

Investigation of Heuristics for PIBT Solving Continuous MAPF Problem in Narrow Warehouse

Toshihiro Matsui^a

Nagoya Institute of Technology, Gokiso-cho Showa-ku Nagoya Aichi 466-8555, Japan

Keywords: Multiagent Pathfinding Problem, Multiagent Pickup-and-Delivery Problem, Continuous, Lifelong, PIBT, Heuristics.

Abstract: We address the heuristics based on map structures in a solution method for continuous multiagent path finding problems particularly in the case of relatively narrow warehouse maps. The multiagent pathfinding problem has been studied as a fundamental problem in multiagent systems, and the lifelong/continuous multiagent pickup-and-delivery problem is a major extension of it that represents the tasks performed by robot carriers in automated warehouses. While basic methods of multiagent pathfinding are generally aimed at resolving collisions among agents using precisely computed/reserved paths or locally performed resolving algorithms, there might also be opportunities to employ information of maps and traffic for the heuristics of solution methods. As such an investigation, we focus on the case of multiagent pickup-and-delivery problems in narrow warehouse environments and the solution method called Priority Inheritance with Backtracking (PIBT), which is not based on the reservation of paths and is applicable to continuous problems within very narrow maps. We experimentally investigate the effect of map settings and additional heuristics based on the structures of maps.


1 INTRODUCTION

We address the heuristics based on map structures in a solution method for continuous multiagent path finding problems particularly in the case of relatively narrow warehouse maps. The multiagent pathfinding problem has been studied as a fundamental problem in multiagent systems where the (ideally) shortest paths, which also avoid collisions, are simultaneously found in a time-space graph. There are various applications of this problem including robot navigation, autonomous taxiing of airplanes and video games; here, we focus on the case of automated warehouses that deploy robot carriers. This class of problems is called the lifelong/continuous multiagent pickup-and-delivery problem (Ma et al., 2017), which is an extension of continuous multiagent pathfinding problems. In a typical system, each agent is repeatedly allocated to one of tasks generated on demand and moves from its current location to a delivery location via a pickup location.

The problem consists of task allocation and multiagent pathfinding problems that are continuously

solved. While the task allocation can be solved as a combinatorial optimization problems for static problems (Liu et al., 2019), greedy allocation methods are often employed for continuous problems with tasks generated on demand (Ma et al., 2017).

There are several solution methods for multiagent pathfinding problems. A greedy approach repeatedly finds and allocates each agent's collision-free path in a predetermined order among agents using the A* algorithm (Hart and Raphael, 1968; Hart and Raphael, 1972) on a time-space graph (Silver, 2005). A major exact solution method, called Conflict Based Search (Sharon et al., 2015), performs two layers of search, where collisions of agents are managed by a tree-search in the high-level layer while a pathfinding algorithm in the low-level layer is used to find a conflict-free path for each agent. However, this approach is computationally expensive, and several extensions of efficient or approximation methods have been proposed (Ma et al., 2019; Barer et al., 2014). Several extended problems and solution methods have also been proposed for more practical situations (Li et al., 2021; Yamauchi et al., 2022; Miyashita et al., 2023; Yakovlev and Andreychuk, 2017; Andreychuk et al., 2022; Andreychuk et al., 2021).

^a  <https://orcid.org/0000-0001-8557-8167>

In another type of greedy method, each agent determines its next move in each time step by locally solving collisions of agents' moves. Priority Inheritance with Backtracking (PIBT) (Okumura et al., 2022; Okumura et al., 2019) is classified into this type of algorithms based on push-and-rotate operations (De Wilde et al., 2014; Luna and Bekris, 2011), and it employs priorities of agents and a limited backtracking method in the process of resolving collisions. Although it requires the relatively restricted condition such that any vertex in a graph representing a floor plan must be contained in cycles, this is generally acceptable in the case of warehouses.

While fundamental methods generally resolve collisions among agents using precisely computed/reserved paths or locally performed resolving algorithms, there might also be opportunities to employ information of maps and traffic for the heuristics of solution methods. As such an investigation, we focus on the case of multiagent pickup-and-delivery problems in narrow warehouse environments and the original version of the solution method PIBT, which is not based on the reservation of paths and is applicable to continuous problems within very narrow maps. We experimentally investigate the effect of map settings and additional heuristics based on the structures of maps.

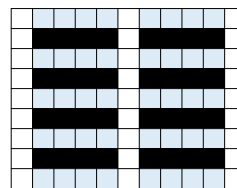
In well controlled automated warehouses satisfying the solvable conditions of this kind of lightweight MAPF algorithms, the low computational cost of the solvers can be promising to develop real-time applications. As effective add-ons for such solvers, the heuristic methods without exact searches/reservations on time-space is also important.

The rest of the paper is organized as follows. In the next section, we present the background of our study, including multiagent pathfinding/pickup-and-delivery problems, the solution method PIBT, and the aim of this study. Then we present our proposed approaches in Section 3. We first briefly consider an appropriate map settings in the case with sufficient space. Then, for the case of narrow maps, we employ additional heuristics based on maps structures and influential agents. We experimentally investigate these approaches in Section 4 and conclude the paper in Section 5.

2 PRELIMINARY

2.1 Multiagent Pathfinding Problem

The multiagent pathfinding (MAPF) problem is a fundamental problem for finding multiple agents' move-



White: passageway. Black: shelf (obstacle). Light blue: pickup/delivery location.

