Proceedings of the 41st Chinese Control Conference
July 25-27, 2022, Hefei, China

Multi-Robot Path Planning in Narrow Warehouse Environments with Fast Feasibility Heuristics

Jiaxi Huo[1], Ronghao Zheng[†1,2], Senlin Zhang[1,2], Meiqin Liu[2,3]
1. College of Electrical Engineering, Zhejiang University, Hangzhou 310027, China
2. State Key Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou 310027, China
3. Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, Xi'an, 710049, China
E-mail: {jiaxi.huo, rzheng, slzhang, liumeiqin}@zju.edu.cn

Abstract: In this paper, we present a novel method for solving the multi-robot path planning problem in narrow warehouse environments. Robots may have to be operated in close proximity to each other in narrow lanes between pod clusters, which challenges robot-robot collision avoidance. This motivates us to consider an efficient algorithm to generate collision-free robot paths in narrow warehouse environments. The algorithm builds a two-stage scheme based on integer programming: (i) multi-robot path planning under one-way constraints; and (ii) fast feasibility heuristics. The developed algorithm has two advantages: (i) the proposed scheme successfully avoids the situation where the robots collide in narrow lanes; and (ii) by using the fast feasibility heuristics which replaces the integer programming solver's built-in feasibility heuristics, the feasibility heuristics' running time can be effectively reduced while producing better initial feasible solutions for the integer programming model solving process. Compared with the existing optimal, sub-optimal, and polynomial-complexity algorithms, our developed algorithm can leverage the advantage of integer programming to effectively reach a balance between running time and optimality in narrow warehouse environments. This point is demonstrated in the simulations.

Key Words: Path planning, Multi-robot or multi-agent systems, Collision avoidance, Logistics.

1 Introduction

Multi-Robot System (MRS) improves the work efficiency in many scenarios, e.g., autonomous logistic warehouse [1]. Among the studies of MRS, the study of the Multi-robot Path Planning (MPP) problem, despite its computational complexity[2], has been extensively carried out for decades for its diverse applications: pickup and delivery[3], communication and coordination[4], to list a few.

This paper addresses the collision-free MPP problem in such a narrow warehouse environment that arranges pods in clusters with narrow lanes in between to maximize the storage capacity. Typically, the Amazon fulfillment centers use KIVA systems in [1]. Basically, these robots lift an entire shelf and bring the entire shelf (called a "pod") to another storage area. We consider a pods-transfer scenario, with a team of mobile robots transferring pods between clusters of the same size and rectangular shape. The narrow warehouse environment is a well-formed infrastructure such that a robot carrying a pod starts its tour inside the cluster, then enters its initial position in a narrow lane; when arriving at its destination, it will go under a nearby cluster and disappear from the lanes, and every two robots cannot move side by side. The hard robot-robot collision occurs when two robots simultaneously move to the same position (meet collision) or reverse their positions (head-on collision) in a narrow lane. Primarily, it is difficult to deal with head-on collisions in a narrow lane due to the lack of free space. We propose two crucial objectives based on distance optimality involved in solving the MPP problem: the total distance and the maximum (single-robot traveled) distance.

We propose the Two-stage Integer Programming-based (TIP) algorithm to solve the MPP problem in narrow warehouse environments. To avoid the hard robot-robot collision in narrow lanes, the TIP algorithm is set up with the collision-free constraints. The framework is shown in Fig. 1.

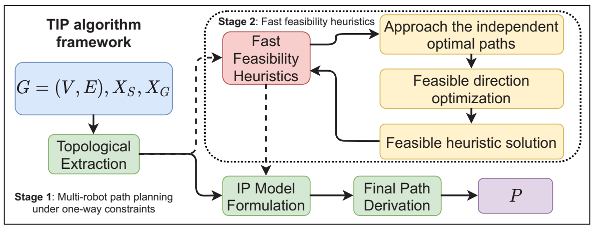
TIP algorithm framework
$G=(V,E),X_S,X_G$
Topological Extraction
Stage 1: Multi-robot path planning under one-way constraints
Stage 2: Fast feasibility heuristics
Fast Feasibility Heuristics
Approach the independent optimal paths
Feasible direction optimization
Feasible heuristic solution
IP Model Formulation
Final Path Derivation
P

Fig. 1: Framework of TIP algorithm.

We present our contributions in the following:

(1) We build the entire framework of the proposed IP-based MPP algorithm that avoids the robot-robot collision in narrow lanes. We showcase its efficiency in solving the MPP problem in narrow warehouse environments.

(2) To further improve the IP-based MPP algorithm's solving performance, we introduce novel fast feasibility heuristics, which are much more efficient than the existing IP solver's general built-in feasibility heuristics in narrow warehouse environments.

(3) Meanwhile, we conduct simulations in simulated narrow warehouse environments. The proposed IP-based MPP algorithm scales well to many robots and produces high-quality solutions in narrow warehouse environments compared with the existing algorithms.

2 Related works

The MPP problem has been studied for decades due to its diverse applications [5–7]. Multiple algorithms are proposed to solve the optimal MPP problem, such as Mixed Integer Programming (MIP) [8] and Conflict Based Search (CBS) [9]. The optimal formulation of the MPP problem gener-

The work was supported by the National Natural Science Foundation of China under Grants 61873235 and 62173294: the NSFC-Zhejiang Joint Fund for the Integration of Industrialization and Informatization under Grants U1909206 and U1809212: the Fundamental Research Funds for the Zhejiang Provincial Universities under Grant 2021XZZX014.

