [4], and automated parking [5].

For transportation in warehouses and factories, AGVs often navigate beneath rack-type carts and convey entire racks with their contents (hereinafter referred to as rack) to a designated location. Prior research on MAPF has mainly considered navigating these racks through areas with passages (Fig. 1(a)). However, this layout leads to the inefficient use of space. In settings where sufficient space cannot be provided, it is crucial to utilize the available area more effectively. Traditional MAPF algorithms struggle to optimize navigation racks efficiently in these dense environments (Fig. 1(b)). The difficulty lies in optimally relocating the obstacle racks.

This study defines multi-agent and multi-rack path finding (MARPF), which focuses on planning paths for AGVs to convey target racks to their designated locations in a grid-like environment without passages. The racks cannot move and, thus, should be conveyed by AGVs. AGVs without racks can pass under the racks; however, to avoid collisions, those with racks cannot. MARPF involves conveying the target racks to their designated destinations, while the obstacle racks are situated freely. In an environment in which racks are densely located, the obstacle racks can be moved to avoid interference with the target racks.

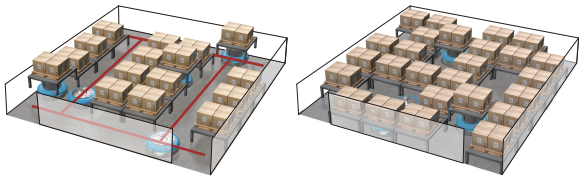

Fig. 1: Environments of MAPF and MARPF. MAPF needs passages, whereas MARPF does not.

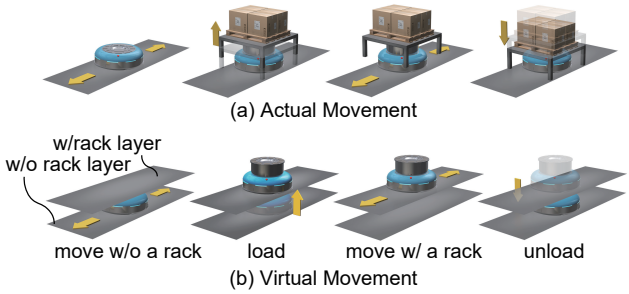

Fig. 2: Two layers to express AGVs' movements. The movements between these layers correspond to whether they are conveying a rack or not. When an AGV loads a rack, it virtually transitions from the "w/o rack" layer to the "w/rack" layer; unloading a rack reverses this transition.

We formulated MARPF as an integer linear programming (ILP) problem in a network flow, which includes the AGV and rack networks. In the AGV network, the proposed method distinguishes whether an AGV is conveying a rack using two virtual layers (Fig. 2). The rack network represents the movements of the racks, which are separated from the AGVs. By synchronizing the AGV network with the rack network, the proposed method enables moving obstacle racks and conveying target racks while avoiding collisions. We aimed to solve the problem with various movement constraints and minimize the makespan (i.e., the latest completion time).

A. Contribution

Original multi-agent pickup and delivery (MAPD) focuses only on conveying target racks. Other studies have considered switching the positions of racks [6], [7]. The difference between MARPF and rearrangement-considering MAPD is as follows:

[1] H. Makino, Y. Ohama, and S. Ito are with Toyota Central R&D Labs., Inc., 41-1, Yokomichi, Nagakute, Aichi, Japan.
[*] Corresponding author. hirom@mosk.tytlabs.co.jp

- **MARPF**: Only the target racks are assigned goals; the obstacle racks are situated freely.
- **Rearrangement-considering MAPD**: All racks are assigned goals.

It could be posited that rearrangement-considering MAPD can potentially address MARPF by assigning the same goal positions as the initial positions for obstacle racks. However, we estimate two typical situations where this assumption may not apply: (1) Environments with highly dense racks, where some situations cannot be solved by rearrangement-considering MAPD (Fig. 3). Sufficient conditions for solvability are discussed in a sliding tile puzzle [8]; and (2) Environments with dense racks and a limited number of AGVs, where relocating racks to their initial positions would require a significant number of time steps. The optimization problem of where to relocate the obstacle racks is challenging, and MARPF constitutes a new problem.

The main contributions of this study are as follows:

- We define a new problem (MARPF) of planning paths for AGVs to convey target racks to their designated locations in dense environments.
- We propose a method for solving MARPF, formulated as an ILP problem in a network flow.
- Solving the full ILP problem would be computationally intractable; therefore, we propose a hybrid approach combined with cooperative A* (CA*) to allow feasible computation times.

B. Related Work

1) MAPF: The MAPF problem relates to finding the optimal paths for multiple agents without collisions, and numerous methods have been proposed [1]. As complete and optimal solvers, there are conflict-based search (CBS) [9], improved CBS [10], and enhanced CBS [11]. Prioritized planning, such as CA* [12] and multi-label A* [13], has a short runtime but is suboptimal. MAPD [14] is a lifelong variant of MAPF. Specifically, in MAPD, each agent is constantly assigned new tasks with new goal locations, whereas in MAPF, each agent has only one task.

