HEHA: Bounded Suboptimal Planning for Constrained Routing and Its Use in Heterogeneous Multi-Robot Exploration of Unknown Environments

Longrui Yang[1,2,*], *Student Member, IEEE,* Yiyu Wang[1,3,*], *Student Member, IEEE,* Jingfan Tang[1,*], *Student Member, IEEE,* Yunpeng Lyu[1], Shizhe Zhao[1], *Member, IEEE,* Chao Cao[4], *Member, IEEE,* and Zhongqiang Ren[1,†], *Member, IEEE*

***Abstract*—This paper considers the path planning problem for autonomous exploration of an unknown environment using multiple heterogeneous robots such as drones, wheeled, and legged robots, with different capabilities to traverse complex terrains. A key challenge there is to intelligently allocate the robots to the unknown areas to be explored and determine the visiting order of those areas subject to traversability constraints, which leads to a large-scale constrained multiple hamiltonian path problem (mHPP) that needs to be quickly and iteratively solved every time when new space is explored. To address the challenge, we propose HEHA (Hierarchical Exploration with Heterogeneous Agents) using a recent hierarchical method that decomposes the exploration into global planning and local planning. The major contribution in HEHA is its global planning, where we propose a new routing algorithm RCH (Routing with Constraints and Heterogeneity) that can quickly find bounded sub-optimal solutions for mHPP to minimize the maximum path length (i.e., makespan) among the agents subject to traversability constraints. Additionally, the local planner in HEHA also considers heterogeneity to avoid repeated and duplicated exploration among the robots. The experimental results show, in the presence of heterogeneous traversability, our RCH can solve mHPP with up to 47% smaller makespan than several baselines, and HEHA can reduce the exploration time by up to 30% compared with existing approaches.**

***Index Terms*—Multi-robot systems, Path planning, Traveling salesman problems, Autonomous exploration**

I. Introduction

Autonomous exploration seeks to navigate one or multiple robots in a bounded unknown environment so as to map the environment as soon as possible, which is of fundamental importance in robotics and arises in applications such as underground mining [1] and planetary exploration [2]. This paper considers autonomous exploration using a team of heterogeneous robots such as drones, wheeled, and legged robots, which have different capabilities to traverse complex terrains. For example, wheeled robots cannot pass stairs while drones and legged robots can do so (Fig. 1). In particular, we consider that all robots always stay in communication, and focus on the planning aspects of the problem: how to quickly plan paths for the robots to minimize the exploration time while considering their heterogeneous terrain traversability.

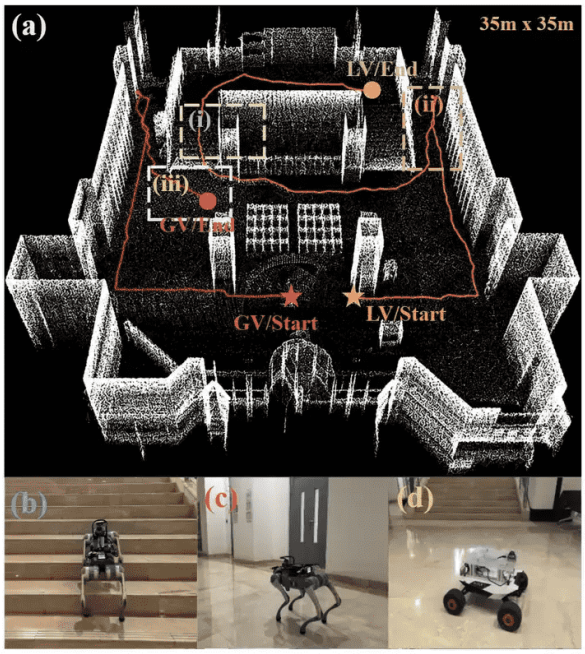


Fig. 1. Exploration using a ground vehicle (GV) and a legged vehicle (LV). (a) shows the results after exploration where the orange and red lines show the robots' trajectories. (b-d) are images of the real world for the areas (i-iii) in (a). The stairs in area (i) as shown in (b) are first found by GV but is only accessible by the LV. Our planner considers such traversability constraints during path planning. Our video attachment shows the exploration process.

