Intelligent Service Robotics (2022) 15:537–555
https://doi.org/10.1007/s11370-022-00436-4

ORIGINAL RESEARCH PAPER

Dual-layer multi-robot path planning in narrow-lane environments under specific traffic policies

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

Received: 16 February 2022 / Accepted: 15 July 2022 / Published online: 5 August 2022

Abstract
Collision-free path planning is indispensable for the multi-robot system. Many existing multi-robot path planning algorithms may no longer work properly in the narrow-lane environment. We propose in this paper a dual-layer algorithm to deal with the multi-robot path planning problem in the narrow-lane environment. In the first layer, the integer programming technique primarily based on distance metrics balances the optimality of the generated collision-free paths and the computation time of the algorithm. In the second layer, fast feasible heuristics are applied to make sure the solvability of the proposed integer programming approach in the first layer. In the dual-layer algorithm, specific traffic policies for each narrow lane are implemented to generate a collision-free path for every robot while maintaining the narrow lane free, besides the collision avoidance approach at the robotic level. With this, inter-robot collision in the narrow lane is avoided, and the algorithm's efficiency in producing collision-free paths increases. Simulations have been launched considerably based on the proposed assessment metrics. According to the extensive simulation data, our algorithm suggests a higher overall performance in the narrow-lane environment when in contrast with the present optimal, sub-optimal, and polynomial-complexity algorithms.

Keywords Multi robot systems · Path planning · Moving robots · Industrial automation · Optimization algorithms

1 Introduction

Multi-robot system (MRS) is now used to enhance the work efficiency in many scenarios. For example, robots can coordinate with each other to execute a complicated goods-transfer task in an automatic warehouse [1]. The studies of MRS, such as evacuation [2], formulation control [3,4] and micro-droplet manipulation [5], have been carried out for decades to explore the functionalities of MRS. As an essential part of the MRS studies, the Multi-robot Path Planning (MPP) problem has been explored for decades.

We consider the MPP problem in a goods-transfer scenario using MRS in a narrow-lane environment. The problem is applicable to the automated warehouse (e.g., the KIVA system of Amazon [6]). Goods are placed on shelves. The shelves can be placed on the occupied cells, which are available for shelves. The free cells are left empty for robots to move. Several occupied cells can be arranged together to form a cluster, and we call the cluster an "obstacle cluster". We then think about a goods-transfer task using a crew of mobile robots. When orders are received, robots take these shelves from the obstacle clusters and carry them to their goal positions. Robots can transfer shelves between obstacle clusters. Taking the KIVA system in Fig. 1a for example, during the task execution, every two robots cannot pass alongside each other because all the obstacle clusters are arranged with narrow lanes.

The inter-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 tough to deal with head-on collisions in a narrow lane

✉ Ronghao Zheng
rzheng@zju.edu.cn

Jiaxi Huo
jiaxi.huo@zju.edu.cn

Senlin Zhang
slzhang@zju.edu.cn

Meiqin Liu
liumeiqin@zju.edu.cn

[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

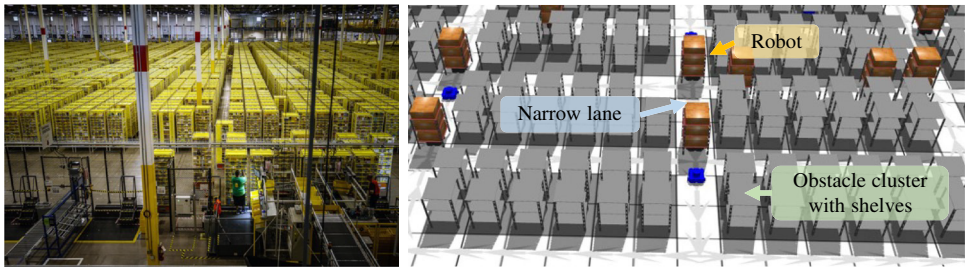


Fig. 1 **a** KIVA system by [7]. **b** Narrow-lane environment modeled by KIVA system

in the existing MPP algorithms due to the lack of free space. Thus, we propose the DLM (Dual-Layer Multi-robot path planning) algorithm, which targets solving the MPP problem in narrow-lane environments.

We set up the DLM algorithm with specific one-way traffic policies to realize the avoidance of inter-robot collisions in narrow lanes. We design the proposed DLM algorithm using two layers: *Layer 1*: the formulated MPP problem is converted into an integer programming (IP) modeled by specific one-way traffic policies, then the IP model can be solved by the existing IP solvers to generate the optimal collision-free solution; *Layer 2*: the fast feasibility heuristics primarily based on the IP model grant the high-quality initial feasible solution to the solving process of the first layer, which satisfies all the constraints in the IP model, replacing the IP solver's built-in feasibility heuristics and ensuring the solvability of the IP model.

To improve the work efficiency of MRS, we propose two essential optimization objectives primarily based on the distance metrics involved in solving the MPP problem: minimizing the total distance and minimizing the maximum (single-robot traveled) distance. It is quintessential to factor out that minimizing total distance can be viewed as reducing the fuel consumption of the whole system. Furthermore, minimizing maximum distance lowers each robot's fuel consumption. We summarize our main contributions in three folds:

1. We improve the efficiency of solving the MPP problem in narrow-lane environments by deriving an IP-based formulation. We use specific traffic policies to ensure inter-robot collision avoidance in narrow lanes.
2. We propose two fast feasibility heuristics to further improve the IP-based formulation's solving process. The proposed fast feasibility heuristics can provide the IP-based formulation with a high-quality initial feasible solution. We demonstrate that the two novel fast feasibility heuristics are more efficient than the IP solver's built-in feasibility heuristics in solving the MPP problem in narrow-lane environments.
3. Compared with the existing algorithms, we reveal by simulation that the DLM algorithm scales well to many robots and produces high-quality solutions in narrow-lane environments. Meanwhile, we use a warehouse environment simulation to verify the algorithm's performance.

2 Related works

In recent decades, many works inherited the existing single-robot path planning concepts (e.g., Dijkstra search [8], A^* search [9]) into solving the path planning problem with multiple robots, known as multi-robot path planning (MPP). The objective of MPP requires the robots to travel from their start positions, following a feasible path, avoiding every other robot and obstacle at any time, to their goal positions. Studies on the MPP problem have been carried out for decades [10–14].

Multiple algorithms are proposed to solve the MPP problem optimally at the global level. Surynek proved the NP-completeness of solving the optimal version of the MPP problem in the paper [15]. Schouwenaars et al. planned optimal multi-vehicle paths in the graph using Mixed Integer Programming (MIP) in the paper [16]. However, the MIP model's complexity may be high due to the robot interaction. Facing this problem, Yu et al. reduced the optimal MPP problem to network flow problems based on the graph in the paper [17]. The graph model's complexity can then be diminished compared to the MIP approach, and the problem can be modeled using Integer Programming (IP) approach. However, we can find in the original publication [17] that the algorithm based on network flow problems still has a long running time in an environment with many obstacles and robots. Based on the interaction between robots, Sharon et al. introduced the Conflict-Based Search (CBS) by inheriting the concept of single-robot A^* search to optimize the single robot path iteratively and resolve the real-time conflicts between robots in the paper [18]. However, since the interaction between robots is often complex, the search space will increase exponentially when there are more robots. Therefore, although the algorithm is optimal, it is still challenging in the case of many robots. Therefore, the solving process of the existing optimal MPP algorithms has the bottleneck of intractability.

Accordingly, fast polynomial-time MPP algorithms are proposed in practice at the cost of optimality when facing the intractability of the optimal MPP algorithm. A state-of-the-art iterative approach called Priority Inheritance with Backtracking (PIBT) was introduced in [19] by Okumura et al. This is a novel and decentralized fast algorithm that can solve MPP problems efficiently and has a polynomial time-complexity bound. However, the fast algorithm has no optimality guarantee.

Later, we study the existing sub-optimal algorithms introduced to balance running time and optimality. Besides the optimal IP approach using network flow formulation, Yu et al. also introduced the sub-optimal IP-based approaches with splits in the paper [17]. In this original publication of IP with splits (IP k-way split), as k increases, we observe a general trend of reduced running time at the expense of loss of optimality. In an environment with dense obstacles, the scalability advantage of the algorithm with a large split number is reflected. Another mentioned sub-optimal algorithm is called Enhanced Conflict-Based Search (ECBS), introduced by Barer et al. in the paper [20]. ECBS is a bounded sub-optimal variant of the popular optimal MPP algorithm CBS. ECBS uses focal search with a user-defined parameter w to obtain the trade-off between running time and optimality.

Based on the analysis, our motivations for studying the MPP problem in narrow-lane environments and proposing the DLM algorithm are as follows.

Firstly, narrow lanes in environments challenge the representative MPP algorithms. Complete optimal algorithms in [16,18] devour many computing resources to deal with inter-robot collisions in narrow lanes leading to the lack of scalability. To reduce the computational complexity, one can apply the decoupled algorithm and sub-problem splitting heuristics in optimal algorithms. We now take the recent paper [21] for example, Guo et al. applied spatial and temporal division to existing MPP algorithms, and the global MPP problem can be divided into multiple easily solved sub-problems. In such a case, the deadlocks tend to occur if particular sub-problems are unsolvable due to the failure of collision avoidance in narrow lanes. Han et al. in [22] has pointed out that they cannot perform multi-robot path planning in such a narrow-lane environment. This case leads to the incompleteness of the algorithm in the narrow-lane environment. However, as we mentioned above, fast MPP algorithms such as PIBT [19], and sub-optimal MPP algorithms such as ECBS [20] can achieve good running-time performance at the high cost of optimality. Thus, we propose the MPP algorithm, which focuses on solving the optimal MPP problem by reducing the problem's scale to balance the optimality and efficiency.

Secondly, the ideas of imposing specific one-way traffic policies and creating a topological skeleton representation of a graph have been studied in path-planning research. In order to restrict robot motion and avoid head-on robot collisions in the lanes, Wang et al. in the paper [23] introduced the flow restriction idea where passing directions restrict the robot moves in the lanes. In a similar sense, to enhance the optimal CBS algorithm with an alternative between optimality and efficiency, "highways" (a.k.a., specific one-way traffic policies) were therefore introduced in [24] by Cohen et al. Such an idea that imposes traffic policies on the lanes in the smart warehouse can also be seen in the paper [25] by Bolu et al. Inspired by the works above, the IP-based optimized direction assignment is studied in this paper as a specific traffic policy to avoid inter-robot collisions in narrow lanes.

Thirdly, fast feasibility heuristics (a.k.a., warm start of IP solving process) can accelerate the solution of an IP model and make sure the IP model always has feasible solutions by taking advantage of the initial solution given by fast feasibility heuristics. IP solvers like Gurobi [26] allow feeding in initial binary solutions through the "MIP start" setting. Ralphs et al. in the paper [27] proved the rationality of providing the IP solving process with a warm start. Furthermore, Chen et al. in the paper [28] demonstrated the overall performance of a warm start in the IP solving process.

Organization: The rest of this paper is organized as follows. Section 3 defines the MPP problem in narrow-lane environments with several essential assumptions. The first layer of the DLM algorithm with IP model formulation and specific traffic policies is given in Sect. 4. Section 5 describes the second layer of the DLM algorithm. Section 6 gives the evaluation results of the DLM algorithm along with the comparison between two proposed fast feasibility heuristics. Section 7 describes the DLM algorithm's warehouse environment simulation. Finally, the conclusion of this work is given in Sect. 8.

