2022 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM) | 978-1-6654-1308-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/AIM52237.2022.9863310

Multi-robot Path Planning Algorithm in Dense Environments Using Particular Collision-free Traffic Rules

Jiaxi Huo[1], Ronghao Zheng[1,2,†], Senlin Zhang[1,2] and Meiqin Liu[3,2]

***Abstract*— Scheduling collision-free paths for a large number of robots in dense environments with high efficiency is achieved in this work. We propose an algorithm, OMPP (One-way Multi-robot Path Planning), using a new topological skeleton representation of the dense environment by introducing the particular collision-free traffic rules. We propose the integer programming technique based on the topological skeleton graph to tackle the multi-robot path planning optimization problem using distance metrics. We realize two practical achievements in solving multi-robot path planning problems in dense environments: collision-free robotic path generation and an efficient solving process. We have performed numerous simulations. According to the extensive simulation data, our algorithm suggests a higher overall performance in dense environments than the existing representative algorithms.**

I. INTRODUCTION

Nowadays, Multi-Robot System (MRS) has been used in various real scenarios to improve work efficiency. For example, automated logistics warehouses can use MRS to efficiently carry out the goods-transfer tasks [1]. As an essential part of the MRS studies, the Multi-robot Path Planning (MPP) problem has been explored for decades though researchers find that solving the optimization problem of MPP is intractable [2].

We consider in this paper a dense environment, e.g., an automated logistics warehouse where all the lanes are narrow. Narrow lanes mean that every two robots cannot move side by side. We aim to find a safe path for each robot, moving it from the start position to the goal position without inter-robot collision. Studies such as [3] have pointed out that lack of free space in the narrow lanes for robots causes challenging inter-robot coordination, which always leads to inter-robot collisions. Based on the above bottleneck of solving MPP problems in dense environments, we propose a novel **OMPP** (One-way Multi-robot Path Planning) algorithm, which can efficiently solve the MPP problems in dense environments.

We consider in this paper a goods-transfer scenario using MRS in a dense environment, e.g., automated logistics warehouses, and we propose two essential cost-to-go optimization objectives based on the distance metrics: minimizing the total distance and minimizing the maximum (single-robot traveled) distance. It is worth noting that reducing total distance can be viewed as reducing total fuel consumption. Furthermore, minimizing maximum distance lowers every robot's fuel consumption.

[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 (Email: {jiaxi.huo, rzheng, slzhang, liumeiqin}@zju.edu.cn)

[†] Corresponding author

To avoid the inter-robot collision in narrow lanes, we design the OMPP algorithm with particular collision-free traffic rules on each narrow lane: one-way constraints. The formulated MPP problem is converted into an Integer Programming (IP) model by the collision-free one-way constraints. The present IP solvers can then solve the IP model to generate the optimal collision-free solution.

The contributions of this paper are as follows:

(1) We present a topological skeleton representation of the dense environment to reduce the MPP problem's scale.

(2) We derive an IP formulation under the particular traffic rules, simplify the inter-robot collision avoidance in dense environments and improve the solving efficiency.

(3) Through extensive simulations, we reveal that the OMPP algorithm scales well to many robots and produces high-quality robotic paths in dense environments compared with the existing MPP algorithms.

II. RELATED WORKS

In recent years, various MPP algorithms are proposed due to the MPP problem's numerous applications, e.g., Network Flow [4] and Conflict-Based Search (CBS) [5]. Considering the existing works, the motives for studying the MPP problem in dense environments and proposing the OMPP algorithm are as follows.

Firstly, due to the high computational complexity of the optimal MPP problem, the solving process of the MPP algorithm aims at giving optimal collision-free robotic paths that may be intractable even for a few robots [2]. Lack of free space caused by narrow lanes challenges the existing representative MPP algorithms. Complete optimal algorithms consume many computing resources [6] to deal with inter-robot collisions in narrow lanes leading to the lack of scalability. Sub-problem [7] and spatial/temporal [8] splitting heuristics are designed to improve the efficiency of the MPP algorithm, while deadlocks may occur due to the failure of collision avoidance in narrow lanes, which leads to the incompleteness of the algorithm. In terms of the trade-off between optimality and scalability, fast MPP algorithms have been introduced at the cost of optimality, Enhanced Conflict-Based Search (ECBS) algorithm [9] for example. We focus on solving the optimal MPP problem by reducing the problem's scale to balance the optimality and efficiency.

Secondly, the ideas of imposing particular traffic rules and creating a topological skeleton representation of a graph have been studied in path-planning researches. To avoid collisions in robot movement and accelerate the solving process, fixed lane direction is given to the graph [10], where the lane direction alternates from one lane to the next to avoid collisions in lanes.

For the safety and agility of the robot in narrow corridors, a topological skeleton-based approach is introduced in [11] to solve the single robot global path planning problem. We adopt the topological skeleton-based approach to the multi-robot cases in dense environments.

Motivated by the works above, the IP-based optimized direction assignment and topological skeleton representation are studied here for MPP problems in a dense environment.