Figure 1: Example of grid-world warehouse map.

ment paths avoiding collisions in a time-space graph. A problem consists of graph $G = (V, E)$ representing a two-dimensional map such as a warehouse or maze, a set of agents \mathcal{A} , and a set of start-goal pairs of vertices to be allocated to the agents. Each agent has its origin and destination vertices and should move along its (ideally) shortest path avoiding other agents. There are two cases of colliding paths, called vertex and edge collisions. In a vertex collision, two agents stay at the same location at the same time, while in an edge collision, two agents move on the same edge at the same time from both ends of the edge. In a typical setting, a grid-like map containing obstacles and discrete time steps are employed. There are several classes of solution methods including the CA* algorithm (Silver, 2005), Conflict Based Search (Sharon et al., 2015), and variants of push-and-rotate approach (Okumura et al., 2022).

The continuous MAPF problem is an extended class of the MAPF problems where each agent updates its next goal after the agent moves to its current goal. A solution method for MAPF is repeatedly performed for the new paths.

2.2 Lifelong Multiagent Pickup-and-Delivery Problem

The lifelong multiagent pickup-and-delivery (MAPD) problem (Ma et al., 2017) is a class of continuous MAPF problems, where multiple pickup-and-delivery tasks in a warehouse or construction site are repeatedly allocated to agents. Figure 1 shows an example of warehouse map. The tasks can appear at arbitrary timings in a time span, and they are represented by a set of currently generated pickup-and-delivery tasks \mathcal{T} . Task $\tau_i \in \mathcal{T}$ has its pickup and delivery locations (s_i, g_i) , where $s_i, g_i \in V$. After task τ_i is allocated to an agent, the agent moves from its current location to a delivery location g_i through a pickup location s_i to complete the task. The problem can be decomposed into the task allocation and continuous MAPF problems. While (continuous) MAPF solvers can be applied to pathfinding, (partially) greedy approaches

```

1 UNDECIDED ←  $\mathcal{A}(t)$  // agents list
2 OCCUPIED ←  $\emptyset$  // vertices list
3 update priorities  $p_i(t)$  for all agents  $a_i$ 
4 while UNDECIDED  $\neq \emptyset$  do
5   let  $a$  be the agent with highest priority  $p(a)$  in
   UNDECIDED
6   PIBT( $a, \perp$ )
7 end while

9 function PIBT( $a_i, a_j$ )
10  UNDECIDED ← UNDECIDED  $\setminus \{a_i\}$ 
11   $C_i \leftarrow (\{v | (v_i(t), v) \in E\} \cup \{v_i(t)\})$ 
12     $\setminus (\{v_j(t)\} \cup \text{OCCUPIED})$ 
13  while  $C_i \neq \emptyset$  do
14     $v_i^* \leftarrow \arg \max_{v \in C_i} f_i(v)$  // most preferred move
15    OCCUPIED ← OCCUPIED  $\cup \{v_i^*\}$ 
16    if  $\exists a_k \in \text{UNDECIDED}$  s.t.  $v_i^* = v_k(t)$  then
17      if PIBT( $a_k, a_i$ ) is valid then
18         $v_i(t+1) \leftarrow v_i^*$ 
19        return valid
20      else
21         $C_i \leftarrow C_i \setminus \text{OCCUPIED}$ 
22      end if
23    else
24       $v_i(t+1) \leftarrow v_i^*$ 
25      return valid
26    end if
27  end while
28   $v_i(t+1) \leftarrow v_i(t)$ 
29  return invalid
30 end function

```

$v_i(t)$: location of agent a_i at time step t

Figure 2: PIBT (Okumura et al., 2022) at time step t .

are commonly employed to allocate tasks generated on demand.

A fundamental approach is based on the well-formed MAPD problems that take into account endpoint vertices, which can be pickup, delivery, or parking locations of agents (Čáp et al., 2015; Ma et al., 2017). In this approach, tasks can be greedily allocated under several rules, and their paths are also greedily reserved without deadlock situations. However, the paths basically cannot contain endpoints, except for start/goal vertices, and this requires extra aisle space in warehouses. Moreover, the greedily reserved paths have relatively large redundancy in the parallel execution of tasks.

We focus on a different type of solution method, PIBT (Okumura et al., 2022), as a continuous MAPF solver that can be applied to narrow maps with dense populations of agents. Since this method is also a greedy approach, there is a different restriction as mentioned below, but it is relatively acceptable in the case of warehouses.

2.3 PIBT

PIBT (Okumura et al., 2022) is a solution method for the (continuous) MAPF problem that can be considered a variant of the push-and-rotate approach. The

method employs relatively simple operations that introduce the priority of agents and limited backtracking into the push-and-rotate process, and each agent determines its next move at each time step (Fig. 2).

For continuous problems, each agent a_i has its list of subgoals and moves to the first subgoal, and it also has a priority $p(a_i)$ based on the elapsed time for the current subgoal and its name (for tie-break). Each agent a with locally highest priority $p(a)$ initiates the process (line 6 in Fig. 2). An agent a_i selects its most preferred move based on evaluation function f_i (line 14 in Fig. 2) and asks a neighboring agent on a_i 's shortest path to move if necessary (push) (lines 16 and 17 in Fig. 2). Here we simply define f_i as the distance from the current location of a_i to its first subgoal and implement f_i as a distance map that is computed for the first subgoal. The pushed agent a_j tries to move to its neighboring vertex, and it also pushes a_j 's neighboring agent if necessary. If all agents obstructing agent a_i can move, a chain of moves is determined (line 18 in Fig. 2). Here, a pushed agent in the chain may move into the current location of the agent that initially pushes (rotation). If there is no agent obstructing agent a_i , its move is determined (line 24 in Fig. 2).