3 Problem model and algorithm overview

3.1 Multi-robot path planning problem in the narrow-lane environment

We represent the narrow-lane environment as a grid graph by discretizing the environment into a grid of atomic locations which are called cells. Here, we suppose a binary matrix M of dimension $h \times w$ to represent the occupancy of the grid graph. Each cell corresponds to a vertex v_i whose Cartesian coordinates are (i_1, i_2). Take the automated warehouse as an application of the narrow-lane environment, we use $M(i_1, i_2) = 0$ to represent that the corresponding cell is free. If $M(i_1, i_2) = 1$, the cell is occupied in advance, a shelf can be placed on such a cell. Several occupied cells can be arranged together and form an "obstacle cluster".

Let $G = (V, E)$ be a bidirectional graph which represents the narrow-lane environment, with $V := \{(i_1, i_2) | M(i_1, i_2) =$

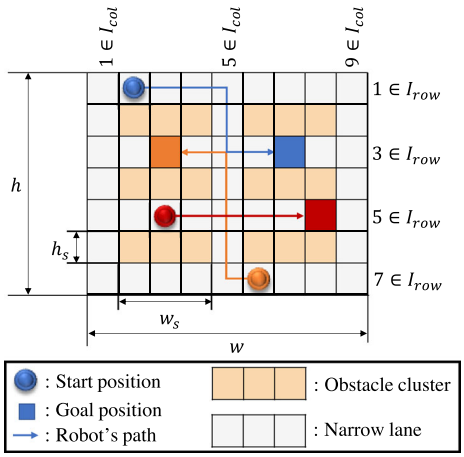


Fig. 2 A narrow-lane environment's diagram

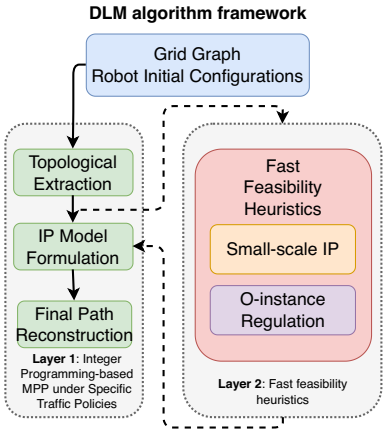


Fig. 3 Framework of the DLM algorithm

0} being the vertex set. Adjacency relationships are defined in four cardinal directions, i.e., for each $(i_1, i_2) \in V$, its neighborhood is $N((i_1, i_2)) := \{(i_1 + 1, i_2), (i_1, i_2 + 1), (i_1 - 1, i_2), (i_1, i_2 - 1)\} \cap V$. Here, each vertex v_i whose Cartesian coordinates are (i_1, i_2) in V has its corresponding index $|v_i| := i_2 + (i_1 - 1)w$. We then describe the narrow-lane environment formally in Assumption 1.

Assumption 1 (*Narrow-lane environment*) We assume that most cells in the environment are occupied by N_{oc} obstacle clusters. Some cells are left empty to form lanes to allow the robots to move. The width of all the narrow lanes equals one cell.

We can take the mimics typical narrow-lane environment in Fig. 2 for demonstration, to locate the vertices, we define the row and column index sets I_{row} and I_{col} as $I_{\text{row}} := \{\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_{\text{row}}\} \subset \{1, 2, \ldots, h\}$ and $I_{\text{col}} := \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_{\text{col}}\} \subset \{1, 2, \ldots, w\}$. Here both I_{row} and I_{col} are ordered sets, and neither I_{row} nor I_{col} is empty. The vertices in horizontal lanes V_{hl} and the vertices in vertical lanes

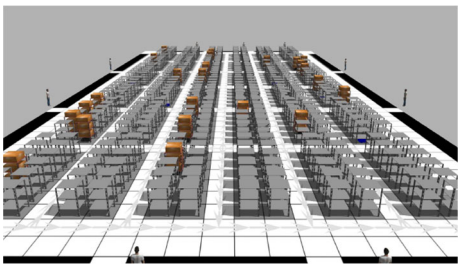


Fig. 4 Top view of the simulated narrow-lane automated warehouse

V_{vl} are represented as:

$$V_{hl} := \{(i_1, i_2) | i_1 \in I_{\text{row}}, i_2 \in \{1, 2, \ldots, w\} \backslash I_{\text{col}}\};$$
$$V_{vl} := \{(i_1, i_2) | i_1 \in \{1, 2, \ldots, h\} \backslash I_{\text{row}}, i_2 \in I_{\text{col}}\}.$$

We let $\{I_{\text{row}} \times I_{\text{col}}\}$ be the set of intersection vertex, which is the intersection of lanes. $V_{hl} \cup V_{vl}$ is presented for the set of vertices in the lanes. We let $M(i_1, i_2) = 0$ for all $(i_1, i_2) \in V_{hl} \cup V_{vl} \cup \{I_{\text{row}} \times I_{\text{col}}\}$, i.e., $V = V_{hl} \cup V_{vl} \cup \{I_{\text{row}} \times I_{\text{col}}\}$.

For the environment settings considered in this paper, the size of each obstacle cluster is fixed as $h_s \times w_s$, and each obstacle is in size of 1×1, such that each obstacle cluster contains $h_s w_s$ individual obstacles. We choose $h_s, w_s \in \mathbb{N}^+$ and $\mathcal{R}_1 = 1, \mathcal{C}_1 = 1, \mathcal{R}_{\text{row}} = h, \mathcal{C}_{\text{col}} = w$, to allow the robots to approach every obstacle. We choose $h_s \in \{1, 2, \ldots, h - 2\}, w_s \in \{1, 2, \ldots, w - 2\}$ such that all $\mathcal{C}_y, \mathcal{C}_{y+1} \in I_{\text{col}}, \mathcal{C}_{y+1} - \mathcal{C}_y = w_s + 1$ and all $\mathcal{R}_x, \mathcal{R}_{x+1} \in I_{\text{row}}, \mathcal{R}_{x+1} - \mathcal{R}_x = h_s + 1$. In such a case, there are $|I_{\text{row}}|(|I_{\text{col}}| - 1) + |I_{\text{col}}|(|I_{\text{row}}| - 1)$ lanes in the graph (e.g., there are 17 lanes with $|I_{\text{row}}| = 4, |I_{\text{col}}| = 3$ in Fig. 2).

To better demonstrate the narrow-lane environment, we use Gazebo [29] (Release version 11) to construct a simulated narrow-lane environment modeled by an automated warehouse with MRS, as shown in Figs. 4 and 5. Assumption 1 tells that the width of all the lanes equals one cell in the narrow-lane environment, and the storage capacity of the environment is maximized. However, if two robots move in the same lane with opposite directions, then they will collide. Since each lane does not allow two robots to move side by side, the free space for the robots to avoid each other is minimal, so this conflict is difficult to resolve.

The MPP problem involves n robots $r_1, \ldots, r_n$, where each robot r_i (can be simplified to i or r to represent the index of the robot in the following content for brevity) has a unique start position $s_i \in V$ and a unique goal position $g_i \in V$. We now denote the joint start positions as $X_S := \{s_1, \ldots, s_n\}$ and the goal positions as $X_G := \{g_1, \ldots, g_n\}$. In the rest of this paper, initial configurations contain robots' start positions and goal positions. The n robots here form a robot set $R := \{r_1, \ldots, r_n\}$ which contains all the robot indices. The objective of the MPP problem is to find a set of feasible paths

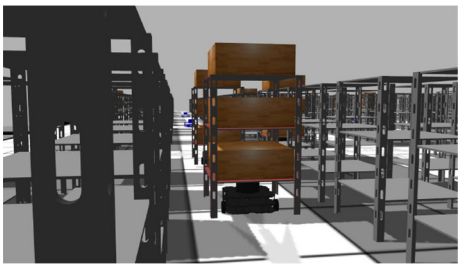


Fig. 5 Each lane cannot allow two robots pass side by side

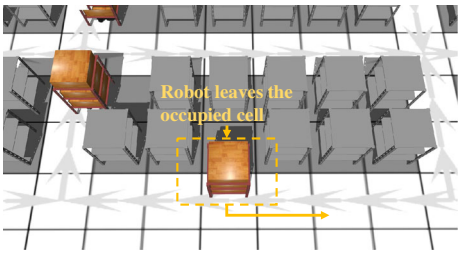


Fig. 6 A robot starts its task and leaves the occupied cell near the initial position

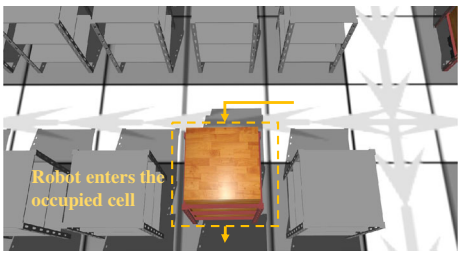


Fig. 7 A robot enters the occupied cell near the goal position and terminates its task

for all robots. Here, we define a path for robot r_i as a sequence of $T_i + 1$ vertices $P_i := \{p_i^0, \ldots, p_i^{T_i}\}$ that satisfies: (1) $p_i^0 = s_i$; (2) $p_i^{T_i} = g_i$; (3) $\forall 1 \leq t \leq T_i$, $p_i^{t-1} \in N(p_i^t)$. The path set of all the robots is then defined as $P := \{P_1, \ldots, P_n\}$. In the warehouse environment, each robot can go under a shelf and lift it to transfer goods. Considering such a scenario of the narrow-lane environment, similar to the sense of [30], we now give the formal well-formed infrastructure assumption.

Assumption 2 (*Well-formed infrastructure*) 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, $t \notin \{0, \ldots, T_i\}$.

Assumption 2 tells that each robot is under the shelf in the occupied cell at the beginning of the task. In Fig. 6, when the task begins, a robot can carry a shelf, leave the occupied cell, and then go into the lane. The occupied cell then has no shelf. In Fig. 7, when a robot reaches its goal position (i.e., $p_r^{T_r}$ or g_r), it will go into the occupied cell near its goal position.

Considering the well-formed infrastructure assumption above, we want P to be collision-free, such that, $\forall 1 \leq i < j \leq n, 1 \leq t \leq \min(T_i, T_j)$, P_i, P_j must satisfy (1) $p_i^t \neq p_j^t$ (no meet collisions on vertices); (2) $\left(p_i^{t-1}, p_i^t\right) \neq \left(p_j^t, p_j^{t-1}\right)$ (no head-on collisions on edges). All edges 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 optimal MPP problems based on different distance metrics in the narrow-lane environment:

Problem 1 *Min-Total-Distance MPP (MinTotalDist): 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 total robot distance* $\sum_{i=1}^{n} len(P_i)$.

Problem 2 *Min-Maximum-Distance MPP (MinMaxDist): 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* $\max_{1 \leq i \leq n} len(P_i)$.

3.2 Algorithm overview

Algorithm 1 The pipeline of the DLM algorithm

Require: G, X_S, X_G
Ensure: P
1: **Layer 1: Topological extraction**
$\bar{G}, \bar{X}_S, \bar{X}_G \leftarrow$ TopologicalExtraction (G, X_S, X_G)
2: **Layer 2: Fast feasibility heuristics**
$D^0, U^0 \leftarrow$ FastFeasibilityHeuristics$(\bar{G}, \bar{X}_S, \bar{X}_G)$
3: **Layer 1: IP model formulation**
$D, U \leftarrow$ IPModelFormulation$(\bar{G}, \bar{X}_S, \bar{X}_G, D^0, U^0)$
4: **Layer 1: Final path reconstruction**
$P \leftarrow$ FinalPathReconstruction (G, D, U)

Instead of modeling collision avoidance spatially and temporally for each robot, we mainly focus on keeping the narrow lane free by assigning the optimized direction to each lane, i.e., imposing specific traffic policies on all the lanes. All the robots should respect the proposed specific traffic policies. Furthermore, the input and output notations in each layer are outlined in Algorithm 1. As shown in Fig. 3, two layers in the DLM algorithm are decoupled. Meanwhile, it is worth noting in the Algorithm 1 that if the final path reconstruction (line 4) is used in the two layers, both individual layers in the DLM algorithm can generate the collision-free path set because lines 2 and 3 both return the variables of lane directions and robot motions. We can transfer these variables into a collision-free path set using variable-path bijection [31] and final path reconstruction. This is why the solutions of the second layer can be the initial solution of the first layer.