III. PROBLEM FORMULATION

A. Dense environment settings

We consider a goods-transfer scenario in a dense environment. The lanes in the dense environment are narrow, such that every two robots cannot move side by side (e.g., Fig. 1). The grey and white cells represent occupied blocks and passable lanes, respectively. Here we take automated logistics warehouses with narrow lanes as an example. Shelves occupy most cells in blocks to maximize the storage capacity. Each robot is small enough to go under the shelf in the convex occupied blocks and carry out the shelf with goods. We build the environment (Fig. 1a) using GAZEBO [12]. We use a grid graph (Fig. 1b) here to represent such a dense environment.

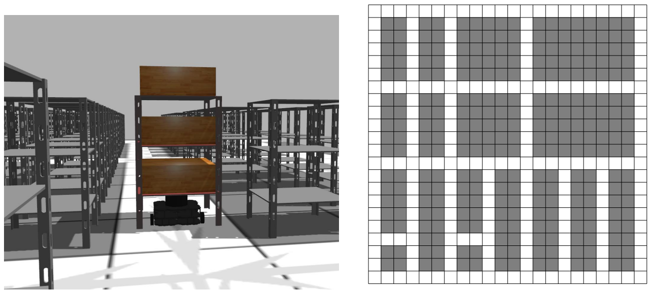

(a) A robot carries a shelf with goods. (b) Simulated grid graph.

Fig. 1: Visualization of dense environment settings.

B. Multi-robot path planning problem

Using the grid graph above, we suppose a binary matrix M of dimension $h \times w$ to represent the cell occupancy of the grid graph. We use $M(i_1, i_2) = 1$ to represent the occupied grid cell (i_1, i_2) and $M(i_1, i_2) = 0$ to represent the free grid cell for passable lanes whose width equals to one cell. Let $G = (V, E)$ be a bidirectional grid graph, with $V := \{(i_1, i_2) | M(i_1, i_2) = 0\}$ being the vertex set. Each free cell corresponds to a vertex. For each $(i_1, i_2) \in V$, its neighborhood is $N\big((i_1, i_2)\big) := \{(i_1+1, i_2), (i_1, i_2+1), (i_1-1, i_2), (i_1, i_2-1)\} \cap V$ using *4-way neighbourhood*. Each vertex v_i whose Cartesian coordinates are (i_1, i_2) in V has its corresponding index $index(v_i) := i_2 + (i_1 - 1)w$. Here, $E := \{(v_i, v_j) | v_i \in V \wedge v_j \in N(v_i)\}$ is defined as the edge set. Let $R := \{1, \ldots, n\}$ be a set of n robots. A *configuration* of the robots is an injective mapping from R to V: the initial and final positions we define in the task assignment before the algorithm execution are denoted as X^I and X^F, we suppose that $\forall r \in R$, $X^I(r) \neq X^F(r)$. We use T_r to represent the task completion time of robot r, and all the robots depart simultaneously. For each robot $r \in R$, a scheduled path is defined as a sequence of vertices $P_r := \{p_r(0), p_r(1), \ldots, p_r(T_r)\}$, $P_r \subset V$:

1) $p_r(0) = X^I(r)$, $p_r(T_r) = X^F(r)$;
2) $\forall t, 1 \le t \le T_r$, $p_r(t) \in \{p_r(t-1)\} \cup N(p_r(t-1))$.

Between time step $t-1$ and t, robot r occupies $\big(p_r(t-1), p_r(t)\big) \in E$. The scheduled path set is $P := \{P_1, \ldots, P_n\}$. Two paths $P_r, P_{r'}$ are in collision if there exists $t \in \mathbb{N}$ such that $p_r(t) = p_{r'}(t)$ (collision on vertices) or $(p_r(t-1), p_r(t)) = (p_{r'}(t), p_{r'}(t-1))$ (head-on collision). A path P_r is *collision-free* if no other $P_{r'}$ is in collision with P_r.

***Problem** 1: MPP problem in grid graph:* Given a 3-tuple $\langle G, X^I, X^F \rangle$, find a path set $P = \{P_1, \ldots, P_n\}$ such that for all $P_r \in P$ satisfying: P_r is collision-free.

In the goods-transfer scenario, a robot carrying a shelf with goods starts from an occupied cell. Once leaving the occupied cell, it enters a cell in a narrow lane, which is $X^I(r)$. When robot r arrives at $X^F(r)$, it goes into a nearby occupied cell, i.e., goes into a pre-occupied block to unload the shelf with goods and disappears from the lanes. Considering such a scenario of the dense environment, similar to the sense of [13], we give the well-formed infrastructure assumption:

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

Then, we introduce the objectives based on distance metrics. Let $P = \{P_1, \ldots, P_n\}$ be a collision-free solution to some fixed MPP instance. For $P_r \in P$, let $len(P_r)$ denote the travel distance of robot r, which is the total number of times that the robot r changes its residing vertex while following P_r.

Objective 1: **MinTotalDist (Minimize Total Distance)**: Compute P that minimizes: $\sum_{r \in R} len(P_r)$.

Objective 2: **MinMaxDist (Minimize Maximum Distance)**: Compute P that minimizes: $\max_{r \in R} len(P_r)$.