2) Rearrangement-considering MAPD: In Double-Deck MAPD (DD-MAPD) [6], agents are tasked to move racks to their assigned delivery locations, thereby changing the overall arrangement of the racks. The algorithm for DD-MAPD solves a DD-MAPD instance with M agents and N racks by first decomposing it into an N-agent MAPF instance, and subsequently into a M-agent MAPD instance with task dependencies. In Multi-Agent Transportation (MAT) [7], all racks are assigned delivery locations without fixed aisles. An algorithm is provided for solving MAT by reducing it to a series of satisfiability problems. In DD-MAPD and MAT, all racks must be assigned goals. However, they do not consider where and how obstacle racks are relocated, which is necessary for MARPF.

II. Problem Definition

In this section, we define the MARPF problem. As noted in Section I, MARPF aims to plan paths for AGVs to convey target racks in an environment without passages. Table I lists the notation used in the following sections.

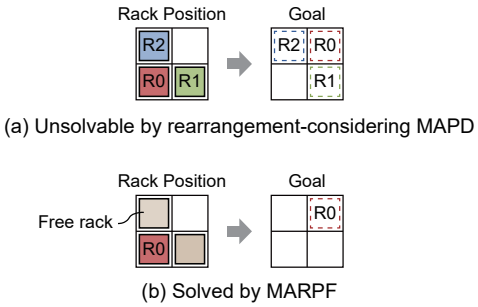

Fig. 3: Unsolvable example by MAPD.

An MARPF instance comprises M AGVs, N racks, and an undirected grid graph, $G = (V, E)$.

$$V = \{v_{x,y} \mid x \in \{0, \ldots, size_x - 1\}, y \in \{0, \ldots, size_y - 1\}\},$$
$$E = \{(v_{x,y}, v_{x',y'}) \mid v_{x,y}, v_{x',y'} \in V, |x - x'| + |y - y'| = 1\},$$

where $v_{x,y} \in V$ is the vertex with a column index x and row index y in the grid. Diagonal edges are not considered to avoid collisions between AGVs and the legs of racks. $loc^t(a_i), loc^t(r_i) \in V$ denote the vertices of AGV a_i and rack r_i at timestep t, respectively. $cnv^t(a_i) \in \{0, 1\}$ refers to the rack-conveying state of AGV a_i at timestep t, where 1 indicates that the AGV is conveying a rack, and 0 indicates otherwise.

Time is assumed to be discretize. At each time step t, an AGV executes one of the following actions:

1) Remain at the current location.
   $loc^t(a_i) = loc^{t+1}(a_i)$.
2) Move to the next location.
   $\left(loc^t(a_i), loc^{t+1}(a_i)\right) \in E$.
3) Load a rack.
   $cnv^t(a_i) = 0$, $cnv^{t+1}(a_i) = 1$, $loc^t(a_i) = loc^{t+1}(a_i)$.
4) Unload a rack.
   $cnv^t(a_i) = 1$, $cnv^{t+1}(a_i) = 0$, $loc^t(a_i) = loc^{t+1}(a_i)$.

A rack also executes one of the following actions: remain $loc^t(r_i) = loc^{t+1}(r_i)$ or move $\left(loc^t(r_i), loc^{t+1}(r_i)\right) \in E$; however, its movement depends on AGVs because the rack does not move by itself.

There should be no collisions between agents or racks. We define three types of collision for agents (Fig. 4): The first is the vertex conflict [1], where two agents cannot be in the same location at the same timestep. The second is the swapping conflict [1], where two agents cannot move along the same edge in opposite directions at the same timestep. The third is the corner conflict. When an agent moves to the other agent's position and their adjacent agent's movements are in a vertical relationship, their corners collide, as shown in Fig. 4(c).

Formally, for all agents $a_i, a_j (i \neq j)$ and all timesteps t,

TABLE I: Notations.

Symbols	Description
Section II and after	
a_i	AGV i
r_i	Rack i
$x \in \{0,\dots,size_x-1\}$	Column index ($size_x$ is the grid width)
$y \in \{0,\dots,size_y-1\}$	Row index ($size_y$ is the grid height)
$v_{x,y}$	Vertex
V	Set of vertices
E	Set of edges
$G=(V,E)$	Connected undirected graph
t	Timestep
$loc^t(a_i)$	Location vertex of a_i at timestep t
$cnv^t(a_i)$	Rack-conveying state of a_i at timestep t
Section III and after	
$G^+_{ag}=(V^+_{ag},E^+_{ag})$	Time-expanded network representing all AGVs' movements
$G^+_{ar}=(V^+_{ar},E^+_{ar})$	Time-expanded network representing all racks' movements
$G^+_{tr}=(V^+_{tr},E^+_{tr})$	Time-expanded network representing the target rack's movements
$\mathcal{T}$	Set of timesteps in time-expanded networks
$c(\cdot)$	Cost function of flow
l	Layer (w/rack or w/o rack)
$S_{ag},S_{ar},S_{tr} \subseteq V$	Start location vertices of all AGVs, all racks, and the target racks, respectively
$T_{tr} \subseteq V$	Goal location vertex of the target rack
$\alpha^t_{vi,vo,li,lo},\beta^t_{vi,vo},\gamma^t_{vi,vo}$	Flow on $E^+_{ag},E^+_{ar},E^+_{tr}$, respectively