4 Layer 1: Integer programming-based MPP under specific traffic policies

In the first layer of the DLM algorithm, we derive an IP optimization model. We impose a unique passing direction on each narrow lane, optimizing the passing direction by minimizing the distance metrics. Here, we name the specific traffic policies on narrow lanes as *one-way constraints*. The process of the first layer is given as follows: (A) Firstly, the original grid graph is extracted into a topological graph with fewer vertices and edges. (B) Secondly, IP model is formulated in the topological graph to achieve the paths in the topological graph and the optimized direction of each lane. (C) Finally, we transform the paths in (B) into the paths in the original grid graph.

The three steps above are feasible because this allows for vertices whose connectivity degree is less than or equal to two to be compressed into a meta vertex under one-way constraints [32]. The connectivity degree of a vertex in a narrow lane equals two as it only connects to the other two neighboring vertices in the narrow lane or the lane intersection. 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 decreased radically compared to the IP model in the original grid graph. Considering the assumptions above, we propose formally the following lemma to avoid the collisions in narrow lanes using one-way constraints.

Lemma 1 *(*One-way constraints*) A path set obeying one-way constraint P in G requires:*

for all $r, q \in R, T_r, T_q \in \mathbb{N}$*, if edge* $\left(p_r^t, p_r^{t+1}\right)$ *exists in any* P_r*, the reversed edge* $\left(p_r^{t+1}, p_r^t\right)$ *cannot exist in any other* P_q*. The head-on inter-robot collision is avoided in the narrow lanes, if one-way constraints are obeyed by all robots.*

Proof By contradiction, if $\left(p_r^{t+1}, p_r^t\right)$ also exists in another path P_q, such that $\left(p_q^{t_q}, p_q^{t_q+1}\right)$ exists in P_q with $p_q^{t_q} = p_r^{t+1}$, $p_q^{t_q+1} = p_r^t$, it is always possible to generate collision between P_r and P_q by changing robot q's positions in t_q and $t_q + 1$ to its position in t and $t + 1$ (e.g., robot q's velocity changes abruptly) and causing the head-on collision: $p_q^t = p_r^{t+1}$, $p_q^{t+1} = p_r^t$, i.e., $\left(p_q^t, p_q^{t+1}\right) = \left(p_r^{t+1}, p_r^t\right)$. This case does not accord to the existence of collision-free path set. So Lemma 1 holds. □

4.1 Topological extraction

The narrow lanes only allow the robot to pass in a unique direction as a direct result of one-way constraints. We can denote a robot's moving direction in each narrow lane using the robot's occupancy in two intersection vertices at both ends of the lane. Then, we project the original grid graph into a topological graph that contains only the intersection vertices. Here, in Fig. 8, the vertices surrounded by circles (e.g., $v_1, v_5, v_9, v_{19}, \ldots$) on the left side indicate the intersection vertices in the grid graph. We extract them and add bidirectional edges between two neighboring intersection vertices. Therefore, we introduce the following topological graph:

Definition 1 The topological graph 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 in G. For $v_i \in \bar{V}$, let $\bar{N}(v_i) := \{v_j | (v_i, v_j) \in \bar{E}\}$ be the neighborhood of v_i in $\bar{G}$.

Here, we note that $\bar{G} = (\bar{V}, \bar{E})$ is a bidirectional graph, 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 graph and the edges in the topological graph.

Definition 2 Given a lane $\{v_i, v_j\}$ in the original narrow-lane environment, which is bounded by two intersection vertices v_i, v_j. Each lane $\{v_i, v_j\}$ is correlated to bidirectional edges: (v_i, v_j) and (v_j, v_i) in topological graph $\bar{G}$.

Recalling the well-formed infrastructure, X_S and X_G are placed inside the lanes, i.e., $X_S \cap \bar{V} = \emptyset$, $X_G \cap \bar{V} = \emptyset$. To make all the robots go through their corresponding initial configurations by one-way constraints, we set the corresponding vertices of s_r and g_r in $\bar{G}$ as the two intersection vertices at both ends of the narrow lanes where s_r and g_r are located: $\bar{s}_r = \{\bar{s}_r^1, \bar{s}_r^2\}$, $\bar{g}_r = \{\bar{g}_r^1, \bar{g}_r^2\}$, $\bar{X}_S = \{\{\bar{s}_r^1, \bar{s}_r^2\} | \forall r \in R\}$, $\bar{X}_G = \{\{\bar{g}_r^1, \bar{g}_r^2\} | \forall r \in R\}$. For example, if the index of $\bar{s}_r$ is 21, the indices of $\bar{s}_r^1$ and $\bar{s}_r^2$ are 19 and 23, respectively. We then propose the corresponding variants in $\bar{G}$ of Problems 1 and 2 formally.

Problem ***1'*** *MinTotalDist problem in* $\bar{G}$*: Given a 3-tuple* $\langle \bar{G}, \bar{X}_S, \bar{X}_G \rangle$*, find a path set* $\bar{P} := \{\bar{P}_1, \ldots, \bar{P}_n\}$ *from vertices in* $\bar{X}_S$ *to vertices in* $\bar{X}_G$*, and minimizes the total robot distance* $\sum_{i=1}^{n} len(\bar{P}_r)$.

Problem ***2'*** *MinMaxDist problem in* $\bar{G}$*: Given a 3-tuple* $\langle \bar{G}, \bar{X}_S, \bar{X}_G \rangle$*, find a path set* $\bar{P} := \{\bar{P}_1, \ldots, \bar{P}_n\}$ *from vertices in* $\bar{X}_S$ *to vertices in* $\bar{X}_G$*, and minimizes the maximum robot distance* $\max_{1 \le i \le n} len(\bar{P}_r)$.

The two problem variants' results both satisfy the following requirements:

$$\forall r \in R, \bar{P}_r := \left\{\bar{p}_r^0, \bar{p}_r^1, \ldots, \bar{p}_r^{\bar{T}_r}\right\}, \bar{P}_r \subset \bar{V},$$

(i) *Make sure the path continuity:* $\forall t, 1 \le t \le \bar{T}_r$, $\bar{p}_r^t \in \bar{N}\left(\bar{p}_r^{t-1}\right)$;

(ii) *Go through initial configurations:* $\{\bar{p}_r^0, \bar{p}_r^1\} = \bar{s}_r$, $\{\bar{p}_r^{\bar{T}_r - 1}, \bar{p}_r^{\bar{T}_r}\} = \bar{g}_r$.

As discussed, if one robot goes through the lanes where its initial configurations are placed, then it will definitely

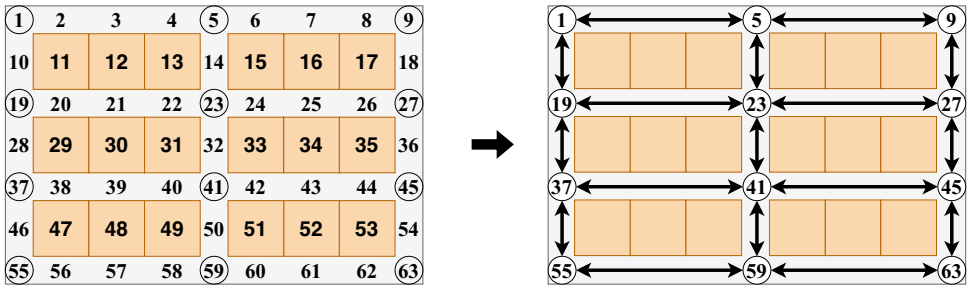


Fig. 8 From the initial grid graph to the topological graph

go through the initial configurations. We can formulate the MPP problem model mathematically using the IP approach in $\bar{G}$ with few decision variables. The mathematical model is solved in $\bar{G}$, and the solution can be transformed to G.

4.2 Integer programming model formulation

Many studies such as [17,31] have shown that the MPP problem can be rewritten as a linear program with integer constraints that count for path continuity and collision avoidance. As we know, solving an IP model may be intractable if the decision variables are coupled and numerous. Thus, we try to use as few decision variables as possible and decouple the decision variables from each other. We derive the mathematical representation of Problem 1' and Problem 2' from path continuity constraints, one-way constraints, and objective functions.

4.2.1 Path continuity constraints

Given $\langle \bar{G}, \bar{X}_S, \bar{X}_G \rangle$, the IP model here introduces a binary decision variable set $U := \{u_1, \ldots, u_n\}$. We use $u_r(v_i, v_j)$ for each robot r and for each edge $(v_i, v_j) \in \bar{E}$ to indicate whether (v_i, v_j) is passed in $\bar{P}_r$, i.e., if robot r traverses (v_i, v_j) in $\bar{G}$, $u_r(v_i, v_j) = 1$; otherwise, $u_r(v_i, v_j) = 0$. The decision variable set U contains the motions of all robots in $\bar{G}$. Above all, to meet the requirement (i) in both Problem 1' and Problem 2', the global path continuity constraint (1) is respected so that vertices in $\bar{P}_r$ are consecutive in $\bar{G}$, i.e., the fact $\bar{p}_r^{t+1} \in \bar{N}(\bar{p}_r^t)$ is followed.

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

According to Fig. 9, three conditions are considered to meet the requirement (ii), such that each robot r's path goes through s_r and g_r.

Condition (a): 0 shared vertex. As shown in Fig. 9a, start and goal positions are on two disjoint lanes, i.e., the two lanes have no intersection, we now recall the representation: $\bar{s}_r = \{\bar{s}_r^1, \bar{s}_r^2\}$, and $\bar{g}_r = \{\bar{g}_r^1, \bar{g}_r^2\}$.

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

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

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

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

Constraint (3) indicates that there is and only one of $\mathcal{X}(\bar{s}_r^1)$ and $\mathcal{X}(\bar{s}_r^2)$ equals to 1, such that, there is and only one of vertices $\bar{s}_r^1$ 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 at most once in $\bar{P}_r$:

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

Condition (b): one shared vertex. As shown in Fig. 9b, robot's start and goal positions are on two adjacent lanes, such that the two lanes have one intersection (e.g., the gray 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 $\bar{P}_r$:

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

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, such that v_i can be passed at most once in $\bar{P}_r$.

Condition (c): two shared vertices. As shown in Fig. 9c, robot's start and goal positions are on the same lane. We reset the initial configuration of robot r in $\bar{G}$ in the same lane as $\{\bar{s}_r', \bar{g}_r'\}$, according to the relative positions of s_r and g_r, i.e., $\bar{s}_r' = \arg\min_{v \in \bar{V}}(dist(v, s_r))$, $\bar{g}_r' =$

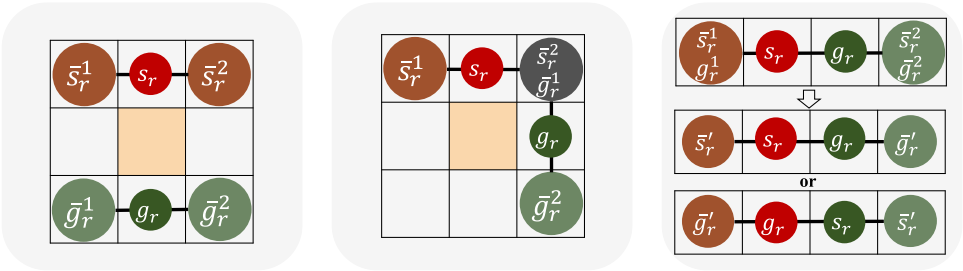


Fig. 9 Relative positions of initial configurations. Each circle represents one vertex in $\bar{G}$

(a) Zero shared vertex. **(b)** One shared vertex. **(c)** Two shared vertices.