† Corresponding author.

ates the optimal collision-free paths but maybe intractable even for a few robots. Accordingly, at the cost of optimality, polynomial-time algorithms are proposed in practice, such as Priority Inheritance with Backtracking (PIBT) [10]. Some works introduce sub-optimal algorithms to balance optimality and efficiency, like IP with splits [11] and Enhanced CBS (ECBS) [12]. ECBS is a sub-optimal variant of CBS.

We show the motivations of this paper. Firstly, narrow lanes challenge the existing MPP algorithms. Optimal algorithms [8, 9] consume many computing resources to treat robot collisions in narrow lanes leading to the loss of scalability. The sub-problem algorithm [13] may be incomplete due to the failure of collision avoidance in narrow lanes.

Secondly, assigning directions to the map is studied to improve efficiency. [14] introduces the flow restriction idea to restrict robot movement and avoid head-on collisions, i.e., the robot movement is restricted by passing directions. The optimized direction assignment is then studied in this paper to prevent the robot-robot collision in narrow lanes.

Thirdly, fast feasibility heuristics (a.k.a., warm start) can accelerate the IP model's solving process by taking advantage of the initial solution [15]. IP solvers like Gurobi [16] allow feeding in initial binary solutions.

3 Preliminaries and assumptions

3.1 Multi-robot path planning (MPP) problem

This paper assumes that the narrow warehouse environment scheme is discretized into a grid of atomic locations called cells. Here, we suppose a binary matrix M of dimension $h \times w$ to represent the occupancy of the grid map. Each cell corresponds to a vertex v whose Cartesian coordinates are (i, j). We use $M(i,j) = 0$ and $M(i,j) = 1$ to represent that the corresponding cell is free and occupied, respectively. The narrow warehouse environment arranges pods clusters with narrow lanes, such that every two robots cannot move side by side. Let $G = (V, E)$ be a bidirectional gird map, with $V := \{(i,j)|M(i,j) = 0\}$ being the vertex set. Adjacency relationships are defined in four cardinal directions, i.e., for each $(i,j) \in V$, its neighborhood is $N((i,j)) := \{(i+1,j),(i,j+1),(i-1,j),(i,j-1)\} \cap V$. Here, each vertex v whose Cartesian coordinates are (i,j) in V has its corresponding index $|v| := j+(i-1)w$ as shown in the left part of Fig. 2. The MPP problem involves n robots $r_1, \dots, r_n$, where each robot r_i has a unique start position $s_i \in V$ and a unique goal position $g_i \in V$. We now denote the joint start configuration as $X_S := \{s_1, \dots, s_n\}$ and the goal configuration as $X_G := \{g_1, \dots, g_n\}$. The n robots here form a robot set $R := \{r_1, \dots, r_n\}$ which contains all the robot indices. The objective of the MPP problem is to find a set of a feasible path set for all robots. Here, we define a path for robot r_i (can be simplified to i) as a sequence of $T_i + 1$ vertices $P_i := \{p_i^0, \dots, p_i^{T_i}\}$ that satisfies: (1) $p_i^0 = s_i$; (2) $p_i^{T_i} = g_i$; (3) $\forall 1 \le t \le T_i, p_i^{t-1} \in N(p_i^t)$. All the robots should go under a pod cluster to transfer inventories in the narrow warehouse environments, we now give the formal well-formed infrastructure assumption.

Assumption 1 *A robot r_i occupies a vertex of G only during 0 to T_i, i.e., robot r_i is not an obstacle for other robots for all t, $t \notin \{0, \dots, T_i\}$.*

Considering the well-formed infrastructure assumption above, we want P to be collision-free, such that, $\forall 1 \le i < j \le n, 1 \le t \le \min(T_i, T_j)$, P_i, P_j must satisfy (1) $p_i^t \ne p_j^t$ (no meet collisions on vertices); (2) $(p_i^{t-1}, p_i^t) \ne (p_j^t, p_j^{t-1})$ (no head-on collisions on edges). Then, all edges of $G = (V, E)$ are assumed to have a length of 1 so that a robot traveling at unit speed can cross it in a single time step.

For a path $P_i \in P$, let $len(P_i)$ denote the length of the path P_i, which is the total number of times the robot r_i changes its residing vertex while following P_i. In this work, we consider two optimization MPP problems:

Problem 1 (Min-Total-Distance MPP): Given $\langle G, X_S, X_G\rangle$, find a collision-free path set P that navigates the robots from X_S to X_G and minimizes total robot distance $\sum_{i=1}^{n} len(P_i)$.

Problem 2 (Min-Maximum-Distance MPP): Given $\langle G, X_S, X_G\rangle$, find a collision-free path set P that navigates the robots from X_S to X_G and minimizes the maximum robot distance $\max_{1\le i\le n} len(P_i)$.

Instead of modeling collision avoidance for each robot and each step, this paper focuses on keeping the narrow lane clear by assigning the optimized direction to each lane.

4 Multi-robot path planning under one-way constraints

In the first stage of TIP algorithm, all the robots should pass each narrow lane in one design direction, which we call the one-way constraints. Motivated by the corridor node combination in [17], the design process of the first stage: (A) the original grid map is extracted into a topological map based on one-way constraints with fewer vertices and edges; (B) IP model is formulated on the topological map to achieve the robot paths on topological map and the optimized direction of each lane; (C) we transform the paths on the topological into the paths on the original grid map. Intuitively, the time to solve an IP model is often negatively correlated with the number of decision variables. This suggests that the time to solve the formulated IP model in step (B) can be reduced greatly compared with the IP model on the original grid map.