the following equations must hold to avoid collisions:

$$loc^t(a_i) \neq loc^t(a_j), \tag{1}$$

$$\begin{aligned} &loc^t(a_i) = loc^{t+1}(a_j) \\ &\Rightarrow \left(loc^{t+1}(a_i)_x - loc^t(a_i)_x\right) = \left(loc^{t+1}(a_j)_x - loc^t(a_j)_x\right) \\ &\quad \wedge \left(loc^{t+1}(a_i)_y - loc^t(a_i)_y\right) = \left(loc^{t+1}(a_j)_y - loc^t(a_j)_y\right), \end{aligned} \tag{2}$$

where $loc^t(a_i)_x$ and $loc^t(a_i)_y$ denote the column and row indices of $loc^t(a_i)$, respectively. (1) represents the vertex conflict. (2) represents both the swapping and corner conflicts. These three types of collision are also applied to racks.

The MARPF problem involves computing collision-free paths for AGVs and minimizing the timesteps required to convey the target racks to their designated locations. Non-target racks act as obstacles and are positioned freely.

Fig. 5 exemplifies moving a rack to avoid interference with the movement of the target rack. The path the AGV must follow to convey the target rack to its designated location is blocked by an obstacle rack (Fig. 5(a)). First, the obstacle rack is removed (Fig. 5(b)), and the target rack is then conveyed to the designated location (Fig. 5(c)).

III. Proposed Method

In this section, we formulate MARPF as a minimum-cost flow problem in a network flow, similar to the approaches for MAPF [5], [15]. Two types of synchronized networks are established, one for AGVs and the other for racks. The network for AGVs distinguishes whether or not an AGV is conveying a rack using two virtual layers to represent conveying a rack.

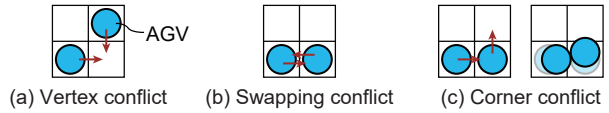

Fig. 4: Type of conflicts.

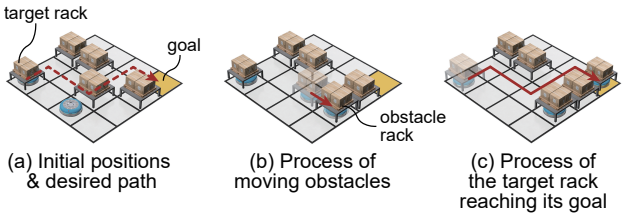

Fig. 5: Problem example.

A. Definition of Networks

When AGVs are not conveying a rack, their obstacles are the other AGVs; however, when they are conveying a rack, other racks also become obstacles. Therefore, whether an AGV is conveying a rack onto a time-expanded network must be determined. The AGVs' movements are classified into the following four actions in terms of the rack-conveying state (Fig. 2(a)):

1) Move (or remain at the current location) without a rack.
2) Load a rack.
3) Move (or remain at the current location) with a rack.
4) Unload a rack.

Our proposed method divides a moving plane into two virtual layers to represent two situations: conveying or not conveying a rack (Fig. 2(b)).

The rack-loading action corresponds to movement from the w/o to the w/rack layers, and vice versa.

Then, all vertices are expanded along the time-axis, and under specific constraints, directed edges are added. The edges indicate AGV motion in the grid. The motions are distinguished by the indices (x,y) of the source and sink vertices and the virtual layers, where the motion is with $(l=1)$ and without $(l=0)$ a rack. Fig. 6 (upper) shows an example of a time-expanded network representing all AGVs' movements. For simplicity, Figs. 2 and 6 show the moving plane in one dimension; however, it is two-dimensional.

For the racks' movements, the networks should be separated from the AGVs. Distinguishing the layers in a time-expanded network is not necessary (Fig. 6 (lower)). The flows expressing the movements of racks are synchronized with the corresponding flows of AGVs.

We define the time-expanded network representing all AGVs, all racks, and the target rack's movements as directed graphs $G^+_{ag}=(V^+_{ag},E^+_{ag})$, $G^+_{ar}=(V^+_{ar},E^+_{ar})$, and $G^+_{tr}=(V^+_{tr},E^+_{tr})$, respectively. G^+_{tr} has the same structure as G^+_{ar}; however, the target rack is distinguished from the other racks.