$\arg\min_{v \in \bar{V} \setminus \{\bar{s}_r'\}}(dist(v, g_r))$.

$$\sum_{v_j \in \bar{N}(\bar{s}_r')} u_r(\bar{s}_r', v_j) = \sum_{v_j \in \bar{N}(\bar{g}_r')} u_r(v_j, \bar{g}_r') = 1. \quad (7)$$

$$\sum_{v_j \in \bar{N}(\bar{s}_r') \setminus \{\bar{g}_r'\}} u_r(v_j, \bar{s}_r') = \sum_{v_j \in \bar{N}(\bar{g}_r') \setminus \{\bar{s}_r'\}} u_r(\bar{g}_r', v_j) = 0. \quad (8)$$

$$u_r(\bar{s}_r', \bar{g}_r') + u_r(\bar{g}_r', \bar{s}_r') = 1. \quad (9)$$

Constraints (7)–(9) ensure that $\bar{s}_r'$ and $\bar{g}_r'$ should be visited in order. If $v_i \in V \setminus \{\bar{s}_r', \bar{g}_r'\}$, constraint (6) makes sure that v_i can be visited only once in $\bar{P}_r$.

4.2.2 Specific traffic policies: one-way constraints

Lemma 1 tells that one-way constraints are imposed on narrow lanes to avoid the inter-robot collisions as specific traffic policies, such that each lane has only one unique passing direction. Because $\bar{G}$ is a bidirectional graph, (v_i, v_j) and (v_j, v_i) stands for the same lane, we then use $\{v_i, v_j\}$ to represent the corresponding lane. As we want to set the optimized direction to each narrow lane, we introduce a decision variable: $D(\{v_i, v_j\})$, $(v_i, v_j), (v_j, v_i) \in \bar{E}$ to represent the lane's direction. Then, the size of decision variable D is $\frac{1}{2}\left|\bar{E}\right| \times 1$.

Numerically, we give the description of lane's direction correlated to $\{v_i, v_j\}$ in $\bar{G}$, v_i and v_j are two intersection vertices in two ends of the lane, $|v_i| < |v_j|$:

1. If the passing direction of $\{v_i, v_j\}$ is from v_i to v_j, $D(\{v_i, v_j\}) = 1$;
2. If the passing direction of $\{v_i, v_j\}$ is 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 $\forall (v_i, v_j) \in \bar{E}, r \in R$:

$$\sum_{r \in R} u_r(v_i, v_j) = 0 \vee \sum_{r \in R} u_r(v_j, v_i) = 0 \quad (10)$$

$$\left|u_r(v_i, v_j) - u_r(v_j, v_i) - D\left(\{v_i, v_j\}\right)\right| \leq 1. \quad (11)$$

Recalling constraint (10), we avoid such a case:

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

Therefore, no two robots can run in opposite directions in the same lane, such that each lane is restricted to be one-way. Constraint (11) 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 Complete mathematical model

Based on the constraints above, we can formulate the complete mathematical model, which an IP solver can solve. In this work, we work on two objectives focusing on distance optimality, as we denote in Sect. 3. We can now transform them into objective functions and propose the complete mathematical model. For each edge $(v_i, v_j) \in \bar{E}$, we suppose that the Cartesian coordinates and indices of v_i, v_j in $\bar{G}$ 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: $W[\{v_i, v_j\}] := |i_1 - j_1| + |i_2 - j_2|$, equals to the length of lane $\{v_i, v_j\}$.

Minimize Total Distance (MinTotalDist) in $\bar{G}$: 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 complete mathematical model can then be derived as:

$$\min \sum_{r \in R} \sum_{(v_i, v_j) \in \bar{E}} W[\{v_i, v_j\}] u_r(v_i, v_j), \text{ subject to} (1) - -(11).$$

Minimize Maximum Distance (MinMaxDist) in $\bar{G}$: To minimize the maximum robot distance, we introduce an additional integer decision variable $umax$. For each robot's distance, we add the upper bound constraint:

$$\sum_{(v_i, v_j) \in \bar{E}} W[\{v_i, v_j\}] u_r(v_i, v_j) \leq umax(r). \quad (12)$$

The upper bound of single robot's distance is minimized:

$$\min_{r \in R} umax(r), \text{ subject to}(1) - (12).$$

Referring to Proposition III.1 in the paper [31], 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, because of path continuity constraints, 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 $u_r \in U$. Thus, we can solve Problem 1' and Problem 2'. Meanwhile, we can provide optimized results of robots' motion and lane directions.

4.3 Final path reconstruction

In the last part of the first layer, for each robot $r \in R$, $\bar{P}_r \subset \bar{V}$, we want to get the collision-free path set P in G from $\bar{P}$. Firstly, between two consecutive vertices v_i, v_j in $\bar{P}_r$ (e.g., vertices 5 and 9 in Fig. 8), $(v_i, v_j) \in \bar{E}$, we should add all vertices $v \in V \backslash \bar{V}$ between v_i and v_j sequentially into $\bar{P}_r$ (e.g., vertices 6,7 and 8 between vertices 5 and 9 in Fig. 8). The path continuity constraints restrict that each $\bar{P}_r$ should traverse the lane where start positions s_r and goal positions g_r are situated. Thus, s_r and g_r are already in P_r, we make them be the first and last vertex of P_r respectively by removing the redundant vertices before s_r and after g_r. For each path $P_r \in P$, we ensure that every vertex in P_r is continuous in $G = (V, E)$. Finally, the simple waiting strategy at the robotic level is used to avoid the collision in the lane intersections if more than one robot is getting into the lane intersection simultaneously. The robot will wait in the lane or enter a nearby available obstacle cluster when it detects that the lane intersection ahead is busy. Finally, all the robots can move to their corresponding goal positions without collision following the scheduled P. Therefore, Problem 1 and Problem 2 are solved with optimized results[1].

After the first layer, to provide the formulated IP model with efficient initial feasible solutions to accelerate the solving process, we provide decoupled fast feasibility heuristics in the second layer to make the formulated IP model solvable and make the DLM algorithm always has feasible solutions.

5 Layer 2: Fast feasibility heuristics

We use the fast feasibility heuristics to operate the IP model's solving procedure with the warm start. We implement the fast feasibility heuristics in the second layer of the DLM algorithm (see Fig. 3). It is imperative to factor out that when the IP solver's general built-in feasibility heuristics are slow in finding an initial feasible solution, it is helpful to preprocess the MPP problem to operate the IP model with an efficient initial solution, i.e., feeding the IP solver with the initial robots' motion and lane directions which satisfy all the constraints. Referring to the manual of Gurobi solver [26], the built-in feasibility heuristics will be called to provide an initial feasible solution for the search process of the optimal solution (or we can consider this initial feasible solution as the starting point of the optimal solution search). We provide two fast feasibility heuristics faster than Gurobi's built-in feasibility heuristics with a better initial solution.

We make sure that the initial solutions in $\bar{G}$ satisfy all the constraints in the IP model of the first layer. Here, the heuristic solutions are presented as U^0 and D^0 correlated to the decision variables: U and D. First of all, we find the start position $\bar{s}_r^0$ and goal position $\bar{g}_r^0$ in $\bar{G}$ for the two approaches. $\bar{g}_r^0$ in $\bar{G}$ are the closest intersection vertices to s_r and g_r, i.e.,

$$\bar{s}_r^0 := \arg\min_{v \in \bar{V}}(dist(v, s_r)), \ \bar{g}_r^0 := \arg\min_{v \in \bar{V} \backslash \{\bar{s}_r^0\}}(dist(v, g_r)).$$

Thus, the notations of initial configurations in this layer are given as: $\bar{X}_S^0 = \{\bar{s}_r^0 | \forall r \in R\}$, $\bar{X}_G^0 = \{\bar{g}_r^0 | \forall r \in R\}$. This section presents two fast feasibility heuristics that can generate initial solutions for the IP model in the first layer.

5.1 Polynomial-complexity approach: O-instance regulation

Figure 10 outlines the approach's brief procedure. The O-instance regulation is called a polynomial-complexity approach since it finishes its solving procedure and return the initial feasible solutions in polynomial time complexity.

5.1.1 Edge potential computation

Inspired by [24,33], edge potential can be used to determine the direction of the edge. We use independent optimal paths to compute edge potential. Independent optimal path $\bar{P}^{ind} := \{\bar{P}_1^{ind}, \ldots, \bar{P}_n^{ind}\}$ for each robot is found with the Dijkstra search ignoring other robots.

Here, we define the continuous vertex sequence $\{v_1, v_2, \ldots, v_m\}$ with m ($m \geq 2$) vertices in $\bar{G}$. Vertices in $\{v_1, v_2, \ldots, v_m\}$ are continuous in $\bar{G}$, such that each continuous pair of vertices $v_i, v_{i+1} \in \{v_1, v_2, \ldots, v_m\}$ satisfies: $v_{i+1} \in \bar{N}(v_i)$. The vertex sequence has corresponding edge set satisfying:

$$\mathcal{E}_{\bar{G}}(\{v_1, v_2, \ldots, v_m\}) := \{(v_1, v_2), \ldots, (v_{m-1}, v_m)\} \cap \bar{E}. \tag{13}$$

By $\bar{P}^{ind}$, we use the total distance traveled by all robots on this edge as the potential of this edge $\mathcal{O} : \left|\bar{E}\right| \times 1$, as we

[1] We provide the complete performance in the attached video.

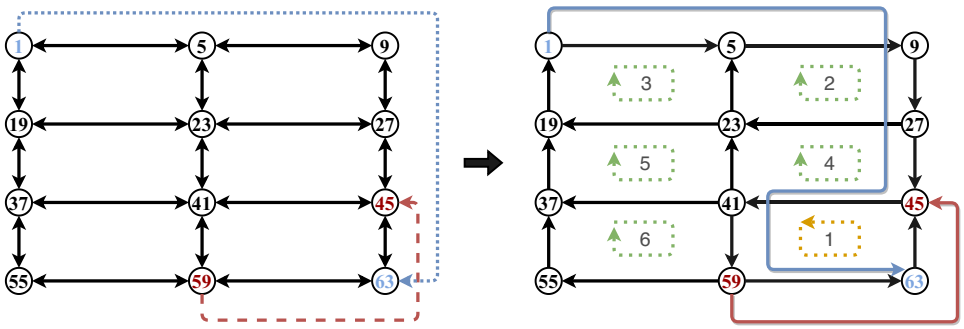


Fig. 10 In the overall process of the O-instance regulation approach shown in the figure above, the environment background is omitted for clearness: (Left) $\bar{G}$ with an independent collision-ignorance path set of robots $\bar{P}^{ind}$. (Right) $\bar{G}$ with the initial feasible solution $\bar{P}$ and the initial lane directions with directions of O-instances

describe in Algorithm 2. We then make the initial solution satisfy the one-way constraints.