We present the OMPP algorithm to optimize the objectives when solving the MPP problem in the dense environment.

IV. GENERAL METHODOLOGY

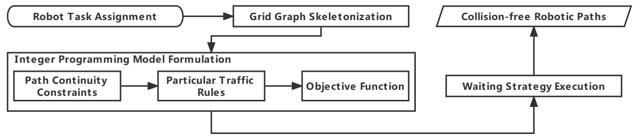

Robot Task Assignment
Grid Graph Skeletonization
Collision-free Robotic Paths
Integer Programming Model Formulation
Path Continuity Constraints
Particular Traffic Rules
Objective Function
Waiting Strategy Execution

Fig. 2: OMPP algorithm's flowchart.

Inspired by the corridor node combination in [14] and graph skeletonization for single robot path planning in [11], the flowchart of the OMPP algorithm (see Fig. 2): (A) the original grid graph is represented as a topological skeleton graph with fewer vertices and edges using particular traffic rules; (B) the optimal robotic paths are developed using the topological skeleton graph by formulating and solving the IP model; (C) the paths are transformed into the actual paths.

The three steps are feasible as vertices in narrow lanes in a dense environment do not have full connectivity. This allows for vertices whose connectivity degree is less than or equal to 2 to be skeletonized into a meta vertex under traffic rules (e.g., one-way). Here, the connectivity degree of a vertex in a narrow lane equals 2 because it only connects to the other two neighbor vertices. Intuitively, the solving time of an IP model is often negatively correlated with the number of decision variables.

This suggests that the IP model's solving time may be reduced if the grid graph can be skeletonized. Thus, we propose the following lemma to avoid the inter-robot collision caused by two opposing robots in narrow lanes and make the grid graph skeletonization reasonable.

***Lemma** 1:* One-way constraints: A collision-free path set P obeying one-way constraint requires: for all $i, j \in R, T_i \in \mathbb{N}$, if edge $(p_i(t), p_i(t+1))$ exists in any P_i, the reversed edge $(p_i(t+1), p_i(t))$ cannot exist in any other P_j. The inter-robot collision caused by two robots moving in opposite directions is avoided in narrow lanes if all robots obey one-way constraints, i.e., each lane is restricted to be one-way.

Proof: By contradiction, if $(p_i(t+1), p_i(t))$ also exists in another path P_j, such that $(p_j(t_j), p_j(t_j+1))$ exists in P_j with $p_j(t_j) = p_i(t+1), p_j(t_j+1) = p_i(t)$, it is always possible to generate collision between P_i and P_j by changing t_j to t (e.g., robot j's velocity changes unexpectedly), lead to the head-on collision: $p_j(t) = p_i(t+1), p_j(t+1) = p_i(t)$, i.e., $(p_j(t), p_j(t+1)) = (p_i(t+1), p_i(t))$. Collisions in vertices can also be generated similarly. It does not accord to the existence of collision-free path set. So Lemma 1 holds. ■

A. *Grid graph skeletonization*

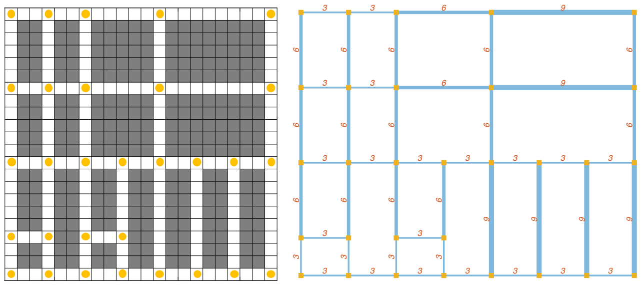

(a) Skeleton points insertion. (b) Topological skeleton graph.

Fig. 3: Illustration of the skeletonization process.

Inter-robot collisions caused by two robots moving in opposite directions in narrow lanes are avoided as a direct result of Lemma 1. As a robot's moving direction in the narrow lane is decided by the robot's occupancy in lane's ends, we skeletonize the grid graph into a topological skeleton graph.

To skeletonize the grid graph, we introduce the approach: *skeleton points insertion*. We insert the skeleton points into the grid graph to generate the topological skeleton graph.

Firstly, the skeleton point is inserted in each lane intersection, such as the orange points in Fig. 3a. We make sure that all the lanes be bounded by two inserted skeleton points.

Secondly, in Fig. 3b, we add bidirectional edges between two adjacent skeleton points following the corresponding actual lane. The edge's thickness is proportional to its length which is indicated by the red Arabic numbers.

Finally, we obtain the topological skeleton graph skeletonized from the grid graph. Each inserted skeleton point corresponds to a vertex in the topological skeleton graph.

***Definition** 1:* The topological skeleton graph associated with $G = (V, E)$: $G_s = (V_s, E_s)$ whose vertex set V_s contains only the skeleton vertices. For $v_i \in V_s$, let $N_s(v_i) := \{v_j | (v_i, v_j) \in E_s\}$ be the neighborhood of v_i in G_s. Each edge $(v_i, v_j) \in E_s$ has its correlated weight $W((v_i, v_j))$.