The goal location of the target rack is fixed on G^+_{tr}.

$$
\begin{aligned}
V^+_{ag} &= \{v^t_{x,y,l} \mid x \in \{0,\dots,size_x-1\}, y \in \{0,\dots,size_y-1\},\\
&\qquad l \in \{0,1\}, t \in \{0,\dots,size_t\}\},\\
E^+_{ag} &= \{\langle v^t_{x,y,l}, v^{t'}_{x',y',l'}\rangle \mid v^t_{x,y,l}, v^{t'}_{x',y',l'} \in V^+_{ag},\\
&\qquad |x-x'|+|y-y'| \le 1, t'-t=1\},\\
V^+_{ar}, V^+_{tr} &= \{v^t_{x,y} \mid x \in \{0,\dots,size_x-1\},\\
&\qquad y \in \{0,\dots,size_y-1\}, t \in \{0,\dots,size_t\}\},\\
E^+_{ar}, E^+_{tr} &= \{\langle v^t_{x,y}, v^{t'}_{x',y'}\rangle \mid v^t_{x,y}, v^{t'}_{x',y'} \in V^+_{ar},\\
&\qquad |x-x'|+|y-y'| \le 1, t'-t=1\},
\end{aligned}
$$

where $size_t$ denotes the maximum timesteps in the time-expanded network. The value of $size_t$ is set large enough for the target agent to reach the goal. Dependencies exist between G^+_{ar} and G^+_{ag} and between G^+_{tr} and G^+_{ar}, and their flows are synchronized.

B. Definition of Variables

The vertices on V^+_{ag} are distinguished by t, v, and l. Each edge consists of the source and sink vertices. $\alpha^t_{vi,vo,li,lo}$ represents the flow on the edge $\langle v^t_{vi_x,vi_y,li}, v^{t+1}_{vo_x,vo_y,lo}\rangle \in E^+_{ag}$. Here, $vi, vo \in V, li, lo \in \{0,1\}$ respectively denote the source vertex, sink vertex, source layer, and sink layer, and $t \in \mathcal{T} = \{0,\dots,size_t-1\}$ represents the source vertex's timestep. G^+_{ar}, G^+_{tr} do not contain multiple layers; therefore, the flows of racks are expressed more simply as $\beta^t_{vi,vo}, \gamma^t_{vi,vo}$, respectively. We define the variables and the cost function $c(\cdot)$ as follows:

$$\alpha^t_{vi,vo,li,lo}, \beta^t_{vi,vo}, \gamma^t_{vi,vo} \in \{0,1\}, \tag{3}$$

$$c(vi,vo,li,lo,t) = \begin{cases} 0 & \text{if } vi = vo, li = lo \\ t+1 & \text{otherwise} \end{cases}. \tag{4}$$

We define the initial location vertices of all AGVs, racks, and the target rack, respectively, as S_{ag}, S_{ar}, and S_{tr}, where $S_{tr} \subseteq S_{ar}$. In the network, we express the initial locations by fixing the flows at $t=0$ (respectively, (5), (6), and (7)). The goal location vertex of the target rack, T_{tr}, is defined by (8).

$$\begin{cases} \alpha^0_{vi,vo,li,lo} = 1 & \text{if } vi = vo \in S_{ag}, li = lo = 0 \\ \alpha^0_{vi,vo,li,lo} = 0 & \text{otherwise} \end{cases} \tag{5}$$

$$\begin{cases} \beta^0_{vi,vo} = 1 & \text{if } vi = vo \in S_{ar} \\ \beta^0_{vi,vo} = 0 & \text{otherwise} \end{cases} \tag{6}$$

$$\begin{cases} \gamma^0_{vi,vo} = 1 & \text{if } vi = vo \in S_{tr} \\ \gamma^0_{vi,vo} = 0 & \text{otherwise} \end{cases} \tag{7}$$

$$\begin{cases} \gamma^{size_t-1}_{vi,vo} = 1 & \text{if } vi = vo \in T_{tr} \\ \gamma^{size_t-1}_{vi,vo} = 0 & \text{otherwise} \end{cases} \tag{8}$$

C. Minimum Cost Flow Problem

The aim is to convey the target rack from the starting position to the goal position as quickly as possible. The set of timesteps $\mathcal{T} \setminus \{size_t - 1\}$ refers to $\mathcal{T}^-$.

$$\min \sum c(vi,vo,li,lo,t)\alpha^t_{vi,vo,li,lo} \tag{9}$$

s.t.