Autonomous exploration involves various challenges such as mapping [3], [4], planning [5], [6] and coordination [7], [8]. While exploration problems have been extensively investigated recently, little of the existing work considers heterogeneous

This work was supported by the Natural Science Foundation of China under Grant 62403313 and the Natural Science Foundation of Shanghai under Grant 24ZR1435900. **Longrui Yang, Yiyu Wang, and Jingfan Tang contributed equally to this work. †Corresponding author: Zhongqiang Ren (email: zhongqiang.ren@sjtu.edu.cn).*

[1]Longrui Yang, Yiyu Wang, Jingfan Tang, Yunpeng Lyu, Shizhe Zhao, and Zhongqiang Ren are with the Global College, Shanghai Jiao Tong University, Shanghai 200240, China (email: tang_jingfan@sjtu.edu.cn; lv_hn_promising@sjtu.edu.cn; shizhe.zhao@sjtu.edu.cn; zhongqiang.ren@sjtu.edu.cn).

[2]Longrui Yang is also with Nanyang Technological University, Singapore 639798 (email: longrui001@e.ntu.edu.sg). This work was done while Longrui Yang was a research assistant with Shanghai Jiao Tong University.

[3]Yiyu Wang is also with the University of Michigan, Ann Arbor, MI 48109, USA (email: wangyiyu@umich.edu).

[4]Chao Cao is with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA (email: ccao1@andrew.cmu.edu).

traversability. The major challenge is that the planner has to consider the allocation of robots to the unknown areas to be explored as well as the visiting order of those areas subject to traversability constraints, which leads to a large-scale constrained optimization problem that needs to be quickly and iteratively solved every time when new space is explored.

To address the challenge, we propose HEHA, a robotic system for Hierarchical Exploration with Heterogeneous Agents, which builds upon the common frontier-based approaches [9] and leverages a recent hierarchical framework [5], with the major difference in its planning components. Specifically, frontiers are the boundaries between explored and un-explored spaces, and the planner needs to iteratively plan paths for the robots during exploration until the frontiers deplete. The recent hierarchical framework [5] decomposes the problem into global and local planning: With full global communication, the global planning divides and groups frontiers into clusters, and then formulates and solves a multiple Hamiltonian Path Problem (mHPP) to allocate to and specify the visiting order of these clusters of frontiers for each robot, where each node in the mHPP graph corresponds to a cluster in exploration. The local planning then finds a detailed local path for each robot subject to kinodynamic constraints to visit all frontiers in the assigned clusters for exploration.

Within HEHA, the **first** and major technical contribution is its global planning, where we propose a new framework for mHPP called RCH (Routing with Constraints and Heterogeneity) that can quickly return bounded sub-optimal solutions to minimize the maximum path length (makespan) among the robots, while satisfying traversability constraints that are described as node-robot *assignment constraints* in the mHPP graph. Although mHPP is a well-known NP-hard problem that has been researched a lot, most of the existing work does not consider assignment constraints, and focuses on minimizing the sum of path lengths (min-sum) of the robots as opposed to minimizing the maximum (min-max) as RCH does.

To name a few, to find solutions with optimality or bounded sub-optimality guarantees, integer programming methods for traveling salesmen problems (TSP) and vehicle routing problems (VRP) usually optimize the min-sum objective and are limited to small-scale problems [10], [11]. Approximation methods [12], [13] convert the problem to simpler ones (such as minimum spanning trees) and then build solutions to the original problem with post-processing in polynomial time. However, these methods depend on specific structure of the problem and cannot be easily extended to handle assignment constraints or min-max objectives. Heuristic methods build an initial solution and then iteratively refine this solution, which often run fast and can be adapted to handle various constraints, yet lack solution quality guarantees [14]–[16].

In exploration and other applications [5], [17] with the need of "real-time" planning with limited on-board computational resources, solvers with heuristic algorithms such as PyVRP [16] and Google OR-Tools [15] are often applied. However, both of them are designed for min-sum objectives. Simply applying them to handle the min-max problem with assignment constraints as required in this work, can either lead to infeasible paths due to the omission of the assignment constraint or longer exploration time due to the min-sum optimization. Fundamentally, the mHPP suffers from the curse of dimensionality which grows exponentially as the number of robots and graph nodes increases.

To address the challenge, we propose RCH, a graph search framework with completeness and solution bounded sub-optimality guarantees, which is fundamentally different from existing TSP or VRP methods. Conventionally, for TSP and VRP problems, search methods suffer from a large branching factor and are under-explored. We take the view that by designing a framework that can fuse existing search techniques, which are often studied in other problem domains very different from TSP/VRP, in a principled and novel way, the mHPP problem can be potentially quickly solved with solution quality guarantees.