We suppose that the Cartesian coordinates and indices of v_i, v_j in G_s are (i_1, i_2), (j_1, j_2) and $i_2 + (i_1 - 1)w$, $j_2 + (j_1 - 1)w$, respectively. The edge weight is derived: $W((v_i, v_j)) := |i_1 - j_1| + |i_2 - j_2|$. Here, we note that $G_s = (V_s, E_s)$ is a bidirectional graph, such that, $(v_i, v_j) \in E_s$, $(v_j, v_i) \in E_s$. We define formally the correspondence between lanes in the grid graph and the edges in the topological skeleton graph:

***Definition** 2:* Given a lane $[v_i, v_j]$ in the grid graph, which is bounded by two skeleton vertices v_i, v_j, is correlated to bidirectional edges in E_s: (v_i, v_j) and (v_j, v_i).

X^I and X^F are known to be placed within the lanes. To make the robots traverse them by one-way constraints, we set their corresponding mirror vertices in G_s as X_s^I and X_s^F:

$$X_s^I(r) := \{X_{s,1}^I(r), X_{s,2}^I(r)\},\ X_s^F(r) := \{X_{s,1}^F(r), X_{s,2}^F(r)\}.$$

We suppose that $\forall r \in R$, $X^I(r)$ and $X^F(r)$ are within the lane $[X_{s,1}^I(r), X_{s,2}^I(r)]$ and the lane $[X_{s,1}^F(r), X_{s,2}^F(r)]$ respectively. Each robot r should pass both $X_{s,1}^I(r)$, $X_{s,2}^I(r)$ and $X_{s,1}^F(r)$, $X_{s,2}^F(r)$, such that $X^I(r)$ and $X^F(r)$ can be traversed in order. We use T_r^s as the path length of robot r in G_s.

***Problem** 2: MPP problem in topological skeleton graph:* Given a 4-tuple $\langle G_s, X_s^I, X_s^F, W\rangle$, find $P^s := \{P_1^s, \ldots, P_n^s\}$, $\forall r \in R$, $P_r^s := \{p_r^s(0), p_r^s(1), \ldots, p_r^s(T_r^s)\}$, $P_r^s \subset V_s$. The following requirements should be satisfied for each robot $r \in R$:

(1) $\forall t, 1 \le t \le T_r^s$, $p_r^s(t) \in N_s(p_r^s(t-1))$;

(2) $\{p_r^s(0), p_r^s(1)\} = X_s^I(r)$, $\{p_r^s(T_r^s - 1), p_r^s(T_r^s)\} = X_s^F(r)$.

We solve Problem 2 at first and each P_s is associated with P_r. We assume that each robot has a virtual path P_r^s from $X_s^I(r)$ to $X_s^F(r)$, then we convert each P_r^s to P_r linearly and all the robots can travel safely from X^I to X^F without loss of generality, Problem 1 can finally be solved. The number of decision variables depends on the number of robots and basic skeleton gadgets. Thus, the problem's scale is reduced significantly due to the grid graph skeletonization.

B. *Integer programming model formulation*

Multiple studies show that the MPP problem can be rewritten as a linear formulation with integer constraints that count for path continuity and collision avoidance [6], [7], [15]. A vital benefit of this IP approach is that the MPP problem can be readily solved using Gurobi [16] with MATLAB [17] and MATLAB-Gurobi interface: YALMIP [18]. We formulate here our IP model from three parts to solve Problem 2: path continuity constraints, particular traffic rules and objective functions.

1) Path continuity constraints: Given $\langle G_s, X_s^I, X_s^F\rangle$, we firstly introduce a binary decision set $u_{r,e}$ for each robot r and each edge $e \in E_s, e := (v_i, v_j)$ to indicate whether P_r^s uses e, i.e., if robot r uses e in G_s, then $u_{r,e} = 1$; otherwise, $u_{r,e} = 0$. We now define the in-degree and out-degree of vertex v_i of each robot's path in G_s to reduce the redundancy:

$$out(r, v_i) := \sum_{v_j \in N_s(v_i)} u_{r,(v_i,v_j)},\ in(r, v_i) := \sum_{v_j \in N_s(v_i)} u_{r,(v_j,v_i)}.$$

Above all, to satisfy the requirement (1) in Problem 2, the global path continuity constraint (1) is respected so that vertices

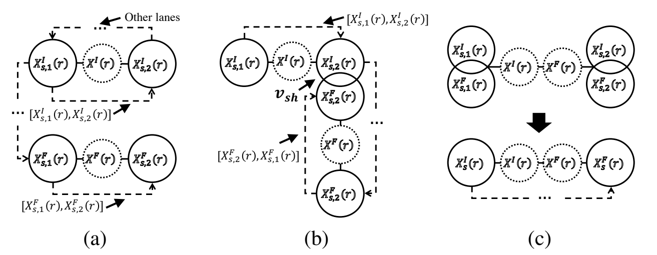

Other lanes
$[X^I_{s,1}(r),X^I_{s,2}(r)]$
$[X^F_{s,1}(r),X^F_{s,2}(r)]$
v_{sh}
$[X^F_{s,2}(r),X^F_{s,1}(r)]$
(a)
(b)
(c)