If one of the (pushed) agents cannot move, backtracking is performed (line 29 in Fig. 2), and an agent tries to push one of its other neighboring agents. This search is limited, and an agent that cannot move stays in its current location at the next time step (line 28 in Fig. 2).

For all vertices v_i and for all vertices v_j neighboring v_i , if the vertices v_i and v_j are contained in a cycle, agents' locations can be rotated through the cycle, and PIBT can solve such a problem. Due to this condition, the method cannot be applied to the maps with a dead end. On the other hand, if the above condition is satisfied, the method can work with narrow aisles and dense populations of agents, even if all non-obstacle vertices/cells are occupied by agents.

For MAPD problems, we employ a basic greedy task allocation method where each idle agent selects a task whose pickup location is nearest from its current location.

2.4 Aim of Study

For MAPF problems, most of solution methods employ a pathfinding algorithm on time-space graphs that computes individual agents' paths, and the paths are reserved even if a given situation can change. Complete algorithms that resolve collisions among agents require a relatively high computational cost, while several greedy methods suffer from redundancy

in floor plans and sparse parallel execution of tasks. These fundamental methods adjust individual agents' paths without considering additional information of map structures and traffic.

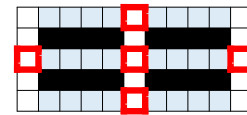
On the other hand, the solution methods based on push-and-rotate operations are flexible for various situations, but there is global redundancy in the moves of agents due to myopic planning. While previous studies partially integrated pathfinding and reservation into PIBT to mitigate this drawback (Okumura et al., 2019), there might be opportunities to investigate several heuristics of agents' moves and adjustments of the maps themselves. We address this issue in warehouse environments with narrow aisles. In particular, we concentrate on the influence of map settings and fundamental additional heuristics that might be components to control relatively low-level solution methods such as PIBT without precise reservation of agent paths. Since it is not straightforward to grasp such influences on a greedy solution method that is affected by several biases, we experimentally investigate a few properties as a first case study.

3 ADJUSTING MAP SETTINGS AND APPLYING HEURISTICS BASED ON MAP STRUCTURES

3.1 Limitation on Moves

A simple way to improve the control of agents is a limitation on moves in maps. Here, conventional grid-like maps represented with undirected graphs are replaced by directed graphs, and some edges are removed to indicate lanes. There is no edge conflict on such a directed edge, and the settings with directed edges have been evaluated as easier example problems in previous studies with the time-space A* algorithms (Li et al., 2021). This modification affects the shortest paths to goals, and possible moves in push operations in PIBT.

When there is relatively sufficient space in a floor plan, setting pairs of opposite-direction lanes is intuitively reasonable. Each lane inhibits agents to move in inverted directions but allows them to turn into another lane. While such maps are effective in general applications, they are particularly suitable to solution methods such as PIBT that locally solves collisions on demand. On the other hand, for the solution methods based on endpoints, two lanes are necessary for each aisle in addition to endpoint zones, particularly in warehouse settings, and this space redundancy may not be acceptable. Note that PIBT allows agents



Non-obstacle cells with thick frames are intersections, and other non-obstacle cells are aisles.

Figure 3: Map Structure.

to pass through pickup and delivery zones in warehouses, and the number of additionally required lanes is relatively small. Therefore, we consider that a setting using such opposite lanes is a promising solution for improving the performance of PIBT for the cases where relatively sufficient space is available.

In the case of narrow aisle cells whose width is a single unit, one-way lanes can be introduced. However, such settings might be too restrictive on the moves of agents in several situations, and thus there are an incentive to analyze the effect of such one-way lanes.

When designing maps with directed graphs, care must be taken not to lose cycles for any vertex, and satisfying this requirement might sometimes be complicated.

3.2 Employing Knowledge of Map Structure

In the following, we extract the map structure and employ that information for heuristics in PIBT. The non-obstacle cells (vertices) in the narrow grid-like maps are categorized into aisles and intersections (Fig. 3). An aisle is a set of neighboring non-obstacle cells that has two neighboring non-obstacle cells, while an intersection is a single non-obstacle cell that has three or four neighboring non-obstacle cells. For simplicity, we assume that maps consist of narrow aisles whose width is one, and whose intersections are single cells. Such structures of simple warehouses can be captured in relatively simple preprocessing or given as additional attributes of maps. In our experiment, the aisles and intersections were extracted from manually labeled areas (for analysis) in a preprocessing by considering the number of neighboring non-obstacle cells, although those can be easily extracted without such labels.

The aisles and intersections are managed as areas. Each area has its basic information including its label, type, set of contained cells, and set of cells neighboring its ends. Each non-obstacle cell also has a label of its corresponding area, and the label is used for cross reference.

We employ a shared information store that has the



Circles: agents. Arrow: a part of an agent's shortest path to its goal. Dominant agent a_k has highest priority $p(a_k)$ within an aisle, and a neighboring intersection that is affected by a_k 's shortest path is identified.

Figure 4: Dominant agent in aisle.

information on map structures and agents related to the parts of the structure. It is assumed that agents access the shared information in a consistent order and then update or refer to the information relevant to them.

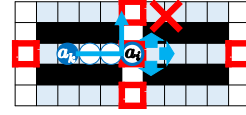
3.3 Managing Information of Dominant Agent in Aisle

With the knowledge of map structure, we summarize the information of agents in each aisle. Here, we focus on the agent that has the highest priority among the agents in the same aisle and call it the dominant agent in the aisle (Fig. 4). This is approximated information, since the agent might not push other agents or might be pushed by an agent outside of the aisle.