With that in mind, our RCH runs systematic A*-like search over the state space to incrementally construct paths, as guided by new heuristic functions tailored for the problem which relax and provide lower bounds to the remaining problems, while using new dominance rules to prune non-promising partial solutions to save computational effort. During the search, after obtaining each complete solution, that solution is guaranteed to be bounded sub-optimal, and RCH further leverages a local refinement procedure to post-optimize the solution quality. The systematic search in RCH may also help avoid being trapped into a local minimum after local optimization. We regard RCH as a framework since different node selection and pruning criteria can be employed during the search, various relaxation can be used for heuristic functions, and different local refinement methods can be used for post-optimization.

The **second** technical contribution in HEHA is its local planner, which develops two simple tricks to include heterogeneity into local planning by prioritizing robots of different types based on their traversability. To maintain fast running speed, the local planner in HEHA incorporates the heterogeneity as cost terms when planning detailed motion of the robots to avoid repeated and duplicated exploration among the robots. Such simple rules allow fast and reactive adaptation of robots' motion based on heterogeneity in local areas.

We test our RCH for mHPP, and HEHA for exploration. First, we compare our RCH against several baselines including PyVRP [16] and Google OR-Tools [15] and their fine-tuned versions, on various mHPP instances with assignment constraints, and our RCH finds solutions with up to 47% smaller makespan than the baselines. Second, we compare our HEHA on three maps in simulation against two baselines that adapt the popular next-best-view planner (NBVP) [18] and a greedy method to heterogeneous robots, with full global communication. The results show that HEHA requires up to 30.2% less exploration time and up to 35.7% shorter total path length than the baselines. Finally, we demonstrate HEHA on physical robots, including a wheeled robot and a legged robot in a bounded space of size about $35m \times 35m$ with flat ground and stairs, and HEHA is able to intelligently navigate the legged robot to explore the stair regions while letting the wheeled robot explore the vast flat ground areas to minimize the completion time of exploration.

II. Related Work

A. Autonomous Exploration

Single-robot exploration was studied using frontier-based [9], [19], information theoretic approaches [20], [21], receding horizon sampling [18], traveling salesman methods [5], [22] and learning [23]–[25], to name a few. Multi-robot exploration is usually more challenging than single-robot due to the coordination and interaction among the robots, such as communication limitation [8], [26]–[29], map merging across the robots [4], assignment and ordering the areas to be explored among the robots [30], etc.

Most existing multi-robot exploration considers homogeneous robots, such as wheeled [31], legged [32], or flying robots [7]. There are only a few works on heterogeneous robots, with the focus on using flying robots for coarse survey followed by detailed exploration using ground vehicles [3], [33], [34], or routing with heterogeneous rewards from various sensors on different robots [35]. Although some work builds systems leveraging the traversability of different robots in complex environments [30], [35]–[38], none of them focuses on the planning with heterogeneous traversability.

This work follows the general hierarchical framework for autonomous exploration in [5] where the global planning decomposes the space into regions and determines the visiting order of those regions by solving a routing problem, while the local planning finds a detailed trajectory for the robot to explore each of those regions. When applying this hierarchical method to heterogeneous robots, a major difficulty lies in its global planning, which requires quickly solving an mHPP problem with assignment constraints, and is addressed by this work. We do not consider limited communication among the robots and assume global communication during exploration.

B. Routing Problems with Multiple Robots

For multi-robot exploration [4], [7] and environment reconstruction [39], the coordination of robots is often achieved by iteratively solving mHPP and VRP problems, which are well-known NP-hard problems [40]. These routing problems have been approached by exact methods [10], [41], bounded sub-optimal approximations [12], [13], and unbounded sub-optimal heuristics [14]–[16], trading off solution quality for runtime efficiency. However, most existing routing algorithms optimize the min-sum objective [12], [13], [41], [42] to reduce total energy or distance. While sometimes suitable for applications in logistics [17], [43], min-sum formulations can lead to long makespan and thus prolong the exploration.

Despite these progresses, few of them consider assignment constraints [12], [44] that are necessary to describe heterogeneous terrain traversability of robots in exploration. In particular, transformation methods [41], [43] can handle assignment constraints, but they can only minimize the sum of costs, which can lead to long makespan.