Fig. 4: Relative positions of $X^I(r), X^F(r)$ and $X^I_s(r), X^F_s(r)$, dotted directional connectors are examples of P^r_s in MPP instances. Two intersecting circles represent a common vertex.

in P^s_r are consecutive in G_s: $v_i = p^s_r(t), t < T^s_r$, v_j can be v_i's next vertex in P^s_r only if $v_j \in N_s(v_i)$. Constraint (2) shows that if $v_i \notin X^I_s(r) \cup X^F_s(r)$, v_i can be passed at most once, meanwhile, an edge leaves v_i only if an edge enters v_i, and if an edge enters v_i, there must be an edge leaves v_i.

$$\sum_{v_j \in V_s \setminus N_s(v_i)} u_{r,(v_i,v_j)} = \sum_{v_j \in V_s \setminus N_s(v_i)} u_{r,(v_j,v_i)} = 0. \tag{1}$$

$$v_i \notin X^I_s(r) \cup X^F_s(r),\ in(r,v_i) = out(r,v_i) \le 1. \tag{2}$$

For each $r \in R$, according to Fig. 4, three conditions are considered to satisfy the requirement (2) in Problem 2.

(a) Zero common vertex:

In Fig. 4a, vertices in $X^I_s(r)$ and $X^F_s(r)$ are in two disjoint lanes, i.e., $X^I_s(r)$ and $X^F_s(r)$ has no common vertex.

$$\begin{aligned} u_{r,\left(X^I_{s,2}(r),X^I_{s,1}(r)\right)} + u_{r,\left(X^I_{s,1}(r),X^I_{s,2}(r)\right)} &= 1, \\ u_{r,\left(X^F_{s,2}(r),X^F_{s,1}(r)\right)} + u_{r,\left(X^F_{s,1}(r),X^F_{s,2}(r)\right)} &= 1. \end{aligned} \tag{3}$$

Constraint (3) ensures that all robots pass both $X^I_s(r)$ and $X^F_s(s)$. We make sure the path continuity in $X^I_s(r)$ and $X^F_s(r)$:

$$\begin{aligned} out\left(r, X^I_{s,1}(r)\right) - in\left(r, X^I_{s,1}(r)\right) + u_{r,X^I_{s,2}(r),X^I_{s,1}(r)} &= 1, \\ out\left(r, X^I_{s,2}(r)\right) - in\left(r, X^I_{s,2}(r)\right) + u_{r,X^I_{s,1}(r),X^I_{s,2}(r)} &= 1, \\ \left|in\left(r, X^F_{s,1}(r)\right) - out\left(r, X^F_{s,1}(r)\right)\right| + u_{r,X^F_{s,1}(r),X^F_{s,2}(r)} &= 1, \\ \left|in\left(r, X^F_{s,2}(r)\right) - out\left(r, X^F_{s,2}(r)\right)\right| + u_{r,X^F_{s,2}(r),X^F_{s,1}(r)} &= 1. \end{aligned} \tag{4}$$

At most one vertex in each of $\{X^I_{s,1}(r), X^I_{s,2}(r)\}$ and $\{X^F_{s,1}(r), X^F_{s,2}(r)\}$ can be passed twice by constraint (4), e.g., $X^I_{s,1}$ is passed twice and $u_{r,\left(X^I_{s,1}(r),X^I_{s,2}(r)\right)} = 1$ in Fig. 4a.

(b) One common vertex:

In Fig. 4b, $X^I_s(r)$ and $X^F_s(r)$ are in two adjacent lanes connected by $v_{com} = \{X^I_{s,1}(r), X^I_{s,2}(r)\} \cap \{X^F_{s,1}(r), X^F_{s,2}(r)\}$.

Firstly, constraint (3) makes sure that P^s_r should pass both $X^I_s(r)$ and $X^F_s(r)$. Secondly, if $v_i = v_{com}$, v_i can be visited more than once in P^s_r when constraint (5) is satisfied.

$$|in(r,v_i) - out(r,v_i)| \le 1. \tag{5}$$

Finally, if $v_i \in X^I_s(r) \cup X^F_s(r)$, $v_i \ne v_{com}$, v_i can be passed only once by constraint (6). Thus, v_{com} is the only vertex can be visited more than once.

$$in(r,v_i) = out(r,v_i) = 1. \tag{6}$$

(c) Two common vertices:

In Fig. 4c, $X^I_s(r)$ and $X^F_s(r)$ bound the same lane, the lane is bounded by $v_i \in V_s$ and $v_j \in V_s$. We reuse $X^I_s(r), X^F_s(r)$ by the positions of $X^I(r)$ and $X^F(r)$, operator $dist(v_i, v_j)$ denotes the Manhattan distance between v_i and v_j:

$$X^I_s(r) := \underset{v \in \{v_i,v_j\}}{\arg\min} \left(dist\left(v, X^I(r)\right)\right),$$

$$X^F_s(r) := \underset{v \in \{v_i,v_j\} \setminus \{X^I_s(r)\}}{\arg\min} \left(dist\left(v, X^F(r)\right)\right).$$

$$out\left(r, X^I_s(r)\right) = in\left(r, X^F_s(r)\right) = 1. \tag{7}$$

$$\begin{aligned} \sum_{v_j \in N_s\left(X^I_s(r)\right) \setminus \{X^F_s(r)\}} u_{r,\left(v_j, X^I_s(r)\right)} &= 0, \\ \sum_{v_j \in N_s\left(X^F_s(r)\right) \setminus \{X^I_s(r)\}} u_{r,\left(X^F_s(r), v_j\right)} &= 0. \end{aligned} \tag{8}$$

$$u_{r,\left(X^I_s(r),X^F_s(r)\right)} + u_{r,\left(X^F_s(r),X^I_s(r)\right)} = 1. \tag{9}$$

Constraints (7)-(8) make sure that $X^I_s(r)$ and $X^F_s(r)$ be visited in order. Constraint (9) makes sure that robot pass the lane bounded by $X^I_s(r)$ and $X^F_s(r)$.

2) Particular traffic rules: By Lemma 1, one-way constraints are imposed on all the lanes to avoid the inter-robot collision caused by two robots moving in opposite directions. According to Definition 2, we use $[v_i, v_j]$ to represent the lane bounded by v_i and v_j. As we want to optimize the direction of each lane according to distance metrics, we introduce a decision variable: $D_{[v_i,v_j]}$, $(v_i,v_j),(v_j,v_i) \in E_s$ to represent the lane's direction. We then give the numerical description of lane's direction correlated to $[v_i,v_j]$, $index(v_i) < index(v_j)$:

1. The allowed lane direction is $v_i \to v_j$: $D_{[v_i,v_j]} = 1$;
2. The allowed lane direction is $v_j \to v_i$: $D_{[v_i,v_j]} = -1$.

The lane direction is restricted as: $\forall (v_i,v_j) \in E_s$, $D_{[v_i,v_j]} \in \{-1, 1\}$. We propose the one-way constraints as follows:

$$\forall (v_i,v_j) \in E_s, r \in R, \sum_{r \in R} u_{r,(v_i,v_j)} = 0 \vee \sum_{r \in R} u_{r,(v_j,v_i)} = 0. \tag{10}$$

$$\forall (v_i,v_j) \in E_s, r \in R, \left|u_{r,(v_i,v_j)} - u_{r(v_j,v_i)} - D_{[v_i,v_j]}\right| \le 1. \tag{11}$$

According to constraint (10), we prevent two robots from passing the same lane in opposite directions to avoid the inter-robot collision. Now we have the fact:

$$\nexists r_1, r_2 \in R, u_{r_1,(v_i,v_j)} = u_{r_2,(v_j,v_i)} = 1.$$

Constraint (11) derives the value of $D_{[v_i,v_j]}$: $\exists r \in R$, if $u_{r,(v_i,v_j)} = 1, u_{r,(v_j,v_i)} = 0$, then $D_{[v_i,v_j]}$ can only be 1, instead, if $u_{r,(v_i,v_j)} = 0, u_{r,(v_j,v_i)} = 1$, then $D_{[v_i,v_j]}$ can only be -1.

3) Objective functions: In this work, we focus on optimizing two objectives based on distance metrics.

Minimize Total Distance (MinTotalDist): For all $r \in R$, with its binary decision variable $u_{r,e}$ for each edge $e \in E_s$, the objective function can then be derived as:

$$\min \sum_{r \in R} \sum_{e \in E_s} W(e) u_{r,e}.$$

Minimize Maximum Distance (MinMaxDist): To minimize the maximum robot distance, we introduce an integer decision variable *umax* and for all $r \in R$, we add the upper bound constraint for each robot's distance and minimize it:

$$\forall r \in R, \sum_{e \in E_s} W(e) u_{r,e} \leq umax(r); \ \min_{r \in R} umax(r).$$

Referring to Proposition III.1 in the paper [15], there exists a bijection between the solution of the decision variables $u_{r,e}$ and all robotic paths in P^s. We can derive each P^s_r easily from the solved $u_{r,e}$. Thus, Problem 2 is solved here.

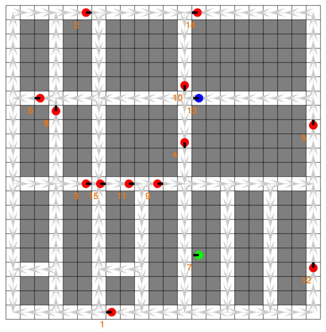
C. *Waiting strategy execution*

For each path P^s_r and each pair of $v_i, v_j \in P^s_r$, $(v_i, v_j) \in E_s$, vertices in $[v_i, v_j] \setminus V_s$ are inserted sequentially to convert P^s_r into P_r, $[v_i, v_j]$ contains all the vertices in the lane bounded by v_i and v_j. As the lanes are one-way and $X^I_s(r)$ and $X^F_s(r)$ are both visited in order, $X^I(r)$ and $X^F(r)$ are already in P_r, we make them be the first and last vertex of P_r respectively by deleting the vertices before $X^I(r)$ and after $X^F(r)$.