$$\sum_{vi,li} \alpha^t_{vi,v,li,l} = \sum_{vo,lo} \alpha^{t+1}_{v,vo,l,lo} \qquad \forall v \in V, l \in \{0,1\}, t \in \mathcal{T}^- \tag{10}$$

$$\sum_{vi} \beta^t_{vi,v} = \sum_{vo} \beta^{t+1}_{v,vo} \qquad \forall v \in V, t \in \mathcal{T}^- \tag{11}$$

$$\sum_{vi} \gamma^t_{vi,v} = \sum_{vo} \gamma^{t+1}_{v,vo} \qquad \forall v \in V, t \in \mathcal{T}^- \tag{12}$$

$$\beta^t_{vi,vo} = \alpha^t_{vi,vo,1,1} \qquad \forall (vi,vo) \in E, t \in \mathcal{T} \tag{13}$$

$$\gamma^t_{vi,vo} \le \beta^t_{vi,vo} \qquad \forall vi, vo \in V, t \in \mathcal{T} \tag{14}$$

$$\alpha^t_{v,v,0,1} \le \sum_{vi} \beta^t_{vi,v} \qquad \forall v \in V, t \in \mathcal{T} \tag{15}$$

$$\sum_{vi,li,lo} \alpha^t_{vi,v,li,lo} \le 1 \qquad \forall v \in V, t \in \mathcal{T} \tag{16}$$

$$\sum_{vi} \beta^t_{vi,v} \le 1 \qquad \forall v \in V, t \in \mathcal{T} \tag{17}$$

$$\sum_{li,lo} \alpha^t_{vi,vo,li,lo} + \sum_{v,li,lo} \alpha^t_{vo,v,li,lo} \le \sum_{\hat{v}_{2x'-x,2y'-y},li,lo} \alpha^t_{vo,\hat{v},li,lo} + 1 \qquad \forall (vi_{x,y}, vo_{x',y'}) \in E, t \in \mathcal{T} \tag{18}$$

The objective function (9) defines the cost function, which implies that the timestep t is directly proportional to the movement cost. Therefore, this problem calculates the flows with the minimum number of steps to convey the target rack from the initial to the goal position.

Constraints (10)–(12) are required to satisfy the flow conservation constraints at the vertices. Racks are conveyed by AGVs, and (13) indicates that the flows of the racks' movements on G^+_{ar} are equal to the corresponding flow on G^+_{ag}. Constraint (13) indicates that G^+_{ar} depends on G^+_{ag} and (14) indicates that G^+_{tr} depends on G^+_{ar}. Considering (7), (12), and (14), only flows related to the movement of the target rack on G^+_{ar} are reflected in G^+_{tr}. Constraint (15) indicates that a rack must exist where an AGV loads a rack ($\alpha^t_{v,v,0,1} = 1$). Eq. $\sum_{vi} \beta^t_{vi,v} = 1$ indicates that one of the flows to vertex v is 1; that is, a rack is placed at vertex v. Constraints (16) and (17) prohibit AGVs and racks from coexisting at the same vertices, respectively. Constraint (18) is required to prevent the swapping conflict and the corner conflict between agents.

The above formulation is for one target rack. However, the formulation can be extended to multiple target racks. The network representing the movement of the i-th target rack is referred to as $G^{+,i}_{tr} = (V^{+,i}_{tr}, E^{+,i}_{tr})$. The corresponding flow is $\gamma^{t,i}_{vi,vo}$ and the constraints are the same as $\gamma^t_{vi,vo}$.

D. Applications for Real-Time Solving

The proposed method complicates the network with longer path lengths and the computational cost increases exponentially. We believe that appropriately dividing the path reduces the computational costs. Hence, we propose an acceleration method combined with CA*, called *CA*-ILP*. This method comprises global and local searches. The global search finds

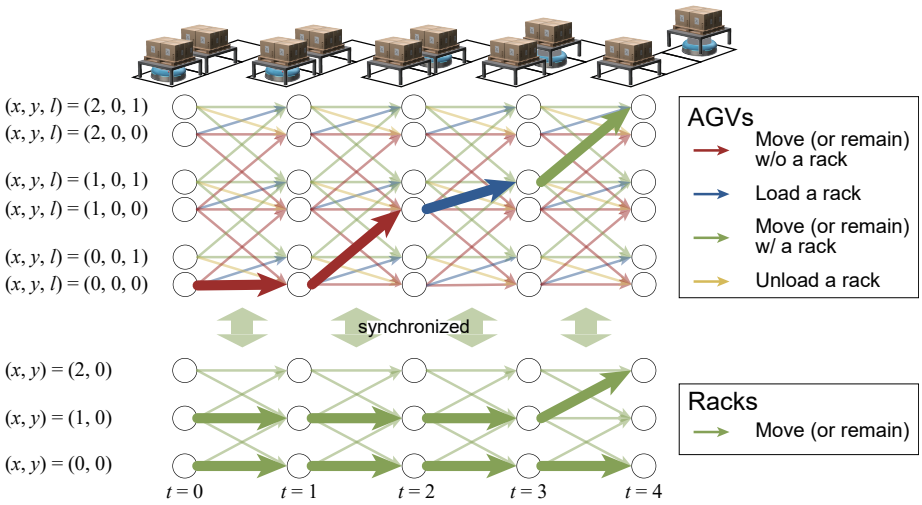

Fig. 6: Time-expanded networks for AGVs (upper) and racks (lower). Loading or unloading a rack (blue and yellow arrows, respectively) only changes the loading status l, whereas moving (red and green arrows) changes the position x or y but not l. The two networks are synchronized, and the movement of racks depends on the movement of AGVs. The movement of the AGV and racks at the top of the figure corresponds to the thick arrows.

waypoints to divide the path, and the local search explores the paths between each waypoint.