In addition to the name of the dominant agent, its preferred move direction is computed and stored. Actually, the preferred direction is represented as a temporary subgoal cell in relation to the current aisle. When a dominant agent goes through the current aisle, one of two neighboring intersections that has the smaller distance value to the agent's goal is selected.

On the other hand, when a dominant agent's goal is within the current aisle, the temporary subgoal cell is selected as follows. When the dominant agent is in its goal cell, the current cell is selected, and this indicates that the agent will not push any other agents. Otherwise, the direction from the dominant agent's current cell to its goal in the current aisle is computed, and a neighboring intersection in that direction is selected. Note that when an agent arrives at its goal cell, its priority is reset. Therefore, such an agent is not the dominant agent in most cases.

The information of the dominant agents is initialized before the recursive process of PIBT (line 1 in Fig. 6). It should also be updated in the process of PIBT, since several agents' next locations and priorities are determined and reserved in this process. When the next states of agents are determined, the information on corresponding aisles are marked as 'dirty' (lines 20, 27 and 32 in Fig. 6) and recomputed using the next states if necessary (line 15 in Fig. 6).



Agent a_i should not be pushed into the path of first pushing agent a_k .

Figure 5: Replacement of agent's location.

3.4 Direction Selection 1: Replacement of Agents' Locations

We employ the information on map structure to adjust the direction selection of agents in the recursive process of PIBT. Since agents cannot pass each other in aisles in the case of narrow maps, the substantial selection of moving direction is performed by the agents at intersections.

An important interaction among agents is to replace the locations of agents going in opposite directions around an intersection (Fig. 5). For this operation, we extend the recursive process of PIBT so that the information of the root (first pushing) agent a_k in the recursion is inherited in the process (lines 7, 10 and 18 in Fig. 6). When an agent a_i at an intersection is pushed, the agent evaluates the preferred move direction d_k of the first pushing agent by referring to the information of a_k . Here, we assume that a default tie-breaking order of moving direction is common to all agents. If direction d_k is the direction where the distance from pushed agent a_i to its goal increases, a_i 's preference value f'_i of direction d_k is modified to avoid the move to the direction (line 15 in Fig. 6). Since we simply use distance values to a goal as the preference values f_i of directions, the distance value for d_k is temporally increased to a sufficiently large value as follows.

$$f'_i(v, a_k) = \begin{cases} \infty & v \text{ is in direction } d_k. \\ f_i(v) & \text{otherwise} \end{cases} \quad (1)$$

By this modification, a pushed agent in an intersection will avoid a move where the agent is pushed into the first pushing agent's moving path.

Although other agents in a push chain might also affect an agent pushed at an intersection, we focus only on the initiator agent of the push chain as the most influential one.

3.5 Direction Selection 2: Tie-Breaking for Shortest Paths to Avoid Dominant Agents

Another situation to be considered by an agent at an intersection is the dominant agents in neighbor-

```

1 UpdateAislesInformation() // *1
2 UNDECIDED ←  $\mathcal{A}(t)$  // agents list
3 OCCUPIED ←  $\emptyset$  // vertices list
4 update priorities  $p_i(t)$  for all agents  $a_i$ 
5 while UNDECIDED ≠  $\emptyset$  do
6   let  $a$  be the agent with highest priority  $p(a)$  in
   UNDECIDED
7   PIBT( $a, a, \perp$ ) // *2
8 end while

10 function PIBT( $a_r, a_i, a_j$ ) // *2
11 UNDECIDED ← UNDECIDED \ { $a_i$ }
12  $C_i$  ← ( $\{v | (v_i(t), v) \in E\} \cup \{v_i(t)\}$ )
13   \ ( $\{v_j(t)\} \cup OCCUPIED$ )
14 while  $C_i$  ≠  $\emptyset$  do
15    $v_i^* \leftarrow \arg \max_{v \in C_i} f'_i(v, a_r)$  // most preferred move // *1, *2
16   OCCUPIED ← OCCUPIED  $\cup \{v_i^*\}$ 
17   if  $\exists a_k \in UNDECIDED$  s.t.  $v_i^* = v_k(t)$  then
18     if PIBT( $a_r, a_k, a_i$ ) is valid then // *2
19        $v_i(t+1) \leftarrow v_i^*$ 
20       MarkUpdateOfAislesInformation() // *1
21       return valid
22     else
23        $C_i \leftarrow C_i \setminus OCCUPIED$ 
24     end if
25   else
26      $v_i(t+1) \leftarrow v_i^*$ 
27     MarkUpdateOfAislesInformation() // *1
28     return valid
29   end if
30 end while
31  $v_i(t+1) \leftarrow v_i(t)$ 
32 MarkUpdateOfAislesInformation() // *1
33 return invalid
34 end function

```

$v_i(t)$: location of agent a_i at time step t
*1: information of dominant agents in aisles
*2: propagation of root (first pushing) agent

Figure 6: Extended PIBT at time step t .