4.1 Topological extraction

A robot's movement direction in one narrow lane can be denoted using the robot's occupancy in two intersection vertices at both ends of the lane. Then, we can simplify G into a topological map that contains only the intersection vertices.

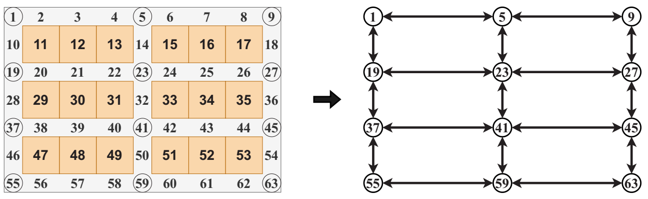

Fig. 2: From the initial grid map to the topological map.

Here, in Fig. 2, the vertices surrounded by circles (e.g., $v_1, v_5, v_9, v_{19}, \dots$) on the left side indicate the intersection vertices in the grid map. We extract them and add bidirectional edges between two neighboring intersection vertices.

Definition 1 *The topological map associated with $G = (V, E)$ is defined as: $\bar{G} = (\bar{V}, \bar{E})$ whose vertex set $\bar{V}$ contains and only contains the intersection vertices. For $v_i \in \bar{V}$, let $\bar{N}(v_i) := \{v_j|(v_i, v_j) \in \bar{E}\}$ be its neighborhood in $\bar{G}$.*

Here, we note that $\bar{G} = (\bar{V}, \bar{E})$ is a bidirectional map, such that, $(v_i, v_j) \in \bar{E}, (v_j, v_i) \in \bar{E}$, and $\bar{V} \subseteq V$. Formally, we define the correspondence between narrow lanes in the grid map and the edges in the topological map.

Definition 2 *Given a lane $\{v_i, v_j\}$ in the grid map, which is bounded by two intersection vertices v_i, v_j, is correlated to bidirectional edges in topological map (v_i, v_j) and (v_j, v_i).*

Recalling the well-formed infrastructure, X_S and X_G are placed in the lanes, i.e, $X_S \cap \bar{V} = \emptyset, X_G \cap \bar{V} = \emptyset$. We set the corresponding vertices of s_r and g_r in $\bar{G}$: $\bar{s}_r = \{\bar{s}_r^1, \bar{s}_r^2\}$ and $\bar{g}_r = \{\bar{g}_r^1, \bar{g}_r^2\}$, $\bar{s}_r \subseteq \bar{X}_S, \bar{g}_r \subseteq \bar{X}_G$ as the two intersection vertices in two ends of the narrow lane where s_r and g_r are placed. Significantly, each robot r should pass both $\bar{s}_r^1$ and $\bar{s}_r^2$, $\bar{g}_r^1$ and $\bar{g}_r^2$, because s_r and g_r in G are placed in the lanes bounded by $\bar{s}_r^1$ and $\bar{s}_r^2$, $\bar{g}_r^1$ and $\bar{g}_r^2$, respectively. As discussed, the IP model can be formulated and solved in $\bar{G}$ and the solution can be transformed to G without loss of generality.

4.2 Integer programming model formulation

We then derive our IP mode using $\bar{X}_S$, $\bar{X}_G$ and $\bar{G}$ above.

4.2.1 Path continuity constraints

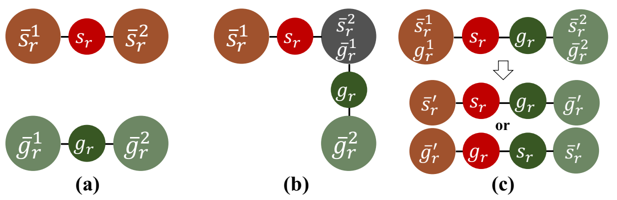

Fig. 3: Relative positions of initial and goal positions. Each circle represents one vertex in $\bar{G}$.

A path set $\bar{P}$ on $\bar{G}$ contains the virtual scheduled non-cyclic paths for all the robots on $\bar{G}$. Given $\langle \bar{G}, \bar{X}_S, \bar{X}_G \rangle$, the IP model here introduces a binary decision set $U := \{u_1, \ldots, u_n\}$, for each robot r: $u_r(v_i, v_j)$ for each edge $(v_i, v_j) \in \bar{E}$ to indicate whether $\bar{P}_r$ uses (v_i, v_j), i.e., if robot r traverses (v_i, v_j) virtually on $\bar{G}$, $u_r(v_i, v_j) = 1$; otherwise, $u_r(v_i, v_j) = 0$. U contains the motions of all robots.

For each $r \in R$, according to the relative position of initial and goal positions (as shown in Fig. 3), three different conditions are considered for path continuity constraints.