Finally, since one-way constraints avoid the collision caused by two robots moving in opposite directions in narrow lanes, a ready-made conflict-based waiting strategy at the robotic level is executed to prevent real-time collisions (e.g., rear-end collisions within the narrow lane) during the task execution of robots. The details of waiting strategy are omitted due to the page limitation. The general performance can be seen in the video[1]. Problem 1 is eventually solved.

V. SIMULATION RESULTS

We have performed extensive simulations to showcase the proposed algorithm's performance in the dense environment.



MinTotalDist Problem

Algorithm \ Metrics	Total distance	Running time (s)
IP exact	NaN[1]	NaN
IP 8-way split	322	1.8454
ECBS $w = 1.5$	494	**0.0293**
OMPP	**314**	0.1103

MinMaxDist Problem

Algorithm \ Metrics	Maximum distance	Running time (s)
IP exact	NaN	NaN
IP 8-way split	**35**	1.7438
ECBS $w = 1.5$	43	**0.0275**
OMPP	38	0.1827

[1]NaN: The algorithm can not solve the MPP problem in 1000 seconds.

Fig. 5: We solve a 15-robot MPP instance in a 22×22 dense environment using different algorithms: Grey arrows indicate the optimized directions of lanes, red and blue circles represent the moving and waiting robots, and the green circle represents the robot that finishes its task.

A. *Simulation setup*

We implement the OMPP algorithm in MATLAB R2021a [17] and use Gurobi Solver (Academic Version 9.1.1) [16] to solve the IP model. Concerning IP exact and k-way split (optimal and sub-optimal MPP algorithm) based on network flow, we use the existing implementation by [7] written in Java. We adopt the ECBS algorithm (Enhanced CBS, fast sub-optimal MPP algorithm) by [9] written in C++. We set ECBS's weight parameter $w = 1.5$ because it seems to be a good balance between optimality and scalability in the original publication [9]. Meanwhile, for the IP k-way split, we set $k = 8$ since it seems to solve the MPP instances more efficiently in the dense environment, as we observe in [7].

[1]https://gitee.com/huo-jiaxi/ompp-simulation-video.git.

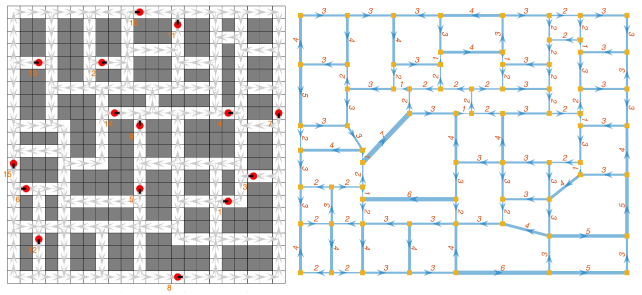


Fig. 6: One MPP instance in a randomly generated dense environment which is used in the simulation.

All experiments are executed on an Intel® Core$^{\text{TM}}$ CPU i7-10700 with 32GB RAM at 2.90GHz. We compare the OMPP algorithm with the above representative MPP algorithms in simulated dense environments. The simulation is repeated on diverse instances consisting of random X^I and X^F in the randomly generated 22×22 dense environments (the examples in Fig. 5 and Fig. 6 are part of instances). Ten tests are done to achieve the mean testing results and the standard deviation for each test point. We limit the maximum Gurobi's running time to 1000 seconds. We present *optimality ratio* which is measured as the solution cost over an underestimated cost. An underestimated cost is the solution cost of the path set ignoring the robot collision.

B. *MinTotalDist problem*

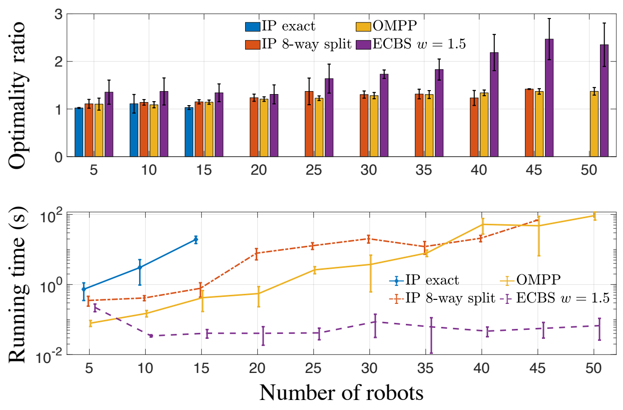


Fig. 7: Comparison of algorithms in MinTotalDist problem.

The general trend is found in Fig. 7 and the table in Fig. 5. We show that the OMPP algorithm can solve MinTotalDist problems with more and more robots without sacrificing much optimality. The proposed OMPP algorithm always solves the MPP instance more quickly than the IP 8-way split and provides high-quality solutions. Meanwhile, ECBS $w = 1.5$

always has better time performance at a significant optimality penalty, and in contrast, the OMPP algorithm yields better solutions in MinTotalDist using reasonable running time when there are more robots. Furthermore, IP exact achieves the lowest solution cost but takes a longer running time using many computational resources. As the number of robots increases, the IP exact and IP 8-way split can no longer finish calculation within the limited time, while the OMPP algorithm still works.