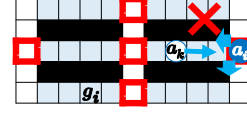
ing aisles (Fig. 7). With the information of dominant agents in aisles, each agent a_i at an intersection evaluates their influences. When a neighboring aisle is in the direction $d_{i,k}$ that decreases the distance to a_i 's goal, and the path of dominant agent a_k (with a priority greater than a_i) in the aisle contains the intersection of a_i 's current location, dominant agent a_k might push back agent a_i . In such a situation, the preference value f'_i of direction $d_{i,k}$ is modified to weakly avoid the move to that direction and to select another shortest path in a different direction if one exists (line 15 in Fig. 6). Here, the preference (distance) value for the direction is slightly increased by adding a small value less than unit distance so that priority value $p(a_k)$ of dominant agent a_k is taken into account as follows.

$$f'_i(v, *) = \begin{cases} f_i(v) + w \times p(a_k) & v \text{ is in direction } d_{i,k}. \\ f_i(v) & \text{otherwise} \end{cases} \quad (2)$$

Here, w is a sufficiently small coefficient value.

This weighting to the preference value for the direction can be integrated with the modification of the values to replace agents' locations.

As an exception case, if the current goal of agent



Agent a_i should not enter the aisle where dominant agent a_k with higher priority is coming, and a direction on another shortest path to goal g_i should be selected.

Figure 7: Tie-breaking of shortest path.

a_i at an intersection is within an aisle neighboring to the intersection, agent a_i enters to the goal aisle.

Note that this modification only affects the tie-breaking among shortest paths, and rerouting the agent to a detour requires an additional operation that introduces new intermediate subgoals and different distance computations. On the other hand, such a rerouting approach requires precise information on reserved paths of agents and might be ineffective, particularly in maps with narrow aisles where other agents are affected by such rerouting.

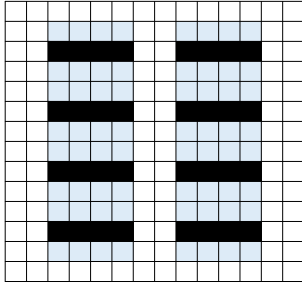
3.6 Limitation on Preference to Move by Map Settings

Our major interest is the control of solution methods using maps and related information, and we investigate the effect of map settings that might be complementary with the heuristics for solution methods. As mentioned in Section 3.1, the agents' moves can be partially restricted using directed graphs, while a modification must satisfy the condition of solvable problems for PIBT. On the other hand, this limitation can be separately applied to the maps used for computing the shortest paths of agents, which only requires the connectivity of directed graphs.

Since a fully automated optimization of such settings needs various analyses, we experimentally compare several heuristic settings that are manually set as a first investigation. Because PIBT has the capability to locally solve a conflict among agents, excessively restrictive settings might be totally ineffective. This also depends on the number of agents in the environment.

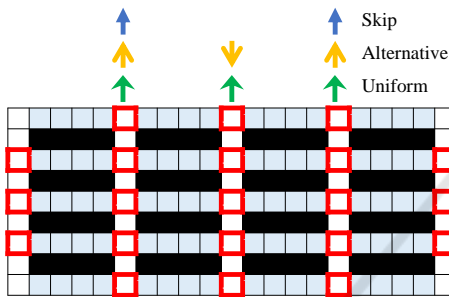
A basic approach loosely causes rotation moves of agents by a partial restriction of their moves in the map referenced for the computation of their shortest paths. In addition, such a restriction distributes shortest paths of several agents that have to avoid restricted moves in several vertices.

We select influential 'streets' in a map and set the restriction on moves to them (Fig. 9). While there are a number of possible selections, we choose several combinations of vertical streets whose ends are



When opposite lanes are set, agents cannot move to backward and must move on left-hand lanes, while they can move to a neighboring opposite lane.

Figure 8: Relatively sparse map for opposite lanes.



Inverted move directions on several vertical streets are restricted.

Figure 9: Map with narrow aisles.

the intersections with the top and bottom cells in a map. Moreover, we investigate the cases where all cells in the streets are restricted and the cases where the restriction is only applied to intersections.

4 EVALUATION

Here, we report the experimental evaluation of our proposed approaches.

4.1 Settings

We employed several maps that are modified versions of typical Kiva-like warehouse maps (Ma et al., 2017), and they are narrowed by eliminating some spaces. Figure 8 shows an example map for the case with opposite lanes as mentioned in Section 3.1, and Fig. 9 shows the case of narrower maps with aisles whose width is a single unit. We varied the number of agents up to the number of non-obstacle cells. For MAPD problems, NpT tasks were randomly generated at every time step, up to 500 tasks in total.

We compared the following methods that were incrementally combined and applied to the case of narrow aisles maps.

- PIBT. Our baseline implementation of PIBT based on the previous literature (Okumura et al., 2022).
- DR. Modification of direction selection to replace locations of agents going opposite directions shown in Section 3.4.
- DA. Modification of direction selection to weakly avoid an aisle in shortest paths where a dominant agent is going in the opposite direction, as mentioned in Section 3.5.

For the limitation on moves in the maps referenced in the computation of shortest paths as described in Section 3.6, we compared several combinations of one-way vertical streets in the map shown in Fig. 9, as follows.

- Skip. Except for the first and last columns of the map whose ends are parts of aisles, every other vertical street are selected, and the downward edges from the corresponding vertices are removed.
- Alternative. Except for the first and last column of the map, all vertical streets are selected, and the downward or upward edges from the corresponding vertices are removed. The directions of removed edges are inverted for each selected street alternatively.
- Uniform. Except for the first and last column of the map, all vertical streets are selected, and the downward edges from the corresponding vertices are removed.

In addition, the restriction is applied to all cells including aisles and intersections (All) or only intersections (Int).

We evaluated makespan (MS), which is the time steps to complete all tasks, and service time (ST) to complete each task. The results over ten executions with random initial locations of agents were averaged for each problem instance. The experiments were performed on a computer with g++ (GCC) 8.5.0 -O3, Linux 4.18, Intel (R) Core (TM) i9-9900 CPU @ 3.10 GHz and 64 GB memory.