Above all, the global constraint (1) is respected in the three conditions so that vertices in $\bar{P}_r$ are consecutive on $\bar{G}$.

$$\sum_{v_j \in V \setminus \bar{N}(v_i)} u_r(v_i, v_j) = \sum_{v_j \in V \setminus \bar{N}(v_i)} u_r(v_j, v_i) = 0. \quad (1)$$

Condition (a): 0 shared vertex. As shown in part (a) of Fig. 3, initial and goal positions are on two disjoint lanes, such that the two lanes have no intersection, we now recall the representation: $\bar{s}_r = \{\bar{s}_r^1, \bar{s}_r^2\}$, and $\bar{g}_r = \{\bar{g}_r^1, \bar{g}_r^2\}$.

$$u_r(\bar{s}_r^2, \bar{s}_r^1) + u_r(\bar{s}_r^1, \bar{s}_r^2) = u_r(\bar{g}_r^2, \bar{g}_r^1) + u_r(\bar{g}_r^1, \bar{g}_r^2) = 1. \quad (2)$$

Constraint (2) ensures that all robots pass $\bar{s}_r$ and $\bar{g}_r$. We define an operator: $\mathcal{X}(v_i) = \sum_{v_j \in \bar{N}(v_i)} u_r(v_i, v_j) - \sum_{v_j \in \bar{N}(v_i)} u_r(v_j, v_i)$, to simplify the literature. Then constraints (3) - (5) ensure the path continuity in $\bar{G}$:

$$\mathcal{X}(\bar{s}_r^1) + u_r(\bar{s}_r^2, \bar{s}_r^1) = \mathcal{X}(\bar{s}_r^2) + u_r(\bar{s}_r^1, \bar{s}_r^2) = 1 \quad (3)$$

$$|\mathcal{X}(\bar{g}_r^1)| + u_r(\bar{g}_r^1, \bar{g}_r^2) = |\mathcal{X}(\bar{g}_r^2)| + u_r(\bar{g}_r^2, \bar{g}_r^1) = 1. \quad (4)$$

By constraint (3), only one of $\mathcal{X}(\bar{s}_r^1)$ and $\mathcal{X}(\bar{s}_r^2)$ equals to 1, such that, only one of $\bar{s}_r^1$ and $\bar{s}_r^2$ can be passed twice in $\bar{P}_r$. The same for $\bar{g}_r^1$ and $\bar{g}_r^2$ using constraint (4). If $v_i \notin \{\bar{s}_r^1, \bar{s}_r^2, \bar{g}_r^1, \bar{g}_r^2\}$, v_i can be passed only once in $\bar{P}_r$:

$$\sum_{v_j \in \bar{N}(v_i)} u_r(v_i, v_j) = \sum_{v_j \in \bar{N}(v_i)} u_r(v_j, v_i) \leq 1. \quad (5)$$

Condition (b): one shared vertex. As shown in part (b) of Fig. 3, robot's initial and goal positions are on two adjacent lanes, such that the two lanes have one intersection (e.g., the yellow circle), we set the shared vertex as $C(r)$, here, we note that $C(r) \in \{\bar{s}_r^1, \bar{s}_r^2, \bar{g}_r^1, \bar{g}_r^2\}$.

Firstly, constraint (2) is respected to ensure that the path pass $\bar{s}_r$ and $\bar{g}_r$. Secondly, if $v_i \notin \{\bar{s}_r^1, \bar{s}_r^2, \bar{g}_r^1, \bar{g}_r^2\}$, constraint (5) is respected. If $v_i = C(r)$, constraint (6) is respected, such that v_i can be visited twice in $\bar{P}_r$:

$$\left| \sum_{v_j \in \bar{N}(v_i)} u_r(v_i, v_j) - \sum_{v_j \in \bar{N}(v_i)} u_r(v_j, v_i) \right| \leq 1. \quad (6)$$

If $v_i \neq C(r)$, but $v_i \in \{\bar{s}_r^1, \bar{s}_r^2, \bar{g}_r^1, \bar{g}_r^2\}$, constraint (3) and constraint (4) are respected.

Condition (c): two shared vertices. In part (c) of Fig. 3, s_r, g_r are in the same lane. We set the configuration of robot r in the same lane as $\{\bar{s}'_r, \bar{g}'_r\}$, according to the relative positions of s_r and g_r, i.e., $\bar{s}_r = \bar{s}'_r = \arg\min_{v \in \bar{V}}(dist(v, s_r))$, $\bar{g}_r = \bar{g}'_r = \arg\min_{v \in \bar{V} \setminus \{\bar{s}'_r\}}(dist(v, g_r))$.

$$\begin{aligned} \sum_{v_j \in \bar{N}(\bar{s}'_r)} u_r(\bar{s}'_r, v_j) &= \sum_{v_j \in \bar{N}(\bar{g}'_r)} u_r(v_j, \bar{g}'_r) = 1, \\ \sum_{v_j \in \bar{N}(\bar{s}'_r) \setminus \{\bar{g}'_r\}} u_r(v_j, \bar{s}'_r) &= \sum_{v_j \in \bar{N}(\bar{g}'_r) \setminus \{\bar{s}'_r\}} u_r(\bar{g}'_r, v_j) \\ &= 0, \\ u_r(\bar{s}'_r, \bar{g}'_r) &+ u_r(\bar{g}'_r, \bar{s}'_r) = 1. \end{aligned} \quad (7)$$

Constraints (7) ensure that $\bar{s}'_r$ and $\bar{g}'_r$ should be visited m in order. If $v_i \in V \setminus \{\bar{s}'_r, \bar{g}'_r\}$, constraint (6) should be respected so that v_i can be visited only once in $\bar{P}_r$.