C. MinMaxDist problem

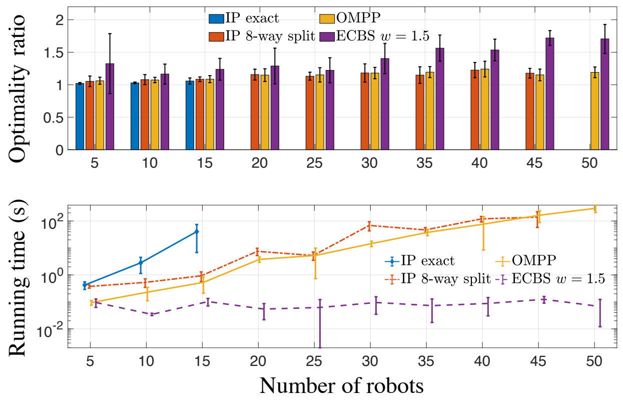


Fig. 8: Comparison of algorithms in MinMaxDist problem.

According to Fig. 8 and the table in Fig. 5, the OMPP algorithm operates more quickly than other IP-based algorithms, even though there are fewer robots in the dense environment. Meanwhile, the OMPP algorithm often yields better solutions using less running time than the IP 8-way split. The fast algorithm ECBS $w = 1.5$ solves the MinMaxDist instances more quickly at the great cost of optimality. The OMPP algorithm can still solve the instances with more than 45 robots while the other IP-based algorithms cannot.

Overall, the OMPP algorithm better balances optimality and running time than the other representative algorithms when solving the MPP problems in dense environments.

The above simulation results also show an interesting trend: As the number of robots keeps increasing, the OMPP algorithm's growth rate of running time decreases against IP exact. This makes sense because the OMPP algorithm's particular traffic rules simplify collision avoidance in the dense environment. Meanwhile, the searching space of IP exact rises rapidly as the number of robots rises. The omitted results due to space constraints are consistent in all cases.

VI. CONCLUSIONS

This work provides the OMPP algorithm to tackle the MPP problem in dense environments, guaranteeing collision-free paths. The proposed algorithm achieves the trade-off between computational efficiency and optimality performance when solving the MPP problem in dense environments. By simulation results, it has been shown that the OMPP algorithm can bring better performance compared with the existing algorithms such as IP exact, IP 8-way split, and ECBS $w = 1.5$ in the proposed dense environment. We want to adopt the proposed OMPP algorithm into the actual multi-robot operations in future work to demonstrate its performance.

ACKNOWLEDGMENT

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.

REFERENCES

[1] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," *AI magazine*, vol. 29, no. 1, pp. 9–9, 2008.
[2] J. Yu and S. M. LaValle, "Structure and intractability of optimal multi-robot path planning on graphs," in *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.
[3] S. D. Han and 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*, vol. 5, no. 2, pp. 1350–1357, 2020.
[4] J. Yu and S. M. LaValle, "Optimal multi-robot path planning on graphs: Structure and computational complexity," *arXiv preprint arXiv:1507.03289*, 2015.
[5] 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.
[6] T. Schouwenaars, B. De Moor, E. Feron, and J. How, "Mixed integer programming for multi-vehicle path planning," in *2001 European control conference (ECC)*. IEEE, 2001, pp. 2603–2608.
[7] J. Yu and S. M. LaValle, "Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1163–1177, 2016.
[8] T. Guo, S. D. Han, and J. Yu, "Spatial and temporal splitting heuristics for multi-robot motion planning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 8009–8015.
[9] M. Barer, G. Sharon, R. Stern, and A. Felner, "Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem," in *Seventh Annual Symposium on Combinatorial Search*, 2014.
[10] K.-H. C. Wang, A. Botea, *et al.*, "Fast and memory-efficient multi-agent pathfinding." in *Proceedings of the 2008 International Conference on Automated Planning and Scheduling (ICAPS)*, 2008, pp. 380–387.
[11] J.-w. Han, S. Jeon, and H.-J. Kwon, "A new global path planning strategy for mobile robots using hierarchical topology map and safety-aware navigation speed," in *2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE, 2019, pp. 1586–1591.
[12] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. IEEE, 2004, pp. 2149–2154.
[13] M. Čáp, J. Vokřínek, and A. Kleiner, "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*, 2015.
[14] V. Nguyen, P. Obermeier, T. C. Son, T. Schaub, and W. Yeoh, "Generalized target assignment and path finding using answer set programming," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, 2017, pp. 1216–1223.
[15] S. D. Han and 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.
[16] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2021. [Online]. Available: https://www.gurobi.com
[17] *MATLAB version 9.10.0.1613233 (R2021a)*, The Mathworks, Inc., Natick, Massachusetts, 2021.
[18] J. Lofberg, "Yalmip: A toolbox for modeling and optimization in MATLAB," in *Proceedings of the 2004 IEEE International Conference on Robotics and Automation (ICRA)*, 2004, pp. 284–289.