CA* is an extension of the A* algorithm, tailored for multi-agent scenarios. It allows multiple agents to plan their paths sequentially, avoiding collisions through a reservation table that records and manages the paths of all agents. This ensures that each agent's path is optimized while preventing overlaps in their paths.

First, path finding is performed using CA* for the global search. In this step, the racks are assumed to move by themselves, and collisions between racks are allowed. However, from the viewpoint of timesteps, removing obstacle racks should be avoided. Therefore, moving to a location with a rack is defined as incurring a $\kappa > 1$ cost in CA*. Moving to a location with no racks incurs a cost of $+1$. Algorithm 1 presents the corresponding pseudo-code.

We define span length τ between two waypoints before procedure. A large span length brings the solution closer to the global optimum; however, the computational cost increases. In the pseudo code, the global search executes CA* [line 1]. It chooses the waypoints according to the global paths [lines 2–7]. The goal locations are also included in the waypoints [line 8].

Second, the local search repeatedly solves the local path-finding problem. Algorithm 2 presents the pseudo-code. The local path-finding problem, which involves conveying multiple racks to their waypoints, is solved using ILP [line 3]. The target racks do not always arrive at their waypoint simultaneously. When one of the racks arrives at its waypoint, a local search is performed again [lines 4–8].

IV. Experiments

In this section, we describe two experiments: (1) a comparative evaluation of the proposed method against existing methods; and (2) an evaluation of the effectiveness of *CA*-ILP*. For the experimental setup, the time-expanded networks were represented by NetworkX[1] and the optimization problem was defined by PuLP[2]. We used the GUROBI solver[3]. All the experiments were run on a system comprising Ubuntu 22.04, Intel Core i9-12900K, and 128 GiB of RAM.

We performed all the experiments in 6×4 grids. Although this size is too small for automated warehouses, small-sized environments, such as inter-process transportation, are utilized in specific areas of the factories.

A. Experiment 1: Comparative Evaluation of the Proposed Method against Existing Methods

We compared our initially proposed method, an ILP-based method (independent of CA*), against widely used MAPF solvers, CA* and CBS. In MARPF, the other racks become obstacles only when the AGV conveys a rack. We used CA* and CBS for comparison, which accounts for these conditions. In the first step, the AGVs moved to the initial locations of the target racks without collisions between AGVs. In the second step, the AGVs moved to the goal locations of the target racks without collisions between AGVs and between racks.

There were initially two AGVs and six racks, including targets. The racks were then added sequentially. Fig. 7(a) illustrates this environment. Two obstacle racks could block the paths. For example, two racks were placed in the lower-left region (Fig. 7(b)). We performed 30 experiments at different locations to add the racks. The solver's time limit was 120 s, and $size_t$ was set to 40.

[1] https://networkx.org/
[2] https://coin-or.github.io/pulp/
[3] https://www.gurobi.com/solutions/gurobi-optimizer/

Algorithm 1 Global Search

Input: Graph G, racks (i-th rack moves from n_0^i to n_g^i)
Parameter: Span length τ between two waypoints
Output: Sequences of waypoints *waypoints*

Find the global paths of the target racks using CA*: $path[i] = (n_0^i, n_1^i, \ldots, n_g^i)$, which is a vertex sequence.
for all i **do**
    $waypoints[i] = []$
    $index = 0$
    while $index < |path[i]| - 1$ **do**
        $waypoints[i] \leftarrow$ add $path[i][current\_index]$
        $index = index + \tau$
    $waypoints[i] \leftarrow$ add n_g^i
Return *waypoints*

Algorithm 2 Local Search

Input: Graph G, agents, racks, waypoints *waypoints*
Output: Paths of all agents and racks *paths*

$paths = []$
while not all racks have arrived at their waypoints **do**
    Use ILP to calculate paths *temp_paths* for all target racks from their current position to their respective waypoints
    min_t is the minimum length of *temp_paths*
    for all i in all agents and racks **do**
        Update the $paths[i]$ with the $temp\_paths[i][:min\_t+1]$
        Update the current position with $temp\_paths[i][min\_t]$
    Check if each rack has reached its waypoint
Return *paths*

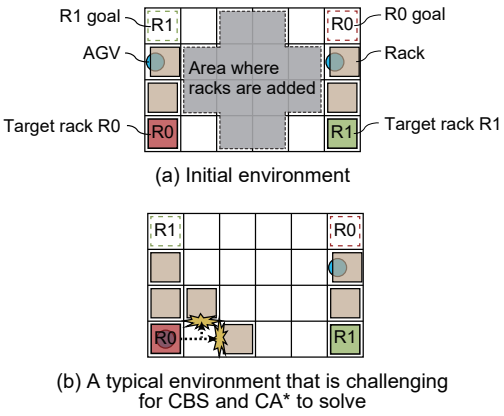

Fig. 7: Environment of Exp. 1.