4.2.2 One-way constraints

One-way constraints are imposed on narrow lanes to prevent two robots from moving in opposite directions. As $\bar{G}$ is a bidirectional map, (v_i, v_j) and (v_j, v_i) stands for the same lane, we use $\{v_i, v_j\}$ to locate the corresponding lane. The decision variable $D(\{v_i, v_j\}), (v_i, v_j), (v_j, v_i) \in \bar{E}$ represents each lane's direction. Then, the size of D is $\frac{1}{2}|\bar{E}| \times 1$.

Numerically, we give the description of lane's direction correlated to $\{v_i, v_j\}$ on $\bar{G}$, v_i and v_j are the two vertices which bound the lane, $|v_i| < |v_j|$. The passing direction of $\{v_i, v_j\}$ can be divided as: from v_i to v_j, $D(\{v_i, v_j\}) = 1$; from v_j to v_i, $D(\{v_i, v_j\}) = -1$.

The direction can be restricted as: $\forall (v_i, v_j) \in \bar{E}$, $D(\{v_i, v_j\}) \in \{-1, 1\}$. We propose the one-way constraints as follows, for $\forall (v_i, v_j) \in \bar{E}, r \in R$:

$$\sum_{r \in R} u_r(v_i, v_j) = 0 \vee \sum_{r \in R} u_r(v_j, v_i) = 0 \quad (8)$$

$$|u_r(v_i, v_j) - u_r(v_j, v_i) - D(\{v_i, v_j\})| \leq 1. \quad (9)$$

It is worth noting that each lane is now one-way. Constraint (9) derives the numerical value of $D(\{v_i, v_j\})$: if $\exists r \in R$, such that $u_r(v_i, v_j) = 1, u_r(v_j, v_i) = 0$, then $D(\{v_i, v_j\})$ can and only can be 1, on the contrary, if $\exists r \in R$, such that $u_r(v_i, v_j) = 0, u_r(v_j, v_i) = 1$, then $D(\{v_i, v_j\})$ can and only can be -1.

4.2.3 Objective functions

We derive the two distance-based objective functions. For each edge $(v_i, v_j) \in \bar{E}$, the correlated weight $|(v_i, v_j)|$ equals to the length of lane, which is denoted as $W[\{v_i, v_j\}]$.

Minimize Total Distance (MinTotalDist): For all $r \in R$, with its binary decision variable $u_r(v_i, v_j)$ for each edge $(v_i, v_j) \in \bar{E}$ to indicate whether the virtual path $\bar{P}_r$ uses (v_i, v_j), the objective function can then be derived as:

$$\min \sum_{r \in R} \sum_{(v_i, v_j) \in \bar{E}} W[\{v_i, v_j\}] u_r(v_i, v_j).$$

Minimize Maximum Distance (MinMaxDist): We introduce an additional integer decision variable xm and $\forall r \in R$, we add the upper bound constraint $xm(r)$ for each robot's traveling distance and minimize it:

$$\sum_{(v_i, v_j) \in \bar{E}} W[\{v_i, v_j\}] u_r(v_i, v_j) \leq xm(r), \ \min_{r \in R} xm(r).$$

Referring to [11], there exists a bijection between the solution to the IP model U and all paths $\bar{P}$ from $\bar{X}_S$ to $\bar{X}_G$. Here, every $\bar{P}_r$ passes vertices in $\bar{s}_r$ to $\bar{g}_r$. We can then derive $\bar{P}_r$ easily from the solved binary decision variable $u_r \in U$.

4.3 Final path derivation

In the last part of Stage 1, we note that for each robot $r \in R$, $\bar{P}_r \subset \bar{V}$, to get actual path set P in G from $\bar{P}$, and all vertices $v \in V \backslash \bar{V}$ should be added between v_i and v_j, $v_i, v_j \in \bar{P}_r, (v_i, v_j) \in \bar{E}$. The path continuity constraints restrict that each $\bar{P}_r$ should traverse the lane where initial X_S and goal positions X_G are situated. Thus, X_S and X_G can be directly inserted into the scheduled path set. For each path $P_r \in P$, we ensure that every vertex in P_r is continuous in $G = (V, E)$. Finally, the velocity adjustment and waiting strategy on each robot is used to avoid the collision in the lane intersections if more than one robot is getting into the lane intersection at the same time step. The detailed description is omitted here due to the page limitation, while the complete performance is provided in the simulation video[1].

After the first stage of the TIP algorithm, to provide the formulated IP model with efficient feasible initial solutions to accelerate the solving process, we provide decoupled fast feasibility heuristics in the second stage.

5 Fast feasibility heuristics

The fast feasibility heuristics, which is used to operate the IP model's solving process with the warm start, is implemented in the second stage of the TIP algorithm (see Fig. 1). It is essential to point out that when the IP solver's general built in feasibility heuristics are slow in finding a feasible initial solution, it is helpful to preprocess the MPP problem to feed the IP solver with the initial robot paths and lane directions which satisfy all the constraints.

Referring to the manual of Gurobi solver [16], the built-in feasibility heuristics will be called to boost the solving process. Thus, we propose fast feasibility heuristics in $\bar{G}$, which is faster than Gurobi's feasibility heuristics with a better initial solution. The Gurobi solver will start warmly solving the IP model using the proposed fast feasibility heuristics instead of the built-in feasibility heuristics.

We work with the fast heuristics on $\bar{G}$, such that the heuristic solutions satisfy all the constraints in the proposed IP model. Here, the heuristic solutions correlated to the decision variables: U and D are presented as U^0 and D^0.