4.2 Results

Tables 1-4 show the results, where the minimum value in each setting of ($\#agent, NpT$) is marked in bold.

Table 1 shows the cases with/without opposite lanes. We can confirm that the setting of lanes reduced time steps for agent moves in most cases. In the

Table 1: Two opposite lanes (PIBT).

| #agent | | 10 | | 30 | | 60 | | 90 | | 120 | | 150 | |
|--------|------------------------|-----------------------|-----------------------|-----------------------|----------------------|-----------------------|---------------------|-----------------------|---------------------|-----------------------|---------------------|---------------------|----------------------|
| NpT | prb. | MS | ST | MS | ST | MS | ST | MS | ST | MS | ST | MS | ST |
| 1 | no limit opp. lanes | 654.3 644.1 | 76.7 73.3 | 529.2 524.2 | 24.2 21.7 | 530.4 524.8 | 24.8 24.3 | 534.2 527 | 28.8 25.9 | 545.3 533.5 | 39.2 29.7 | 576 539.7 | 61.4 35.8 |
| 10 | no limit opp. lanes | 633.4 615.9 | 279.9 270.2 | 287.6 247.8 | 113.4 94.9 | 222.4 183.2 | 82.1 64.7 | 213.8 176.1 | 81.7 60.8 | 247.6 184.6 | 94.7 63.9 | 305.9 200 | 125.3 72.1 |

Table 2: Narrow aisles (no restriction on moves in computation of shortest paths).

| #agent | | 10 | | 30 | | 60 | | 90 | | 120 | | 125 | |
|--------|-------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|---------------|--------------|
| NpT | alg. | MS | ST | MS | ST | MS | ST | MS | ST | MS | ST | MS | ST |
| 1 | PIBT | 859.5 | 175.3 | 554.9 | 37.6 | 588.4 | 65.6 | 691.4 | 121.2 | 1040.8 | 297.4 | 1394.0 | 454.6 |
| | DR | 801.4 | 144.9 | 543.2 | 34.4 | 575.6 | 61.1 | 635 | 107.2 | 1003.9 | 282.4 | 1412.2 | 457.5 |
| | DA | 873.5 | 180.1 | 549.4 | 37.3 | 592.8 | 65.7 | 685.8 | 120.8 | 1018.1 | 290.3 | 1391.8 | 467.2 |
| | DR+DA | 799.1 | 144.5 | 543 | 34.2 | 569.7 | 60.8 | 631 | 101.6 | 968.2 | 264.5 | 1366.9 | 436.7 |
| 10 | PIBT | 849.5 | 384.1 | 465.3 | 195.5 | 461.5 | 194.7 | 582.4 | 252.3 | 907.8 | 421.0 | 1281.2 | 576.1 |
| | DR | 785.4 | 349.2 | 416.6 | 176.0 | 412.3 | 175.3 | 512.8 | 225.5 | 885.4 | 395.3 | 1256.7 | 553.5 |
| | DA | 840.9 | 377.8 | 458.6 | 193.3 | 451.6 | 189.8 | 571.6 | 240.7 | 901.3 | 409.4 | 1274 | 572.5 |
| | DR+DA | 779.1 | 345.6 | 418.7 | 175.7 | 410.8 | 175.0 | 500.4 | 221.9 | 870.2 | 397.4 | 1242.8 | 537.5 |

Table 3: Narrow aisles (restriction on moves in computation of shortest paths, PIBT).

| #agent | | 10 | | 30 | | 60 | | 90 | | 120 | | 125 | |
|---------|----------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|---------------|---------------|---------------|--------------|
| NpT | prb. | MS | ST | MS | ST | MS | ST | MS | ST | MS | ST | MS | ST |
| 1 | no limit | 859.5 | 175.3 | 554.9 | 37.6 | 588.4 | 65.6 | 691.4 | 121.2 | 1040.8 | 297.4 | 1394 | 454.6 |
| | Skip-Int | 864.8 | 175.0 | 553.7 | 37.1 | 589.3 | 65.4 | 688.7 | 124.5 | 1042.4 | 308.9 | 1396.3 | 463.6 |
| | Skip-All | 934 | 208.7 | 579.3 | 49.0 | 606.5 | 72.1 | 710 | 128.4 | 1067.8 | 299.2 | 1375.7 | 435.0 |
| | Alt-Int | 868.4 | 180.3 | 546.1 | 36.5 | 590.6 | 63.7 | 682.9 | 115.9 | 1028.1 | 295.9 | 1371 | 460.0 |
| | Alt-All | 905.6 | 196.2 | 566.1 | 43.1 | 603.9 | 66.1 | 687.6 | 116.9 | 1042.3 | 284.9 | 1328.2 | 420.5 |
| 10 | no limit | 849.5 | 384.1 | 465.3 | 195.5 | 461.5 | 194.7 | 582.4 | 252.3 | 907.8 | 421.0 | 1281.2 | 576.1 |
| | Skip-Int | 864.5 | 392.0 | 455.3 | 193.8 | 454.9 | 189.0 | 588.3 | 251.5 | 922.2 | 416.9 | 1288.8 | 582.1 |
| | Skip-All | 934.8 | 428.7 | 527.9 | 228.4 | 502.1 | 212.0 | 618.3 | 263.9 | 942.9 | 415.2 | 1236.8 | 543.6 |
| | Alt-Int | 859.2 | 387.6 | 444.1 | 189.0 | 431 | 180.8 | 571.9 | 245.0 | 916.8 | 416.6 | 1265.1 | 569.3 |
| | Alt-All | 911.2 | 415.7 | 511.5 | 219.1 | 471.9 | 197.2 | 585.9 | 244.6 | 923.7 | 411.3 | 1266.8 | 549.4 |
| 10 | Uni-Int | 922 | 204.3 | 552.3 | 38.4 | 576.3 | 60.0 | 660.3 | 109.9 | 1009.3 | 292.7 | 1425.9 | 480.6 |
| | Uni-All | 1183.5 | 334.0 | 657.8 | 84.4 | 623.2 | 70.5 | 714.7 | 125.3 | 1046.1 | 286.0 | 1314.5 | 411.3 |
| | Uni-Int | 911.3 | 414.1 | 452.3 | 193.0 | 422.4 | 176.6 | 537 | 225.6 | 896.6 | 404.6 | 1291.4 | 581.3 |
| Uni-All | 1162 | 542.6 | 635.1 | 280.3 | 527.9 | 219.2 | 601.3 | 255.9 | 953.2 | 406.8 | 1230.8 | 539.3 | |