Currently, robotic exploration systems often rely on public routing solvers such as Google OR-Tools [15], PyVRP [16] and LKH [14], which are usually designed for min-sum objectives originally. Although some of them provide options and parameters to configure the solver to indirectly minimize makespan subject to assignment constraints, they usually run slowly or return infeasible or poor quality solutions that still violate the assignment constraints, which eventually leads to poor performance in robotics application.

III. Preliminaries

A. Problem Statement of Exploration

Let $\mathcal{W} \subseteq \mathbb{R}^3$ denote a bounded 3D space to be explored, and let $\mathcal{S} \subset \mathcal{W}$ denote all the surfaces in the workspace to be mapped, which is the boundary between obstacles $\mathcal{W}_{\text{obs}}$ and free space $\mathcal{W}_{\text{free}}$. Let $I = \{1, 2, \cdots, N_a\}$ denote a set of N_a robots, and we use a superscript i over a variable to indicate to which robot that variable belongs. Let $q^i \in \mathrm{SE}(3)$ denote the view pose of the on-board sensor on robot $i \in I$, which can perceive a volume $\mathcal{W}(q^i) \subseteq \mathcal{W}_{\text{free}}$ that is a subset of the obstacle-free workspace. Let $\mathcal{S}(q^i) \subseteq \mathcal{S}$ denote the surface that the robot perceives at pose q^i.

Let $A : \mathrm{SE}(3) \to 2^I$ denote the *capability map*, which maps any pose $q \in \mathrm{SE}(3)$ to a subset of robots $A(q) \subseteq I$ that can reach that pose q. Here, 2^I is the power set of I. This paper assumes A is available for any pose within the perceived volume of any robot during exploration. In practice, additional terrain analysis is needed to get A.

All robots share the same global clock and start their motion at time $t = 0$. Let $\tau^i = \{q^i_1, q^i_2, \cdots, q^i_\ell\}$ denote a trajectory of robot i, where each trajectory point is an element of $\mathrm{SE}(3)$, and any two subsequent trajectory points have a time interval t_Δ in between. A trajectory is *feasible* if it satisfies both the kinodynamic constraints of the robot and *traversability constraints*, that is, $i \in A(q^i_k), k = 1, 2, \cdots, \ell$. Let $\mathrm{cost}(\tau^i) = (\ell - 1)t_\Delta$ denote the finish time of trajectory τ^i, and let $\bigcup_{q^i \in \tau^i} \mathcal{S}(q^i)$ denote the union of all surfaces that are perceived by the robots along τ^i.

Problem 1 (Exploration)**.** *The goal of the exploration problem considered in this paper is to find a set of trajectories $\{\tau^i, i \in I\}$ such that (i) each trajectory τ^i is feasible, (ii) all surfaces are perceived $\bigcup_{q^i \in \tau^i, i \in I} \mathcal{S}(q^i) = \mathcal{S}$, and (iii) the exploration time $\max_{i \in I} \mathrm{cost}(\tau^i)$ reaches the minimum.*

B. System Overview

Our HEHA system follows the same architecture as in [5], which consists of Perception, Global Planning, Local Planning, and Motion Control (Fig. 2). The major difference lies in the planning modules including both the global and local planning. All robots are planned and managed in a centralized fashion in a sense that there is global communication among the robots.

The perception module runs Lidar SLAM to both build a point cloud map of the surrounding environment and estimate the pose of the robot. All robots start from initial poses with known relative transformation so that all robots' point cloud maps can be transformed to the same common frame. The perception also runs terrain analysis to understand the areas that each robot can access, as elaborated in Sec. V-A.

During exploration, the boundary between known and unknown areas forms the frontiers and these frontiers are to be visited by the robots to explore the environment. These