5.1 Approach the independent optimal paths

We make the initial robot paths approach the independent optimal paths. We find the initial configuration $\bar{s}_r^0$ and goal configuration $\bar{g}_r^0$ on $\bar{G}$, which is the closest intersection vertices to s_r and g_r, i.e., $\bar{s}_r^0 = \arg\min_{v \in \bar{V}} (dist(v, s_r))$, $\bar{g}_r^0 = \arg\min_{v \in \bar{V} \backslash \{\bar{s}_r^0\}} (dist(v, g_r))$.

Firstly, inspired by [18], we compute independent optimal paths $\bar{P}^{opt}$ on $\bar{G}$ using Dijkstra algorithm [19] for each robot from $\bar{s}_r^0$ to $\bar{g}_r^0$, ignoring the robot collisions.

Secondly, according to $\bar{P}^{opt}$, we denote the direction of each robot r traversing each lane $\{v_i, v_j\}$, $(v_i, v_j), (v_j, v_i) \in \bar{E}$: $A_r(\{v_i, v_j\})$. v_i and v_j are the two vertices which bound the lane, $|v_i| < |v_j|$. The value of $A_r(\{v_i, v_j\})$ is decided by the three cases: robot does not pass the lane, $A_r(\{v_i, v_j\}) = 0$; robot traverses from v_i to v_j, $A_r(\{v_i, v_j\}) = 1$; robot traverses from v_j to v_i: $A_r(\{v_i, v_j\}) = -1$.

We can concatenate all the robots' A_r into a numerical matrix $A : |R| \times \frac{1}{2}|\bar{E}|$, which is called the robots' independent direction matrix. The independent direction matrix reflects robots' motions in the independent optimal paths.

Thirdly, we set the $\frac{1}{2}|\bar{E}| \times 1$ decision variables $D^0(\{v_i, v_j\}) \in \{-1, 1\}$ which is the initial solution to $D(\{v_i, v_j\})$ for each lane. The objective function for stage 2 can then be derived as follows to make the initial lane direction as close to the independent optimal paths as possible:

$$\min(\sum_{r \in R} A_r \cdot D^0).$$

5.2 Feasible direction optimization

To make the fast heuristics feasible, two cases of feasible constraints are proposed to keep $\bar{G}$ connected.

(1) **Vertex degree constraints**: To make the vertices be accessible, we restrict the degree of each vertex $v \in \bar{V}$: $deg^-(v) \neq 0 \wedge deg^+(v) \neq 0$. $deg^-(v)$ and $deg^+(v)$ represents the indegree and outdegree of v, respectively. The directions of lanes $D^0(\{v_i, v\}), v_i \in \bar{N}(v)$ correlated to the vertex are restricted.

(2) **Lane direction constraints**: For each pair of opposite lanes on $\bar{G}$: $\{v_i, v_j\}$ and $\{v_m, v_n\}$, for example, $\{v_1, v_5\}$ and $\{v_{19}, v_{23}\}$, $\{v_1, v_{19}\}$ and $\{v_5, v_{23}\}$ in Fig. 2, the lane direction constraint should be respected: if $\{v_i, v_j\}, \{v_m, v_n\}$ are opposite, then $D^0(\{v_i, v_j\}) + D^0(\{v_m, v_n\}) = 0$. Under such a case, we restrict that the initial lane direction alternates from one row (or column) to the next.

The small-scale IP model is then formulated using the feasible constraints above. We note that there is always a feasible solution for the small-scale IP model, e.g., Fig. 4 in [20].

[1] `https://www.bilibili.com/video/BV1pr4y1D7Dy/`.

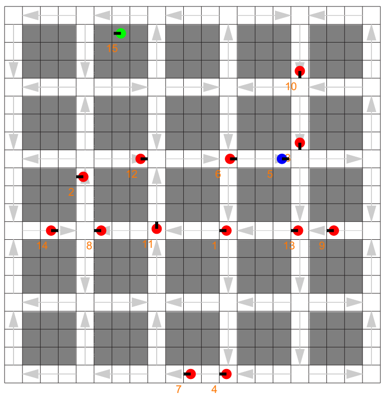

Fig. 4: The 21×21 narrow warehouse environment with 3×3 pod clusters.

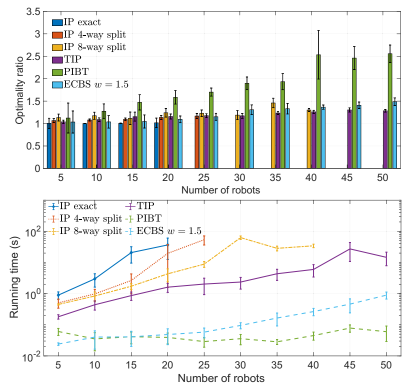

Fig. 5: We used the MPP algorithms to solve Min

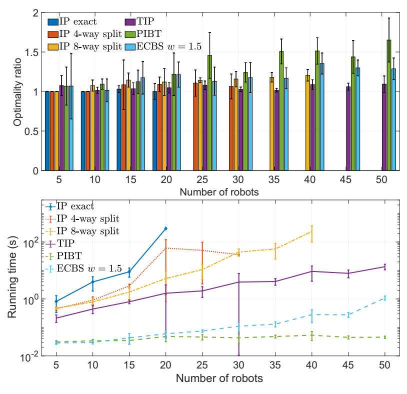

Fig. 6: We used the MPP algorithms

5.3 Feasible heuristic solution

Here, a directed map $\bar{G}_d = (\bar{V}, \bar{E}_d)$ is provided based
D^0 and $\bar{G}$, such that only one between (v_i, v_j) and $(v_j,$
exists in $\bar{E}_d$. Because feasible constrains in the small-sc
IP model ensure that $\bar{G}_d$ be connected, initial paths $\bar{P}^0$ fr
$\bar{X}_S$ and $\bar{X}_G$ can be searched easily using A^* [21] or Di
stra's search [19] on $\bar{G}_d$, satisfying all the constraints.