Algorithm 2 Edge_Potential_Computation

Require: $\bar{X}_S^0, \bar{X}_G^0, \bar{G}$
Ensure: $\mathcal{O}$
1: $\mathcal{O}(\bar{E}) = \mathbf{0}$
2: **for** $\forall r \in R$ **do**
3: $\quad \bar{P}_r^{ind} = \text{Dijkstra}\left(\bar{G}, \bar{s}_r^0, \bar{g}_r^0\right)$
4: $\quad$ **for** $\forall e \in \mathcal{E}_{\bar{G}}(\bar{P}_r^{ind})$ **do**
5: $\quad\quad \mathcal{O}(e) = \mathcal{O}(e) + |e|$
6: $\quad$ **end for**
7: **end for**
8: **return** $\mathcal{O}$

5.1.2 O-instance regulation

As shown in Fig. 10, the shortest continuous end-to-end vertex cycle in $\bar{G}$ departing from $v_i \in \bar{V}$: $\{v_i, v_j, v_k, v_l, v_i\}$ is formed by four vertices: $v_i, v_j, v_k, v_l \in \bar{V}$, the end-to-end vertex cycle surrounds the position an obstacle cluster in the narrow-lane environment. Using the same four vertices, there is another end-to-end vertex cycle $\{v_i, v_l, v_k, v_k, v_i\}$, the two vertex cycles form a pair of shortest continuous end-to-end vertex cycles surrounding the same obstacle cluster. Taking Fig. 10 for example, $\{v_{41}, v_{45}, v_{63}, v_{59}, v_{41}\}$ and $\{v_{41}, v_{59}, v_{63}, v_{45}, v_{41}\}$ are such a pair of vertex cycles. There are N_{oc} pairs of such vertex cycles in $\bar{G}$ because there are N_{oc} obstacle clusters. Since $\bar{G}$ is bidirectional, two such vertex cycles departing from the same vertex will pass the same vertices in opposite directions. Setting $v_i \in \bar{V}$ in the upper left corner of each obstacle cluster as the departure vertex, and $v_k \in \bar{V}$ in the lower right corner as the passing vertex, since the vertices in the vertex circle are in the shape of "O", we call the pair of vertex circles as *O-instance*, we then define it formally.

Definition 3 An *O-instance* $O_s := \{o_s^1, o_s^2\}$ is defined as the pair of the two shortest continuous end-to-end vertex cycles surrounding the position of the same obstacle cluster s, $s \in \{1, \ldots, N_{oc}\}$, $o_s^1 := \{v_i, v_j, v_k, v_l, v_i\}$ and $o_s^2 := \{v_i, v_l, v_k, v_j, v_i\}$. A child of *O-instance* is defined as one of the shortest continuous vertex cycles in $\bar{G}$. We define the *clockwise* child as o_s^1 and *counter-clockwise* child o_s^2.

Recalling Eq. (13), each child in O_s is also a vertex sequence, so it has a corresponding edge set: $\mathcal{E}_{\bar{G}}(o_s^1)$ and $\mathcal{E}_{\bar{G}}(o_s^2)$. To satisfy one-way constraints, we want to form a new directed topological graph $\bar{G}_d = (\bar{V}_d, \bar{E}_d)$. Edges in $\mathcal{E}_{\bar{G}}(o_s^1)$ and $\mathcal{E}_{\bar{G}}(o_s^2)$ cannot exist simultaneously in $\bar{E}_d$.

Here, we set the corresponding heuristic weight of O-instance's child $H(o_s^1)$ and $H(o_s^2)$ as the total distance traveled by the robots in $\mathcal{E}_{\bar{G}}(o_s^1)$ and $\mathcal{E}_{\bar{G}}(o_s^2)$.

$$H(o_s^1) = \sum\nolimits_{e \in \mathcal{E}_{\bar{G}}(o_s^1)} \mathcal{O}(e),\ H(o_s^2) = \sum\nolimits_{e \in \mathcal{E}_{\bar{G}}(o_s^2)} \mathcal{O}(e).$$

$$o_s^{opt} = \arg\max_{o \in O_s} (H(o)).$$

We use o_s^{opt} to denote the one of o_s^1 and o_s^2 whose edges robots go through more frequently, and we add the edges in $\mathcal{E}_{\bar{G}}(o_s^{opt})$ into $\bar{E}_d$. We sort o_s^{opt}, $s \in \{1, \ldots, N_{oc}\}$ in descending order by the heuristic weight $H(o_s^{opt})$.

Therefore, the construction of the directed graph $\bar{G}_d = (\bar{E}_d, \bar{V}_d)$ is launched sequentially from the highest-weight $o_{N_{oc}}^{opt}$ to the lowest-weight o_1^{opt}, and we can reconstruct $\bar{E}_d$ and $\bar{V}_d$ using the edges in $\mathcal{E}_{\bar{G}}(o_s^{opt})$ and vertices in o_s^{opt}. Algorithm 3 gives the process of O-instance regulation.

Considering the directions of the edges shared by two adjacent o_s^{opt}, as indicated in lines 9–11 of Algorithm 3, o_s^{opt} with higher potential determines these edges' directions, i.e., the lower-priority o_s^{opt} cannot change the edges in $\bar{E}_d$ which have been regulated by higher-priority o_s^{opt}. Finally, in the right of Fig. 10, $\bar{G}_d$ is now directed (thin black straight arrows). And the directions of O-instances using the dotted bend rounded arrows. The black Arabic numerals (1-6) inside the arrows indicate the priorities of the chosen o_s^{opt}.

Proposition 1 *$\bar{G}_d$ is strongly connected, and $\bar{G}_d$ satisfies that any $v_s, v_t \in \bar{V}_d$, there exists at least one path from v_s to v_t in $\bar{G}_d$.*

Proof Firstly, there is no doubt that for all $s \in \{1, 2, \ldots, N_{oc}\}$, $v_1, v_2 \in o_s^{opt}$, there exists a path from v_1 to v_2 since the graph

Algorithm 3 O-instance_Regulation

Require: $\mathcal{O}, \bar{G}$
Ensure: $\bar{G}_d$
1: **for** $\forall s \in \{1, \ldots, N_{oc}\}$ **do**
2: $o_s^{opt} = \arg\max_{o \in O_s} (H(o))$
3: **end for**
4: Sort all $o_s^{opt}, s \in \{1, \ldots, N_{oc}\}$ in the descending order of $H(o_s^{opt})$.
5: $\bar{E}_d = \mathcal{E}_{\bar{G}}(o_1^{opt}), \bar{V}_d = o_1^{opt}$
6: **for** $\forall s \in \{2, \ldots, N_{oc}\}$ **do**
7: $E_{tmp} = \mathcal{E}_{\bar{G}}(o_s^{opt}), V_{tmp} = o_s^{opt}$
8: **for** $\forall (v_i, v_j) \in \mathcal{E}_{\bar{G}}(o_s^{opt})$ **do**
9: **if** $(v_i, v_j) \in \bar{E}_d \vee (v_j, v_i) \in \bar{E}_d$ **then**
10: $E_{tmp} = E_{tmp} \backslash \{(v_i, v_j)\}, V_{tmp} = V_{tmp} \backslash \{v_i, v_j\}$
11: **end if**
12: **end for**
13: $\bar{E}_d = \bar{E}_d \cup E_{tmp}, \bar{V}_d = \bar{V}_d \cup V_{tmp}$
14: **end for**
15: **return** $\bar{G}_d = (\bar{V}_d, \bar{E}_d)$

formed by vertices in regulated o_s^{opt} and edges in $\mathcal{E}_{\bar{G}}(o_s^{opt})$ is strongly connected. Secondly, we form $\bar{G}_d$ sequentially using o_s^{opt}, after each iteration, the connectivity of the original $\bar{G}_d$ is not affected since the lower-priority o_s^{opt} cannot change the edges in $\bar{E}_d$ which have been regulated by higher-priority o_s^{opt}. □

5.1.3 Initial feasible solution generation

Here, using the strongly connected $\bar{G}_d$, feasible initial heuristic solution can be found easily to satisfy all the constraints in IP model by applying Dijkstra algorithm: $\bar{P}_r^0 = \text{Dijkstra}\left(\bar{G}_d, \bar{s}_r^0, \bar{g}_r^0\right)$. The directed lines over the lane directions in the right part of Fig. 10 represent the robots' feasible paths. The initial feasible value of decision variables x is derived using the initial path set $\bar{P}^0$.

Proposition 2 *O_instance regulation's time complexity is polynomial:* $O(n\left|\bar{V}\right|^2)$.

Proof In Sect. 5.1.1, we apply the Dijkstra search [8] for n robots with time complexity of $O(n\left|\bar{V}\right|^2)$. In Sect. 5.1.2, *O-instance regulation* is terminated in $O(\left|\bar{V}\right|^2)$. In Sect. 5.1.3, $\bar{P}^0$ is searched in $O(n\left|\bar{V}\right|^2)$ with Dijkstra Algorithm. □

5.2 Optimality-based approach: small-scale IP

The O-instance regulation approach cannot guarantee the solution's optimality, although its time complexity is proven to be polynomial. Thus, we derive a small-scale optimization IP model which achieves the same function as the O-instance regulation approach but improves the optimality of solutions. Unlike the formulated optimization IP model in the first layer, we directly use "edges" as decision variables in the small-scale IP model. Thus, the dimension of decision variables is limited to be $\frac{1}{2}\left|\bar{E}\right|$.

5.2.1 Objective formulation

We want to make the initial paths approach the independent optimal paths by formulating a new directed graph $\bar{G}_d$ based on the bidirectional $\bar{G}$ and satisfying all the constraints in the first layer.

Firstly, we compute independent optimal paths $\bar{P}^{ind}$ in $\bar{G}$ using Dijkstra algorithm [8] for each robot from $\bar{s}_r^0$ to $\bar{g}_r^0$, ignoring the robot collisions.

Secondly, according to $\bar{P}_r^{ind}$, we use $A_r(\{v_i, v_j\})$ to denote the direction of each robot r traversing each lane $\{v_i, v_j\}$. Here, v_i and v_j are the two intersection vertices which bound the lane, $|v_i| < \left|v_j\right|$ and $(v_i, v_j), (v_j, v_i) \in \bar{E}$. The value of $A_r(\{v_i, v_j\})$ is decided by the three cases:

1. If robot does not pass $\{v_i, v_j\}$, $A_r(\{v_i, v_j\}) = 0$;
2. If robot traverses from v_i to v_j, $A_r(\{v_i, v_j\}) = 1$;
3. If 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 $A : n \times \frac{1}{2}\left|\bar{E}\right|$, which is called the robots' independent direction matrix. The independent direction matrix reflects robots' motions in the independent optimal paths.

Thirdly, we set the $\frac{1}{2}\left|\bar{E}\right| \times 1$ 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 the second layer can then be derived as follows to make the initial lane direction as close to the direction in the independent optimal paths as possible:

$$\min \left(\sum_{r \in R} A_r \cdot D^0 \right). \tag{14}$$

5.2.2 Feasible direction optimization

To make the solutions of the fast heuristics feasible, we propose two cases of feasible constraints here to keep the directed graph strongly connected.

(1) *Vertex degree constraints*: To make all the vertices 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. Thus, the directions of lanes $D^0(\{v_i, v\}), v_i \in \bar{N}(v)$ are restricted.

(2) *Lane direction constraints*: In the topological graph $\bar{G}$, for each pair of opposite lanes: $\{v_i, v_j\}$ and $\{v_a, v_b\}$, for example, $\{v_1, v_5\}$ and $\{v_{19}, v_{23}\}$, $\{v_1, v_{19}\}$ and $\{v_5, v_{23}\}$ in Fig. 8, the lane direction constraint should be respected: if $\{v_i, v_j\}$, $\{v_a, v_b\}$ are opposite, then

$$D^0(\{v_i, v_j\}) + D^0(\{v_a, v_b\}) = 0.$$

The lane direction then alternates from one row (or column) to the next.

We build an IP model using the objective function and constraints above. Gurobi solver is used to quickly solve such a small-scale IP model to get the optimal lane direction D^0 under the above feasibility constraints.

Proposition 3 *The small-scale IP model can always generate solutions that satisfy all the constraints in the formulated IP model in the first layer of DLM algorithm.*

Proof We note that there is always a feasible solution for the small-scale IP model, e.g., Fig. 4 in [24]. The small-scale IP model is always feasible to be solved. □

5.2.3 Feasible initial path generation

Here, a directed graph $\bar{G}_d = (\bar{V}_d, \bar{E}_d)$ is provided through the solved lane direction D^0, meanwhile $\bar{V}_d = \bar{V}$. Since the feasible constrains in the small-scale IP model make sure that $\bar{G}_d$ is strongly connected, initial paths $\bar{P}^0$ from $\bar{X}_S$ and $\bar{X}_G$ can be searched easily using A^* [9] or Dijkstra's search [8], satisfying all the constraints in the first layer.