Table 4: Narrow aisles (restriction on moves in computation of shortest paths, DR+DA).

| #agent | | 10 | | 30 | | 60 | | 90 | | 120 | | 125 | |
|--------|----------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|---------------|--------------|
| NpT | prb. | MS | ST | MS | ST | MS | ST | MS | ST | MS | ST | MS | ST |
| 1 | no limit | 799.1 | 144.5 | 543 | 34.2 | 569.7 | 60.8 | 631 | 101.6 | 968.2 | 264.5 | 1366.9 | 436.7 |
| | Skip-Int | 815.1 | 152.8 | 545.4 | 34.6 | 574.7 | 62.6 | 643.6 | 107.4 | 997 | 278.4 | 1361.9 | 438.6 |
| | Skip-All | 819.2 | 154.9 | 541.3 | 35.9 | 576.3 | 63.6 | 643.9 | 110.7 | 1037.8 | 293.1 | 1457.5 | 464.6 |
| | Alt-Int | 818.9 | 155.0 | 543.7 | 33.8 | 575.9 | 59.4 | 634 | 103.3 | 1003.2 | 276.8 | 1371.6 | 443.0 |
| | Alt-All | 820 | 155.2 | 545 | 34.1 | 575.6 | 60.5 | 639.5 | 104.5 | 1046.1 | 288.4 | 1464.2 | 448.1 |
| 10 | no limit | 779.1 | 345.6 | 418.7 | 175.7 | 410.8 | 175.0 | 500.4 | 221.9 | 870.2 | 397.4 | 1242.8 | 537.5 |
| | Skip-Int | 791.3 | 352.3 | 420.7 | 178.3 | 415.7 | 177.1 | 510.8 | 227.3 | 881.7 | 397.0 | 1232.8 | 539.3 |
| | Skip-All | 797.9 | 357.7 | 427.2 | 178.8 | 424.9 | 181.4 | 514.2 | 230.8 | 917.2 | 401.5 | 1320.8 | 563.4 |
| | Alt-Int | 797.7 | 357.1 | 412.2 | 173.7 | 394.5 | 168.7 | 506.1 | 226.8 | 873.5 | 394.0 | 1248.2 | 546.3 |
| | Alt-All | 795.6 | 356.7 | 413 | 173.5 | 402.8 | 171.3 | 492.6 | 219.1 | 952.1 | 409.1 | 1352.5 | 563.2 |
| 10 | Uni-Int | 857.6 | 384.2 | 418.7 | 179.3 | 372.3 | 157.2 | 465.6 | 202.0 | 871.2 | 390.8 | 1234.9 | 540.4 |
| | Uni-All | 890.7 | 402.3 | 433.3 | 185.4 | 383.7 | 158.8 | 477.6 | 208.2 | 955.1 | 408.9 | 1403 | 596.6 |

cases of relatively high concurrency of tasks ($NpT = 10$), and in the cases of dense populations of agents, MS and ST were relatively reduced.

Table 2 shows the case of narrow aisles without the restriction on moves in the computation of shortest paths. The influence of heuristics for direction selection varied with the density of agents' population. Except very dense cases, DR was basically effective in reducing MS and ST, while DA alone was often not so effective. It revealed that the replacement of agents' locations is a fundamental operation and more important. On the other hand, the combination of them was relatively effective in most cases. We found that these local heuristics might disturb the process of original solution method in several instances, while the results were relatively better in average.

Table 3 shows the case of narrow aisles with a restriction on moves in the computation of shortest paths. There seems to be several complicated trade-offs among different settings. While the comparison between Skip and Alternative revealed that a sufficient restriction is necessary to affect agents' moves, the restriction is often excessive in the cases of sparse populations of agents. In most cases, Int that only limit moves for intersection vertices reduced the time steps than All. It revealed some trade-off between the limitation/control of agents' moves and the planning of agents on demand. The combination of Uniform and Int was relatively better than others except the cases of sparse populations of agents. Different restrictions on agents' moves might enforce different rotation paths, and it can be effective according to situations, as low-cost heuristics.

The case of combinations of heuristics for direction selection and the restriction on agents' moves in the computation of shortest paths is shown in Tbl. 4. Similar to the case without any restriction on agents' moves, the additional heuristics for PIBT were relatively effective where the population of agents is not too dense, and both approaches were complementary. On the other hand, it was revealed that the combined heuristics were ineffective with the effect of map settings in very dense cases. In the dense cases, a global control/limitation method of agents' moves is more important and the heuristics for local interactions of among can be inconsistent with such a control method. There might be opportunities to employ dedicated heuristics in agents' interaction for the dense cases.