Finally, according to the bijection between the binary
cision variable set U and all paths $\bar{P}$ from $\bar{X}_S$ to $\bar{X}_G$, ini
solution U^0 can be derived from $\bar{P}^0$ with initial solution D^0 for all lane directions D.

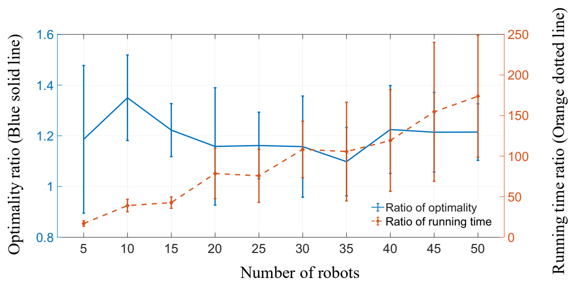

Fig. 7: The result of comparing the fast feasibility heuristics with Gurobi's general built-in feasibility heuristics.

6 Simulations and results

We evaluate the TIP algorithm to demonstrate its efficiency. Part of the results is given in this section.

6.1 Experiment setup

We build our simulation platform in MATLAB 2021a with the MATLAB-Gurobi interface: Yalmip[22]. Concerning IP exact and k-way split (optimal and sub-optimal MPP algorithm), we use the existing implementation by [11] formulated in Java and solved by Gurobi solver (Academic Version 9.1.1). We adopt the PIBT method (MPP algorithm with polynomial complexity) by [10] and Enhanced CBS (fast sub-optimal MPP algorithm) by [12] written in C++. All simulations are executed on a server equipped with an Intel(R) Xeon(R) CPU E5-2660 v4 with 128GB RAM at 2.00GHz. In all the simulations, the start and goal configurations are uniformly randomly sampled in the narrow warehouse environment with 21×21 grids and 3×3 clusters. As shown in Fig. 4, red robots are moving, blue robots are waiting, and the green robot has reached its goal. Grey arrows indicate the lanes' directions. Black lines on the robots indicate their moving directions. The solving efficiency of TIP algorithm is shown in Fig. 5 and Fig. 6.

6.2 Performance of fast feasibility heuristics

We use fast feasibility heuristics to operate the IP model's solving process with a feasible warm start. Here, we compare the proposed fast feasibility heuristics with Gurobi's general built-in feasibility heuristics in terms of efficiency. We use two heuristics to solve the same MPP problem in narrow warehouse environments and compare their running time and optimization performance. In particular, we obtain the Gurobi's feasibility heuristics solution by setting the optimality gap value to 100%, i.e., once Gurobi's feasibility heuristics find the feasible solutions which satisfy all the constraints, the Gurobi solver returns it without optimization. Here, the optimality ratio is measured as the solution cost using Gurobi's feasibility heuristics over the solution cost using the proposed fast feasibility heuristics. The running time ratio is measured as the running time of Gurobi's feasibility heuristics over the running time of the proposed fast feasibility heuristics.

As shown in Fig. 7, the proposed fast feasibility heuristics always generate better solutions than Gurobi's built-in feasibility heuristics, using less running time. The fast feasibility heuristics is about hundreds of times faster than Gurobi's built-in feasibility heuristics when the number of robots is large. The results suggest that using the proposed fast feasibility heuristics generates better initial solutions more efficiently than using Gurobi's built-in feasibility heuristics in the proposed problem settings.

6.3 Performance of TIP algorithm

We conducted simulations of solving the MPP problem in the narrow warehouse environment. We compare our proposed TIP algorithm with other algorithms, which are, to the best of our knowledge, some of the fastest (sub-)optimal or polynomial-complexity solvers for the MPP problem: IP exact, IP 4-way split, IP 8-way split, ECBS, and PIBT. Notably, for ECBS, we set its weight parameter $w = 1.5$ because it seems to be a good balance between optimality and scalability, in the original publication [12] and algorithm comparisons in [13] and from our observation.

The optimality ratio is measured as the solution cost over an underestimated cost. An underestimated cost is the solution cost of the path set, ignoring the robot-robot collision.

Solving MinTotalDist problem: Under the case of the MinTotalDist problem, we compare the TIP algorithm with five other existing MPP algorithms. For this case, between 5 and 50 robots are attempted in the 21×21 narrow warehouse environment. The evaluation results are shown in Fig. 5. Here, IP exact, IP 4-way split, and IP 8-way split algorithm can no longer finish each and every calculation in 10^4 seconds after the number of robots exceeds 40. The evaluation results show that the TIP algorithm is the most scalable among the six in the narrow warehouse environment. In particular, when the number of robots grows to 30, the TIP algorithm is about 10 times faster than the IP 8-way split and holds a better optimality ratio. Compared with the fast MPP algorithms ECBS with $w = 1.5$ and PIBT, the TIP algorithm always maintains a better optimality ratio by sacrificing part of the solving speed.