With the feasible initial path $\bar{P}^0$ generated by the two fast feasibility heuristics, according to the bijection between the binary decision variable set U and all paths $\bar{P}$ from $\bar{X}_S$ to $\bar{X}_G$, initial solution U^0 can be derived from $\bar{P}^0$. We can now achieve the initial feasible solutions of the decision variable set for all robots' motions U and lane directions D: U^0, D^0 in the proposed IP model.

Proposition 4 *The proposed DLM algorithm always has feasible solutions in narrow-lane environments.*

Proof Firstly, the two fast feasibility heuristics always generate solutions that satisfy all the constraints of the formulated IP model in the first layer (ref. Proposition 1 and Proposition 3). Then the formulated IP model can always be solved because once we provide the IP solver with initial feasible solutions, it will search for the optimized solution starting from the given solutions. The worst solution the IP solver finds is the given initial feasible solution. Finally, the final path reconstruction always generates collision-free paths using the solutions found by the IP solver. Thus, the DLM algorithm can always return feasible collision-free paths. □

6 Evaluation results

We now evaluate the DLM algorithm in the grid narrow-lane environment.

6.1 Comparative evaluation setup

To simplify the comparative analysis of algorithms, we apply all the algorithms in a grid graph based on the narrow-lane environment with a group of robots and obstacle clusters in MATLAB. We assign the number of robots and environment settings (e.g., the environment size, the number of the size of each obstacle cluster) before the algorithm execution. In the simulation implementation for comparative analysis, the robot is assumed to be a unit-sized object. Each robot is assigned a pair of two different start and goal positions. Paths are generated to navigate robots from their start positions to goal positions. Each algorithm solves the same instance independently for each test point, and we store the solved paths and running time for comparative analysis. The results obtained for comparative analysis are evaluated by storing each robot's path length and running time elapsed as algorithm output, and we use them to formulate the comparative metrics. All experiments are executed on an Intel® Core™ CPU i7-10700 with 32GB RAM at 2.90GHz.

We implement the DLM algorithm in MATLAB 2021a. Concerning IP exact and k-way split (optimal and sub-optimal MPP algorithm) based on network flows, we use the existing implementation by [17] formulated in Java and solved by Gurobi solver (Academic Version 9.1.1). We adopt the PIBT method (MPP algorithm with polynomial complexity) by [19] and Enhanced CBS (fast sub-optimal MPP algorithm) by [20] written in C++. The maximum running time is limited to 10^3 seconds. In the implementation, part of the simulation data obtained is displayed in Table 1.

6.2 Comparison between fast feasibility heuristics

To demonstrate the performance of the two fast feasibility heuristics, we compare them in the simulated narrow-lane environment. Here, we demonstrate the efficiencies of the proposed two fast feasibility heuristics along with Gurobi's general built-in feasibility heuristics. We use the three heuristics to solve the same instance of MPP problem in each test. We design the problem instances in the narrow-lane environments with different sizes: 19×43 with 36 units of 2×6 obstacle clusters and 21×21 with 25 units of 3×3 obstacle clusters. We compare their running time and optimality performance.

We use the final path reconstruction to achieve the complete collision-free path from the output of the fast feasibility heuristics. 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, Gurobi solver returns it without optimization. We introduce two metrics to evaluate our two proposed feasibility heuristics: (1) total path length ratio; (2) running time ratio.

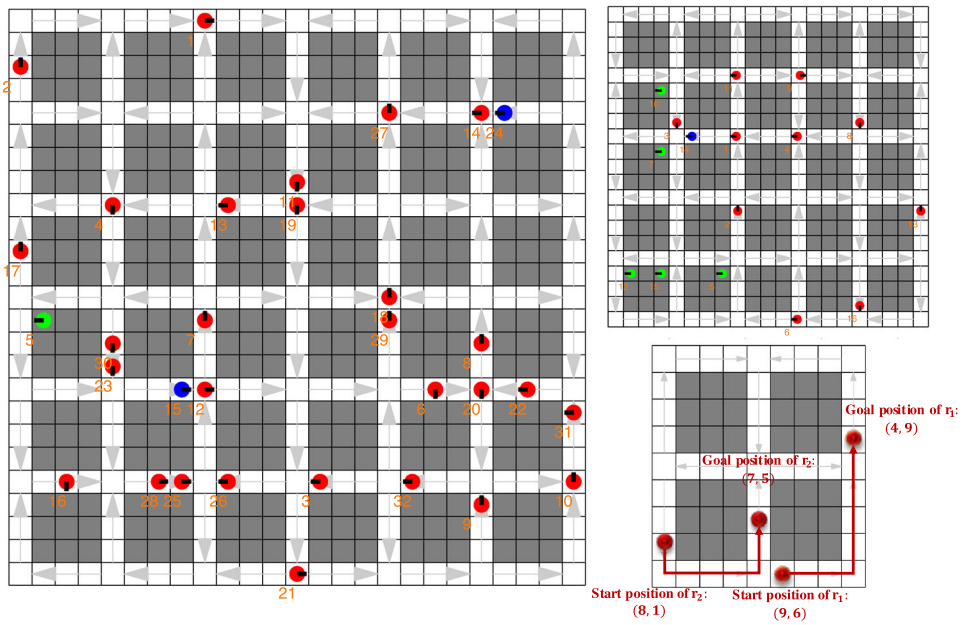


Fig. 11 The narrow-lane environments with 36, 25, 4 obstacle clusters in size of 3×3, with 32, 16, 2 robots respectively. Red robots are moving, blue robots are waiting, and the green robot has reached its goal. Gray arrows indicate the optimized lanes' directions. Black lines on the robots indicate their moving directions

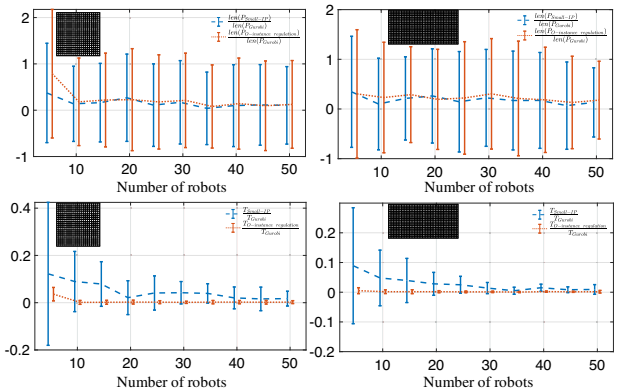


Fig. 12 Comparison between the fast feasibility heuristics

To simplify the literature, we use $len(P_{small-scale\ IP})$, $len(P_{O-instance\ regulation})$, $len(P_{Gurobi})$ to represent the total length of feasible paths generated by small-scale IP approach, O-instance regulation approach, Gurobi's built-in feasibility, respectively. We use $T_{small-scale\ IP}$, $T_{O-instance\ regulation}$, T_{Gurobi} to represent the running time of small-scale IP approach, O-instance regulation approach, Gurobi's built-in feasibility, respectively. Each test point is repeated 10 times using random X_S and X_G such that we can obtain the accurate error bars in Fig. 12.

As shown in Fig. 12, the proposed fast feasibility heuristics always generate better solutions than Gurobi's built-in feasibility heuristics using much less running time. The fast feasibility heuristics are about hundreds of times faster than Gurobi's built-in feasibility heuristics when the number of robots is large.

Considering the different performances of the two approaches, it is worth noting that the small-scale IP approach always yields better solutions than the O-instance regulation approach. Meanwhile, although the polynomial-complexity algorithm shows better time performance whether the environment is square or non-square, the difference is minimal based on the running time of Gurobi's built-in feasibility heuristics (~ 0.01 second). As the number of robots increases, the gap in running time between the two approaches gradually gets smaller, especially in the environment with more obstacle clusters.

We can point out that the time performance of the O-instance regulation approach is in polynomial correlation with the number of robots and the number of vertices $|\bar{V}|$ (ref. Proposition 2). In contrast, the time complexity of the small-scale IP approach is affected by the number of lanes $|\bar{E}|$ in $\bar{G}$ (ref. Equation 14, the dimension of the decision variable is $\frac{1}{2}|\bar{E}| \times 1$), the number of robots has little effect on the running time of the approach. Here, we should notice that $|\bar{E}|$ and $|\bar{V}|$ are correlated with the number of obstacle clusters.

The results suggest that using the proposed fast feasibility heuristics generates better initial solutions for IP models more efficiently than Gurobi's built-in feasibility heuristics in the proposed problem settings. The following simulations replaced Gurobi's built-in feasibility heuristics with the small-scale IP heuristic to simplify the implementation without sacrificing completeness.

6.3 Performance of DLM algorithm

We conducted simulations of solving the MPP problem in the narrow-lane environment. We compare the DLM algorithm with other existing works, which are, to the best of our knowledge, some of the efficient (sub-)optimal and fast polynomial-complexity solvers for the MPP problem: IP exact, IP 4-way split, IP 8-way split [17], ECBS [20], and PIBT [19].

In the original publication [17], IP k-way split is the temporal divisions of IP exact based on the Network flow approach. Empirically, as k increases, we observe a general trend of reduced running time at the expense of loss of optimality. Because IP 2-way split has a similar optimality performance to IP exact, we choose IP exact, IP 4-way split, and IP 8-way split here.

In terms of fast polynomial-time algorithm, we use PIBT [19] because PIBT is a complete MPP algorithm which resolves the potential deadlock between robots, and it shows in its original publication [19] that it performs well in the dense environment with numerous robots.

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 [20]. In addition, algorithm comparisons in [22] and our observations confirm the conclusion about ECBS $w = 1.5$ above. Table 1 leads to the conclusion first that the solving speed of the proposed DLM algorithm is dozens of times faster than that of IP exact algorithm, which is the global version of global MPP algorithm without heuristics, and it can even solve large-scale instances that IP exact algorithm cannot solve.

Later, extensive simulations are launched for general conclusions. 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 inter-robot collision. Recalling the mathematical model in Sect. 4.2.3, the objectives are involved with the edge weight W. The length of the lane decides the edge weight. Thus, we want to verify if the lane length will influence the proposed algorithm's performance. We have built two grid narrow-lane environments to demonstrate the comparative algorithm performance in environments with different sizes and lane lengths: one square environment in size of 21×21 with 25 units of 3×3 obstacle clusters, another rectangle environment in size of 19×43 with 36 units of 2×6 obstacle clusters. The corresponding simulation environment for each comparative analysis can be seen in the top left of Figs. 13, 14, 15 and 16. Here, for all the simulations, the start and goal positions are uniformly randomly sampled in the narrow-lane environment with 21×21 grids and 3×3 obstacle clusters. Each test point is repeated 10 times using random X_S and X_G such that we can obtain the accurate error bars in Figs. 13, 14, 15 and 16.

6.3.1 MinTotalDist problem

To demonstrate the DLM algorithm's efficiency in solving the MinTotalDist problem, we compare the DLM algorithm with five other existing MPP algorithms. Between 5 and 50 robots are attempted in the 21×21 narrow-lane environment, the evaluation results are shown in Figs. 13 and 14. The evaluation results demonstrate that the DLM algorithm is the most scalable and efficient among the six because it generates high-quality solutions using reasonable running time, although the number of robots grows.