With our experimental implementation, the computation time was 0.24 and 0.26 seconds for PIBT and DR+DA, in the case of $NpT = 1$ and 125 agents shown in Tbl. 2.

5 CONCLUSION

In developing methods that employ certain information of maps and traffic as the heuristics to control low-level solution methods for continuous multi-agent pathfinding/pickup-and-delivery problems, we focused on the case of such problems in narrow warehouse environments and the solution method PIBT. For this case study, we experimentally investigated the effect of map settings and additional heuristics based on the structures of maps. The experimental results reveal the potential to employing such fundamental components to improve the behavior of solution methods, while the optimum combination of such settings remains a future work. One possible approach is using statistics and learning for cases of several situations in environments. Although we addressed the static restriction on agents' moves as a first case study, methods to dynamically apply such settings based on summarized information of agent behaviors without precise reservation of agents' paths should also be included in future work. For real-world applications, there are opportunities of several extensions including those of PIBT itself in practical situations. In well controlled automated warehouses satisfying the solvable conditions of this kind of lightweight MAPF algorithms, such an application with effective add-ons including the proposed approaches can be promising.

ACKNOWLEDGEMENTS

This study was supported in part by The Public Foundation of Chubu Science and Technology Center (thirty-third grant for artificial intelligence research) and JSPS KAKENHI Grant Number 22H03647.

REFERENCES

- Andreychuk, A., Yakovlev, K., Boyarski, E., and Stern, R. (2021). Improving Continuous-time Conflict Based Search. In *Proceedings of The Thirty-Fifth AAAI Conference on Artificial Intelligence*, volume 35, pages 11220–11227.
- Andreychuk, A., Yakovlev, K., Surynek, P., Atzmon, D., and Stern, R. (2022). Multi-agent pathfinding with continuous time. *Artificial Intelligence*, 305:103662.
- Barer, M., Sharon, G., Stern, R., and Felner, A. (2014). Suboptimal Variants of the Conflict-Based Search Algorithm for the Multi-Agent Pathfinding Problem. In *Proceedings of the Annual Symposium on Combinatorial Search*, pages 19–27.

- De Wilde, B., Ter Mors, A. W., and Witteveen, C. (2014). Push and rotate: A complete multi-agent pathfinding algorithm. *J. Artif. Int. Res.*, 51(1):443–492.
- Hart, P., N. N. and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Science and Cybernetics*, 4(2):100–107.
- Hart, P., N. N. and Raphael, B. (1972). Correction to 'a formal basis for the heuristic determination of minimum-cost paths'. *SIGART Newsletter*, (37):28–29.
- Li, J., Tinka, A., Kiesel, S., Durham, J. W., Kumar, T. K. S., and Koenig, S. (2021). Lifelong Multi-Agent Path Finding in Large-Scale Warehouses. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence*, pages 11272–11281.
- Liu, M., Ma, H., Li, J., and Koenig, S. (2019). Task and Path Planning for Multi-Agent Pickup and Delivery. In *Proceedings of the Eighteenth International Conference on Autonomous Agents and MultiAgent Systems*, pages 1152–1160.
- Luna, R. and Bekris, K. E. (2011). Push and swap: Fast cooperative path-finding with completeness guarantees. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, volume 1, pages 294–300.
- Ma, H., Harabor, D., Stuckey, P. J., Li, J., and Koenig, S. (2019). Searching with consistent prioritization for multi-agent path finding. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, pages 7643–7650.
- Ma, H., Li, J., Kumar, T. S., and Koenig, S. (2017). Lifelong Multi-Agent Path Finding for Online Pickup and Delivery Tasks. In *Proceedings of the Sixteenth Conference on Autonomous Agents and MultiAgent Systems*, pages 837–845.
- Miyashita, Y., Yamauchi, T., and Sugawara, T. (2023). Distributed planning with asynchronous execution with local navigation for multi-agent pickup and delivery problem. In *Proceedings of the Twenty-Second International Conference on Autonomous Agents and Multiagent Systems*, page 914–922.
- Okumura, K., Machida, M., Défago, X., and Tamura, Y. (2022). Priority Inheritance with Backtracking for Iterative Multi-Agent Path Finding. *Artificial Intelligence*, 310.
- Okumura, K., Tamura, Y., and Défago, X. (2019). winPIBT: Expanded Prioritized Algorithm for Iterative Multi-agent Path Finding. *CoRR*, abs/1905.10149.
- Sharon, G., Stern, R., Felner, A., and Sturtevant, N. R. (2015). Conflict-Based Search for Optimal Multi-Agent Pathfinding. *Artificial Intelligence*, 219:40–66.
- Silver, D. (2005). Cooperative Pathfinding. pages 117–122.
- Čáp, M., Vokřínek, J., and Kleiner, A. (2015). Complete Decentralized Method for On-Line Multi-Robot Trajectory Planning in Well-Formed Infrastructures. In *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling*, pages 324–332.
- Yakovlev, K. S. and Andreychuk, A. (2017). Any-angle pathfinding for multiple agents based on SIPP algorithm. In *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling*, pages 586–593.
- Yamauchi, T., Miyashita, Y., and Sugawara, T. (2022). Standby-based deadlock avoidance method for multi-agent pickup and delivery tasks. In *Proceedings of the Twenty-First International Conference on Autonomous Agents and Multiagent Systems*, pages 1427–1435.