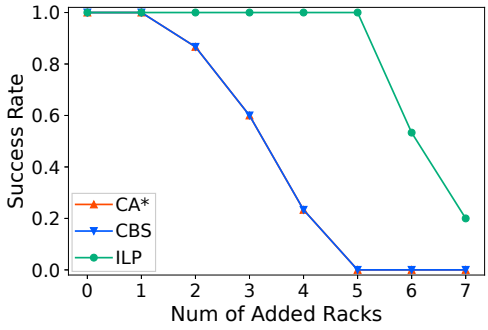

Fig. 8: Comparison of success rates on Exp. 1.

Fig. 8 shows a comparison of the success rates and average makespans of the successful tasks[4]. As Fig. 8 shows, with few added racks, all methods succeeded. However, with many added racks, CA* and CBS sometimes failed to find the path. CA* and CBS consider only the movements of the target racks and cannot remove obstacle racks. Therefore, they could not find the path when many racks were added and the paths were blocked. Our proposed method considered all rack movements, and as a result, it found the movement necessary for removing obstacle racks.

Although the proposed method is theoretically capable of solving these problems if there are one or more empty vertices, it failed in half of the tasks when adding six racks. When the problem was complex and the computational cost was high, the solver failed to find a path within the time limit. Therefore, we evaluated an acceleration method in the following experiment.

B. Experiment 2: Evaluation of the Effectiveness of CA*-ILP

Solving the path-finding problem using the proposed method was computationally expensive. We confirmed that *CA*-ILP* reduced the computational cost. We assigned a cost $\kappa = 3$ to move to a location occupied by a rack[5].

The evaluations were conducted in sparse (12 racks) and dense (18 racks) environments. Each environment contained eight AGVs. Fig. 9 is an example of a dense environment. We performed 30 experiments using the AGVs and non-target racks in different initial locations to compare the makespans of the tasks of conveying the two target racks to their goals. The comparison of the makespans is shown in Table II, with the differences in span lengths between waypoints (*CA*-ILP*) highlighted. Fig. 10 also shows a comparison of the makespans between the initially proposed method (*ILP-only*) and the acceleration method (*CA*-ILP*). We set the time limit of 120s for each local search conducted by *CA*-ILP*, and each $size_t$ was set to 24. *ILP-only* was computationally expensive; therefore, we set the time limit to three different values, 120s, 240s, and 1200s, and compared them. $size_t$ was set to 40. In the dense environment (Fig. 10(b))[6], *ILP-only* failed to find solutions in some experiments within the time limits of 120s and 240s.

First, we evaluated the difference in span between waypoints (*CA*-ILP*). According to Table II, Span 4 yielded the best result. Span 1, being excessively fine-grained and ignoring future rack locations, had a larger makespan. We adopted Spans 2 and 4 for the subsequent comparisons.

[4]In Fig. 8, there was no difference between CA* and CBS in such a simple environment.

[5]We set the cost κ to 3 because we assumed that the task involved three steps: loading, moving, and unloading the obstacle rack.

[6]The calculation time included the time required for network construction.

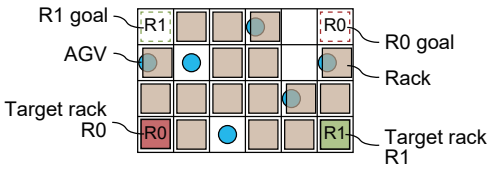

Fig. 9: Example of the environment in Exp. 2.

TABLE II: Comparison of CA*-ILP makespans on Exp. 2.

Span length τ between waypoints	1	2	4
12 racks	21.9	18.1	<u>14.8</u>
20 racks	39.9	28.9	<u>23.9</u>

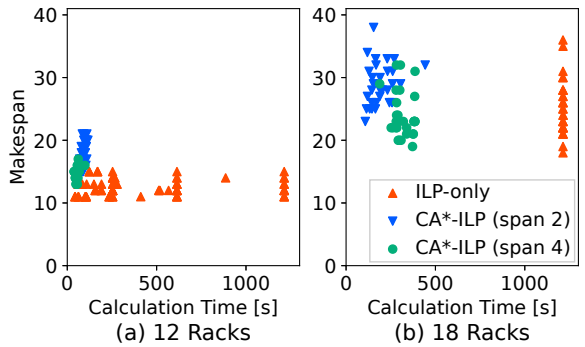

Fig. 10: Comparison of makespans on Exp. 2.