In the case of 21×21 with 3×3 obstacle cluster size, when the number of robots grows to 35, the DLM algorithm is about 10 times faster than the IP 8-way split and holds a better optimality ratio. In terms of optimality ratio, when the number of robots reaches 40, a useful reduction of around 15% is found in the DLM algorithm compared to the IP 8-way split. In addition, IP exact, IP 4-way split, and IP 8-way split algorithms can no longer finish their calculation in 10^3 seconds after the number of robots exceeds 40. Although IP exact and IP 4-way split have better optimality ratios, their running time is too long relative to the DLM algorithm and lacks scalability. Compared with the fast MPP algorithms ECBS $w = 1.5$ and PIBT, the DLM algorithm always maintains a better optimality ratio by sacrificing reasonable solving speed: when the number of robots reaches 50, a significant of around 51% compared to PIBT and around 48% compared to ECBS, in terms of optimality ratio.

In the case of 19×43 with 2×6 obstacle cluster size, the comparative performance has a similar trend. The DLM algorithm always has a better running-time performance than the other IP-based MPP algorithm. When the number of robots reaches 35, the DLM algorithm reduces the optimality ratio by around 5% with a running-time reduction of around 95% compared to the IP 8-way split (IP exact and IP 4-way split cannot work within limited running time). Compared to the fast algorithms, when the number of robots reaches 50, a significant reduction of around 40% and 42% compared to ECBS and PIBT, respectively, in terms of optimality ratio, at the reasonable running time cost.

6.3.2 MinMaxDist problem

In the second comparative evaluation, we switch to the case of solving the MinMaxDist problem. The evaluation results are shown in Figs. 15 and 16, which show similar performance trends as the MinTotalDist cases. In the case of 21×21 with 3×3 obstacle cluster size, compared with the other IP-based algorithms, the DLM algorithm provides more than 10 times speed up, even provides solutions with a lower optimality ratio than the IP 4-way split and IP 8-way split: When the number of robots reaches 30, the DLM algorithm reduces the optimality ratio by around 6% compared to IP 4-way

Table 1 Simulation results obtained for solving MinMaxDist problem in the warehouse with 3 × 3 obstacle clusters using DLM algorithm and IP exact algorithm

No. of obstacle clusters	No. of robots	Start position	Goal position	Path length	Running time (Sec)	IP exact (Sec)
9	4	(2,9) (13,7) (6,13) (1,3)	(2,13) (1,11) (10,9) (1,8)	06 16 14 05	0.0330	0.9482
25	16	(3,17) (3,9) (7,17) (9,14) (5,14) (9,16) (7,13) (18,17) (20,13) (14,21) (21,2) (1,7) (21,16) (20,17) (12,21) (9,10)	(1,16) (13,15) (5,7) (14,21) (9,15) (6,5) (18,13) (2,13) (5,3) (10,5) (1,12) (5,15) (13,10) (15,21) (2,17) (21,14)	13 16 20 12 07 24 19 28 27 26 30 24 16 21 24 24	0.1694	4.1047
36	32	(11,5) (1,6) (5,14) (24,25) (23,17) (1,24) (5,20) (4,5) (16,5) (8,21) (17,2) (1,7) (25,20) (12,23) (20,25) (25,19) (9,23) (17,19) (5,16) (16,25) (19,21) (20,1) (8,13) (6,1) (7,17) (7,21) (1,16) (10,1) (25,12) (21,11) (6,5) (9,16)	(21,10) (24,1) (9,12) (4,1) (13,7) (22,25) (1,23) (1,14) (15,13) (21,2) (9,2) (18,5) (1,8) (25,14) (5,8) (5,6) (1,16) (17,15) (1,2) (5,3) (18,17) (17,14) (21,12) (10,25) (8,9) (17,6) (23,17) (5,16) (11,21) (17,11) (10,13) (25,8)	19 34 12 44 20 36 07 42 33 40 16 29 42 22 32 41 31 04 20 33 09 38 34 44 13 43 41 30 43 08 22 35	0.8314	NaN[1]

[1]NaN: The algorithm cannot solve this problem instance in 10^3 seconds

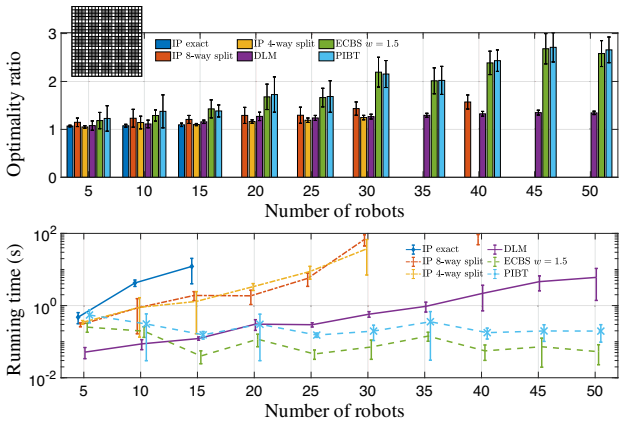


Fig. 13 DLM and other mentioned existing algorithms solve the MinTotalDist problem in the 21 × 21 environment with 3 × 3 obstacle cluster size

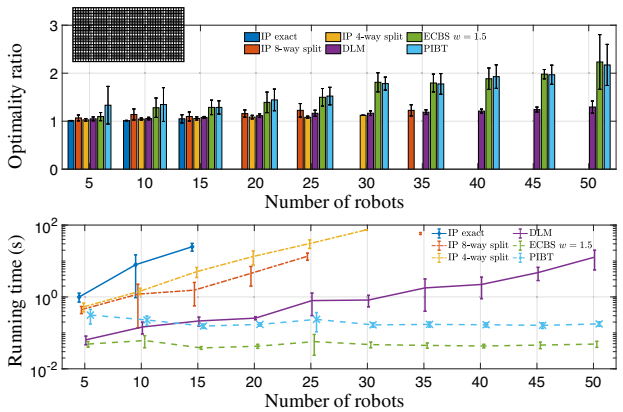


Fig. 14 DLM and other mentioned existing algorithms solve the MinTotalDist problem in the 19 × 43 environment with 2 × 6 obstacle cluster size

split and around 8% compared to IP 8-way split. Meanwhile, in terms of running time, the DLM algorithm reduces around 99% compared to IP 4-way split and around 97% compared to IP 8-way split. Compared with the fast algorithms, the DLM algorithm provides better solutions with a much lower optimality ratio, although the number of robots grows.

In the case of 19 × 43 with 2 × 6 obstacle cluster size, the comparative performance yielded similar results. The DLM algorithm always has a much better running-time performance than the other IP-based MPP algorithm. When the number of robots reaches 40, the DLM algorithm reduces the optimality ratio by around 17% with a running-time reduction of around 99% compared to the IP 8-way split (IP exact and IP 4-way split cannot work within limited running time). Compared to the fast algorithms, when the number of robots reaches 50, a significant reduction of around 20% and 18% compared to ECBS and PIBT, respectively, in terms of optimality ratio, at the reasonable running time cost.

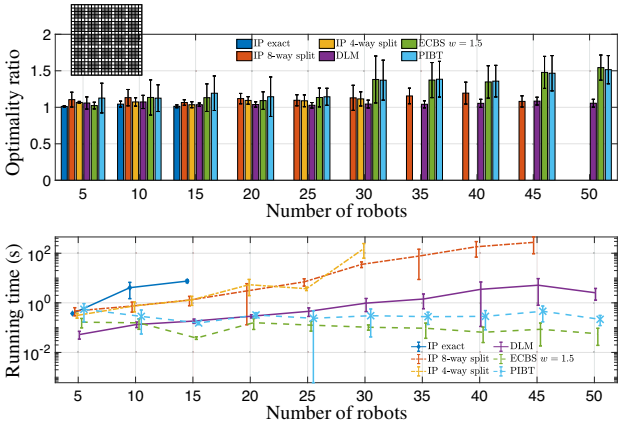


Fig. 15 DLM and other mentioned existing algorithms solve the MinMaxDist problem in the 21 × 21 environment with 3 × 3 obstacle cluster size

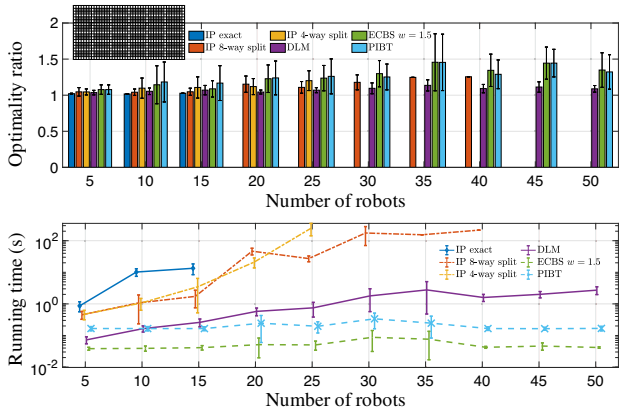


Fig. 16 DLM and other mentioned existing algorithms solve the MinMaxDist problem in the 19 × 43 environment with 2 × 6 obstacle cluster size

6.4 Evaluation summary

The evaluation results demonstrate that the DLM algorithm can solve the collision-free MPP problem in narrow-lane environments with numerous robots, achieve the balance between optimality and running-time performance, and provide safe and efficient paths for robots.

In terms of running-time performance in solving MPP problems in the narrow-lane environment, the proposed DLM algorithm is faster than other IP-based algorithms, significantly faster than IP exact. Referring to the original publication of IP exact and IP k-way split [17], in the process of increasing the value of IP k-way split's parameter k, along with the rapid decline of the optimality performance, the running-time performance of the IP k-way split algorithms has a significant improvement, especially when there are many robots and obstacles. We can find that other IP-based algorithms may fail to work at higher robot density, but the DLM algorithm can always solve.

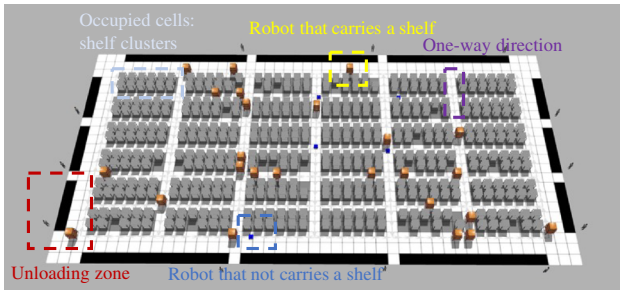


Fig. 17 Bird's view of the narrow-lane environment modeled in Gazebo

In terms of optimality performance, the simulation data also show an interesting trend: the optimality performance of the DLM algorithm has a small fluctuation range, and the rising speed of the running time is also relatively slow compared to IP k-way split [17]. This makes sense because the one-way constraints and topological extraction implemented in the DLM algorithm simplify the collision avoidance in the narrow-lane environment, but the optimal IP approach maintains the optimality. At the same time, IP k-way split searching space expands rapidly as the number of robots increases, due to the increase of the k value, the optimality performance will be lost. Compared with PIBT [19], and ECBS [20], the DLM algorithm achieves a much better optimality performance with a reasonable loss in running-time performance.

7 Warehouse environment simulation

The DLM algorithm is applicable to the automated warehouse. We build a Gazebo-based warehouse environment to verify the DLM algorithm's performance, as shown in Fig. 17. We use ROS (Robot Operating System) [34,35]. Considering the assumption of well-formed infrastructure, we add the nearest occupied cell of s_r and g_r to the head and the end of each collision-free path P_r given by the DLM algorithm. Here, we use the motion controller on each robot to make the robots move along the collision-free paths.

To better demonstrate the control process on each robot, we use Method 1 to describe the motion controller briefly. As we can see in Fig. 19, at each time step, we can query the odometer of the robot for the robot's current position (x, y, yaw) and calculate the distance and angle error from the next position (lines 7–8). Then, the velocity and angular velocity of the robot can be generated by the error (lines 12–13). By the above results, we use the ROS's built-in differential drive control [36] (line 14) to compute the velocity of each robot wheel and control the robot to follow the scheduled collision-free path. The simulation results show that the paths given by the DLM algorithm can make the robot perform the task safely[2].