Solving MinMaxDist problem: In the second evaluation, we switch to the case of solving the MinMaxDist problem. In the previous evaluation, between 5 and 50 robots are attempted in the same environment. The evaluation results are shown in Fig. 6, which show similar performance trends as Fig. 5 of the MinTotalDist cases. Compared with the other IP-based algorithms, the TIP algorithm provides more than 10 times speed up and even offers better solutions than the IP 4-way split and IP 8-way split. Compared with the fast algorithms PIBT and ECBS $w = 1.5$, the TIP algorithm provides better solutions with a much lower optimality ratio, although the number of robots grows.

All tests are repeated with varied start and goal configurations. The results, which are omitted due to space constraints, are consistent in all cases.

7 Conclusion

This work provides the TIP algorithm to tackle the MPP problem in narrow warehouse environments, guaranteeing collision-free paths. The proposed algorithm achieves the trade-off between computational efficiency and optimality performance. By simulation results, it has been shown that the TIP algorithm can bring better performance compared with existing algorithms such as PIBT, ECBS, IP exact, and IP k-way split in the proposed environment. While the initial iteration of the TIP algorithm shows promising performance by simulation results, we want to provide theoretical completeness and optimality guarantee for the TIP algorithm in future work. Also, the TIP algorithm with the dynamic one-way constraints may further boost performance in solving the MPP problem in more complex situations.

References

[1] P. R. Wurman, R. D'Andrea, M. Mountz, Coordinating hundreds of cooperative, autonomous vehicles in warehouses, AI magazine 29 (1) (2008) 9–9.

[2] J. Yu, Intractability of optimal multirobot path planning on planar graphs, IEEE Robotics and Automation Letters 1 (1) (2015) 33–40.

[3] Z. Chen, J. Alonso-Mora, X. Bai, D. D. Harabor, P. J. Stuckey, Integrated task assignment and path planning for capacitated multi-agent pickup and delivery, IEEE Robotics and Automation Letters 6 (3) (2021) 5816–5823.

[4] R. Luna, K. E. Bekris, Network-guided multi-robot path planning in discrete representations, in: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2010, pp. 4596–4602.

[5] M. Erdmann, T. Lozano-Perez, On multiple moving objects, Algorithmica 2 (1) (1987) 477–521.

[6] S. M. LaValle, S. A. Hutchinson, Optimal motion planning for multiple robots having independent goals, IEEE Transactions on Robotics and Automation 14 (6) (1998) 912–925.

[7] M. R. K. Ryan, Exploiting subgraph structure in multi-robot path planning, Journal of Artificial Intelligence Research 31 (2008) 497–542.

[8] T. Schouwenaars, B. De Moor, E. Feron, J. How, Mixed integer programming for multi-vehicle path planning, in: 2001 European control conference (ECC), IEEE, 2001, pp. 2603–2608.

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

[10] K. Okumura, M. Machida, X. Défago, Y. Tamura, Priority inheritance with backtracking for iterative multi-agent path finding, arXiv preprint arXiv:1901.11282 (2019).

[11] J. Yu, S. M. LaValle, Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics, IEEE Transactions on Robotics 32 (5) (2016) 1163–1177.

[12] M. Barer, G. Sharon, R. Stern, A. Felner, Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem, in: Seventh Annual Symposium on Combinatorial Search, Vol. 5, 2014, pp. 19–27.

[13] S. D. Han, J. Yu, Ddm: Fast near-optimal multi-robot path planning using diversified-path and optimal sub-problem solution database heuristics, IEEE Robotics and Automation Letters 5 (2) (2020) 1350–1357.

[14] K. Wang, A. Botea, Fast and memory-efficient multi-agent pathfinding, in: ICAPS, 2008, pp. 380–387.

[15] Y. Chen, F. Wang, Y. Ma, Y. Yao, A distributed framework for solving and benchmarking security constrained unit commitment with warm start, IEEE Transactions on Power Systems 35 (1) (2019) 711–720.

[16] Gurobi Optimization, LLC, Gurobi Optimizer Reference Manual, `https://www.gurobi.com` (2021).

[17] V. Nguyen, P. Obermeier, T. C. Son, T. Schaub, W. Yeoh, Generalized target assignment and path finding using answer set programming, in: Twelfth Annual Symposium on Combinatorial Search, 2019, pp. 1216–1223.

[18] S. D. Han, J. Yu, Integer programming as a general solution methodology for path-based optimization in robotics: Principles, best practices, and applications, in: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2019, pp. 1890–1897.

[19] E. W. Dijkstra, A note on two problems in connexion with graphs, Numer. Math. 1 (1) (1959) 269–271.

[20] L. Cohen, T. Uras, S. Koenig, Feasibility study: Using highways for bounded-suboptimal multi-agent path finding, in: International Symposium on Combinatorial Search, Vol. 6, 2015, pp. 2–8.

[21] P. E. Hart, N. J. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths, IEEE transactions on Systems Science and Cybernetics 4 (2) (1968) 100–107.

[22] J. Lofberg, Yalmip: A toolbox for modeling and optimization in matlab, in: 2004 IEEE international conference on robotics and automation (IEEE Cat. No. 04CH37508), IEEE, 2004, pp. 284–289.