Second, we compared the initially proposed method (*ILP-only*) with the acceleration method (*CA*-ILP*). In the sparse environment (Fig. 10(a)), *CA*-ILP* exhibited a shorter total calculation time compared with *ILP-only*; however, its makespan was larger. *CA*-ILP* divided the paths into several parts, each easier to solve, thereby reducing the overall calculation cost compared to *ILP-only*. However, because the waypoints in *CA*-ILP* were not always optimal, its makespan exceeded that of the global optimal solution by *ILP-only*.

In the dense environment (Fig. 10(b)), the makespan of *CA*-ILP* was smaller than *ILP-only*, and the total calculation time of *CA*-ILP* was considerably shorter than *ILP-only*. In the dense environment, the computation cost was higher than that in the sparse environment, and *CA*-ILP* was more effective in calculating time. In all the experiments, *ILP-only* found a feasible solution within the time limit of 1200s; however, *ILP-only* did not always find a good feasible solution within the time limit. Consequently, the average makespan of *CA*-ILP* was smaller than *ILP-only*.

V. Conclusion

In this study, we defined the MARPF problem for planning the paths of target racks to their designated locations using AGVs in dense environments without passages. We proposed an ILP-based formulation for synchronized time-expanded networks by dividing the movements of AGVs and racks. The proposed method optimized the paths to move the obstacles and convey the target racks. Based on the complexity that increases with path length, we also presented an acceleration method combined with CA*. Our experiments confirmed that the acceleration method reduced computational cost. While we acknowledge the need for further research on more efficient algorithms, it is sufficient if the paths are calculated before the following product is completed in conveyance between production processes in factories. In these instances, our proposed algorithm can be practically applied.

Although we performed our experiments on a small grid, the problem setting of MARPF can be applied to large-scale warehouses. However, solving bigger problems increases the computational cost. In the future, we plan to investigate more efficient and faster algorithms.

Acknowledgments

We thank Kenji Ito, Tomoki Nishi, Keisuke Otaki, and Yasuhiro Yogo for their helpful discussions.

References

[1] R. Stern, N. Sturtevant, A. Felner, S. Koenig, H. Ma, T. Walker, J. Li, D. Atzmon, L. Cohen, T. K. Kumar, R. Barták, and E. Boyarski, "Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 10, 2019, pp. 151–158.

[2] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses." *AI Magazine*, vol. 29, no. 1, pp. 9–20, 2008.

[3] W. Honig, S. Kiesel, A. Tinka, J. W. Durham, and N. Ayanian, "Persistent and Robust Execution of MAPF Schedules in Warehouses," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1125–1131, 2019.

[4] J. Li, H. Zhang, M. Gong, Z. Liang, W. Liu, Z. Tong, L. Yi, R. Morris, C. Pasareanu, and S. Koenig, "Scheduling and Airport Taxiway Path Planning Under Uncertainty," in *Proceedings of the 2019 Aviation and Aeronautics Forum and Exposition*, 2019, pp. 1–8.

[5] A. Okoso, K. Otaki, S. Koide, and T. Nishi, "High Density Automated Valet Parking via Multi-agent Path Finding," in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, 2022, pp. 2146–2153.

[6] B. Li and H. Ma, "Double-Deck Multi-Agent Pickup and Delivery: Multi-Robot Rearrangement in Large-Scale Warehouses," *IEEE Robotics and Automation Letters*, pp. 1–8, 2023.

[7] P. Bachor, R.-D. Bergdoll, and B. Nebel, "The Multi-Agent Transportation Problem," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, 2023, pp. 11 525–11 532.

[8] W. W. Johnson and W. E. Story, "Notes on the "15" Puzzle," *American Journal of Mathematics*, vol. 2, no. 4, pp. 397–404, 1879.

[9] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2015.

[10] E. Boyarski, A. Felner, R. Stern, G. Sharon, O. Betzalel, D. Tolpin, and E. Shimony, "ICBS: The Improved Conflict-Based Search Algorithm for Multi-Agent Pathfinding," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 6, 2015, pp. 223–225.

[11] M. Barer, G. Sharon, R. Stern, and A. Felner, "Suboptimal Variants of the Conflict-Based Search Algorithm for the Multi-Agent Pathfinding Problem," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 5, 2014, pp. 19–27.

[12] D. Silver, "Cooperative Pathfinding," in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 1, 2005, pp. 117–122.

[13] F. Grenouilleau, W.-J. van Hoeve, and J. N. Hooker, "A Multi-Label A* Algorithm for Multi-Agent Pathfinding," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 29, 2021, pp. 181–185.

[14] H. Ma, J. Li, T. K. S. Kumar, and S. Koenig, "Lifelong Multi-Agent Path Finding for Online Pickup and Delivery Tasks," in *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems*, 2017, pp. 837–845.

[15] J. Yu and S. M. LaValle, "Planning Optimal Paths for Multiple Robots on Graphs," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 3612–3617.