Method 1 Motion controller on each robot

Input: $TB_r, r, P_r, V_d^{max}, V_y^{max}, k_d, k_y$
Output: TE_r
1: $step = 0$ ▷ Terminal sends the task begin signal TB_r.
2: **while** TB_r **do**
3: $(x, y, yaw) = Query\_Odometer(r)$ ▷ Get the current position.
4: **if** (x, y) coincides with the end vertex of P_r **then**
5: $TE_r = True, break$ ▷ Reach at the goal position.
6: **end if**
7: $(x_t, y_t) = p_r^{step}$, $yaw_t = \arctan(\frac{y_t - y}{x_t - x})$ ▷ Next position p_r^{step+1}.
8: $err_y = yaw_t - yaw, err_d = \sqrt{(x_t - x)^2 + (y_t - y)^2}$
9: **if** $err_d < 0.1$ **then**
10: $step = step + 1$ ▷ Turn to the next position.
11: **end if**
12: $V_d = k_d \cdot err_d, V_y = k_y \cdot err_y$ ▷ Determine the velocity.
13: $V_d = \min(V_d, V_d^{max}), V_y = \min(V_y, V_y^{max})$ ▷ Limit the velocity.
14: $Differential\_Drive(r, V_d, V_y)$ ▷ Send the control signal to robot.
15: **end while**
16: **return** TE_r ▷ Send the task end signal TE_r to the terminal.

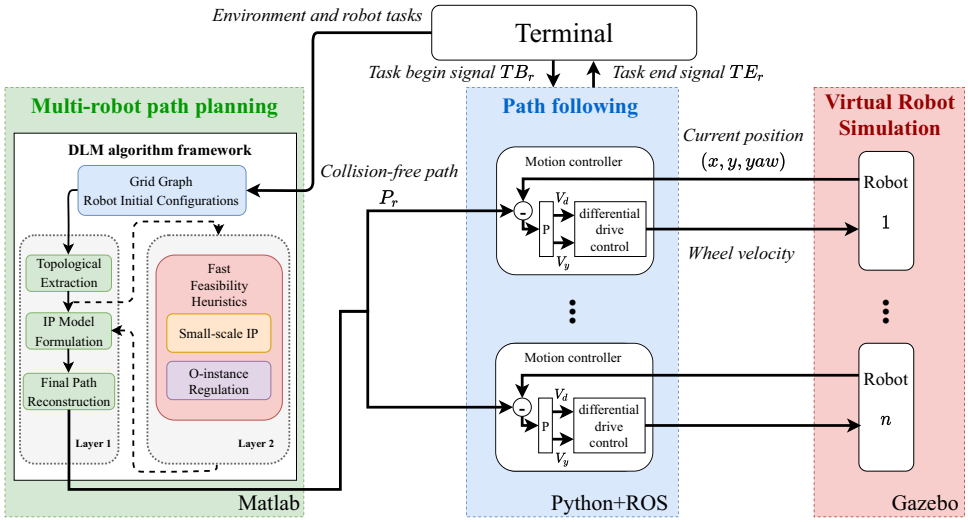
8 Conclusions and discussions

Combining the advantages of the IP approach and the fast feasibility heuristics, this work affords the efficient DLM algorithm to address the MPP problem in narrow-lane environments, guaranteeing collision-free paths. Overall, this work has both academic and industrial implications.

From the academic perspective, the proposed DLM algorithm achieves the trade-off between computational efficiency and optimality performance in solving the MPP problem without deadlock in the narrow-lane environment. By simulation results, we show that the DLM algorithm can carry higher overall performance using reasonable running time compared with existing algorithms such as PIBT, ECBS, IP exact, and IP k-way split in the proposed environment. While the preliminary new release of the DLM algorithm shows promising performance by simulation results, its applicability is limited to the narrow-lane environment.

From the industrial perspective, the proposed DLM algorithm can provide the robots with a collision-free path set. The robots can perform goods-transfer tasks safely and efficiently in the narrow-lane environment, as we can see in the algorithm verification of the warehouse environment simulation. Therefore, we can significantly achieve personnel

[2] Please see the attached video for the warehouse simulation results of the DLM algorithm.



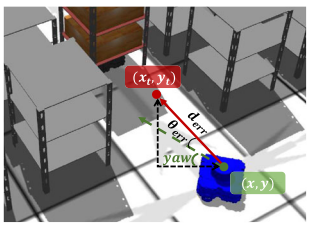


Fig. 19 Green elements are the robot's current location and orientation, and red elements are its next position and orientation

savings and improve warehouse productivity to meet the E-Commerce industry expansion.

The prospects of the DLM algorithm encompass multi-robot path planning in a more complex environment. The proposed algorithm considers the optimal objectives based on distance metrics. While in the case of a multi-robot system with a massive range of robots, the simple strategy at the robotic level may cause serious waiting, which reduces the algorithm efficiency. Thus, adapting the DLM method to an approach with temporal and spatial optimization based on both distance and time metrics would be pragmatic in the coming time and the future extension of this work. Also, the DLM algorithm with the dynamic traffic policies may further boost performance to solve the MPP problem in more complex situations. Later, we will build the physical multi-robot algorithm verification platform using actual robots in future works.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s11370-022-00436-4.

Funding The work was supported by the Natural Science Foundation of China under Grants 61873235 and 62173294, the NSFC-Zhejiang Joint Fund for the Integration of Industrialization and Informatization under Grant U1909206, the Fundamental Research Funds for the Zhejiang Provincial Universities under Grant 2021XZZX014, the Fundamental Research Funds for the Central Universities under Grant xtr072022001.

Declaration

Conflict of interest The authors declare that they have no conflict of interest.

Ethics approval All the authors mentioned in the manuscript have agreed for authorship, read and approved the manuscript, and given consent for submission and subsequent publication of the manuscript.

Consent to participate Not applicable.

Consent for publication All the authors mentioned in the manuscript have agreed to the publication of the manuscript.

Code availability The MATLAB and Python codes used and analyzed during the study are available from the first author on reasonable request.

References

1. Sharma K, Doriya R (2021) Coordination of multi-robot path planning for warehouse application using smart approach for identifying destinations. Intel Serv Robot 14(2):313–325
2. Rodriguez S, Amato NM (2010) Behavior-based evacuation planning. In: 2010 IEEE international conference on robotics and automation. IEEE, pp 350–355
3. Ren W, Sorensen N (2008) Distributed coordination architecture for multi-robot formation control. Robot Auton Syst 56(4):324–333
4. Alonso-Mora J, Baker S, Rus D (2017) Multi-robot formation control and object transport in dynamic environments via constrained optimization. Int J Robot Res 36(9):1000–1021
5. Griffith EJ, Akella S (2005) Coordinating multiple droplets in planar array digital microfluidic systems. Int J Robot Res 24(11):933–949
6. Weidinger F, Boysen N, Briskorn D (2018) Storage assignment with rack-moving mobile robots in kiva warehouses. Transp Sci 52(6):1479–1495
7. Singer S. Helped by robotics, Amazon expands windsor facility. https://www.courant.com/business/hc-biz-amazon-tech-warehouse-20180524-story.html

8. Dijkstra EW et al (1959) A note on two problems in connexion with graphs. Numer Math 1(1):269–271
9. Hart PE, Nilsson NJ, Raphael B (1968) A formal basis for the heuristic determination of minimum cost paths. IEEE Trans Syst Sci Cybern 4(2):100–107
10. Erdmann M, Lozano-Perez T (1987) On multiple moving objects. In: IEEE International conference on robotics and automation
11. Lavalle SM, Hutchinson SA (2002) Optimal motion planning for multiple robots having independent goals. In: Proceedings of IEEE international conference on robotics and automation
12. Yu J, Lavalle SM (2015) Optimal multi-robot path planning on graphs: structure and computational complexity. Comput Sci
13. Li J, Ran M, Xie L (2020) Efficient trajectory planning for multiple non-holonomic mobile robots via prioritized trajectory optimization. IEEE Robot Autom Lett 1–1:99
14. Huang G, Cai Y, Liu J, Qi Y, Liu X (2021) A novel hybrid discrete grey wolf optimizer algorithm for multi-UAV path planning. J Intell Rob Syst 103(3):1–18
15. Surynek P (2010) An optimization variant of multi-robot path planning is intractable. In: Proceedings of the AAAI conference on artificial intelligence, vol 24
16. Schouwenaars T, De Moor B, Feron E, How J (2001) Mixed integer programming for multi-vehicle path planning. In: 2001 European control conference (ECC). IEEE, pp 2603–2608
17. Yu J, LaValle SM (2016) Optimal multirobot path planning on graphs: complete algorithms and effective heuristics. IEEE Trans Rob 32(5):1163–1177
18. Sharon G, Stern R, Felner A, Sturtevant NR (2015) Conflict-based search for optimal multi-agent pathfinding. Artif Intell 219:40–66
19. Okumura K, Machida M, Défago X, Tamura Y (2019) Priority inheritance with backtracking for iterative multi-agent path finding. arXiv preprint arXiv:1901.11282
20. Barer M, Sharon G, Stern R, Felner A (2014) Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem. In: Seventh annual symposium on combinatorial search
21. Guo T, Han SD, Yu J (2021) Spatial and temporal splitting heuristics for multi-robot motion planning. arXiv preprint arXiv:2103.14111
22. Han SD, Yu J (2020) Ddm: fast near-optimal multi-robot path planning using diversified-path and optimal sub-problem solution database heuristics. IEEE Robot Autom Lett 5(2):1350–1357
23. Wang KHC, Botea A et al (2008) Fast and memory-efficient multi-agent pathfinding. In: ICAPS, pp 380–387
24. Cohen L, Uras T, Koenig S (2015) Feasibility study: using highways for bounded-suboptimal multi-agent path finding. In: Eighth annual symposium on combinatorial search
25. Bolu A, Korçak Ö (2019) Path planning for multiple mobile robots in smart warehouse. In: 2019 7th International conference on control, mechatronics and automation (ICCMA). IEEE, pp 144–150
26. Gurobi Optimization, LLC: Gurobi optimizer reference manual (2021). https://www.gurobi.com
27. Ralphs T, Güzelsoy M (2006) Duality and warm starting in integer programming. In: The proceedings of the 2006 NSF design, service, and manufacturing grantees and research conference (2006)
28. Chen Y, Wang F, Ma Y, Yao Y (2019) A distributed framework for solving and benchmarking security constrained unit commitment with warm start. IEEE Trans Power Syst 35(1):711–720
29. Koenig N, Howard A (2004) 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, pp 2149–2154
30. Čáp M, Vokřínek J, Kleiner A (2015) Complete decentralized method for on-line multi-robot trajectory planning in well-formed infrastructures. In: Twenty-fifth international conference on automated planning and scheduling
31. Han SD, Yu J (2019) 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, pp 1890–1897
32. Nguyen V, Obermeier P, Son TC, Schaub T, Yeoh W (2019) Generalized target assignment and path finding using answer set programming. In: Twelfth annual symposium on combinatorial search
33. Cohen L, Uras T, Kumar TS, Xu H, Ayanian N, Koenig S (2016) Improved solvers for bounded-suboptimal multi-agent path finding. In: Proceedings of the 2016 international joint conference on artificial intelligence (IJCAI), pp 3067–3074
34. Stanford Artificial Intelligence Laboratory et al.: Robotic operating system. https://www.ros.org
35. Quigley M, Conley K, Gerkey B, Faust J, Foote T, Leibs J, Wheeler R, Ng AY et al (2009) Ros: an open-source robot operating system. In: ICRA workshop on open source software, vol 3. Kobe, Japan, p 5
36. wiki.ros.org: diff_drive_controller. https://wiki.ros.org/diff_drive_controller. last edited 2022-06-13 15:36:18 by Dr.No