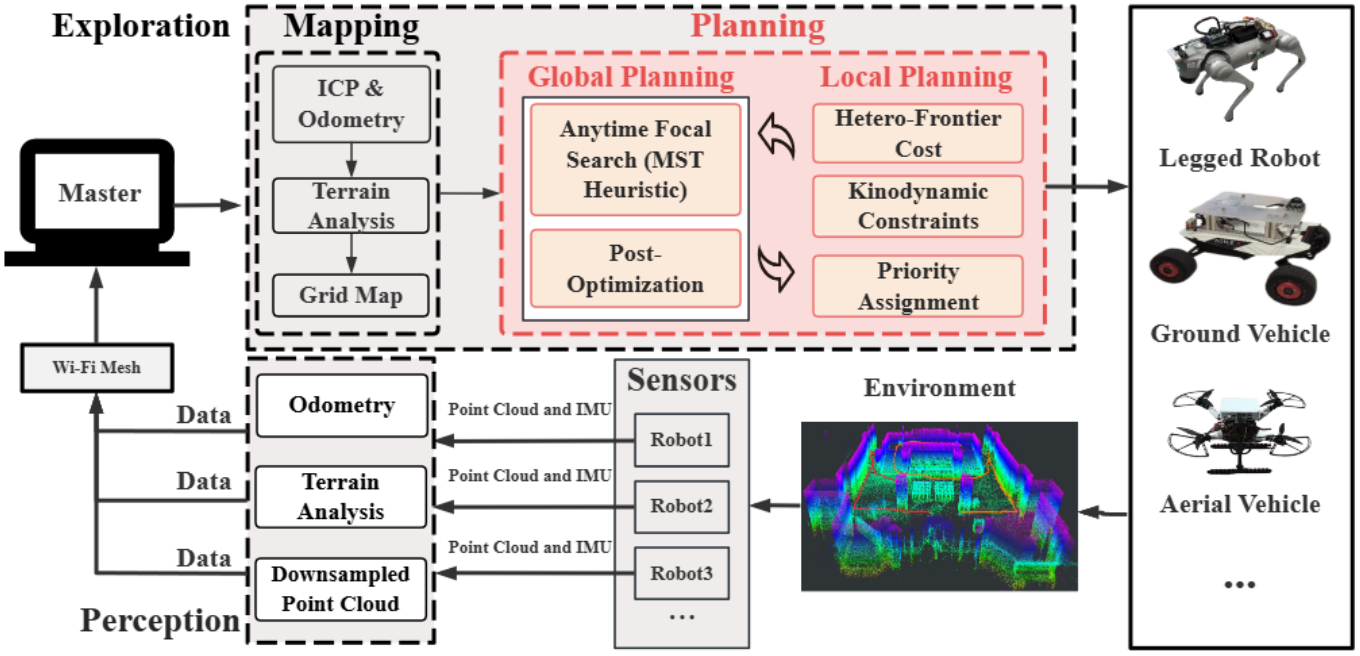


Fig. 2. HEHA System Overview. The red boxed modules, namely global and local planning, are the innovative part of our approach. Other components are similar to the existing hierarchical exploration system [5]. Different types of robots collect point clouds, and IMU measurements through onboard sensors. Then each robot transmits them back to the master computer via the communication network. The system first performs Iterative Closest Point based registration and combines the results with terrain analysis to construct a grid map. Finally, the planning layer conducts global and local path planning.

frontiers are grouped into clusters, and the global planning module determines the visit order of these clusters by solving a routing problem on a graph, in which each graph vertex corresponds to a cluster of frontiers, and each graph edge corresponds to the traversal between two clusters with an estimated traversal cost. Sec. IV presents the global planning.

Given the visit order of clusters, the local planning module (Sec. V-B) plans a detailed trajectory in the region enclosing the frontiers of that cluster, for exploration subject to the robots' kinodynamic constraints. The planned trajectory is then sent to the motion control for execution. The entire process iterates until the environment is fully explored. We detail the local and global planning later in Sec. V-C

IV. Global Planning

A. Problem Statement of Global Planning

Let $G = (V, E)$ denote a complete graph, where $V = \{v_1, v_2, ..., v_{N_v}\}$ is the set of N_v nodes, and E is the set of edges connecting all pairs of nodes. Each node $v \in V$ is associated with a set of agents $A(v) \subseteq I$ that is capable of visiting the node v.[1] These sets $A(v), v \in V$ specify the node-agent assignment constraints in G (Fig. 3(a)). The cost between any two nodes $u, v \in V$ when traversed by robot i is denoted as $c^i(u, v)$. If robot i cannot reach either u or v, then $c^i(u, v)$ is infinity. In G, each agent i has a start node v_s^i and a goal node v_g^i. Each agent i seeks a path π^i, an ordered sequence of nodes in G. The joint path $\pi = \{\pi^1, \pi^2, ..., \pi^k\}$ is the set of all agents' paths. Let $C(\pi^i)$ denote the cost of path π_i, which is $C(\pi^i) = \sum_{(v_k, v_{k+1}) \in \pi^i} c^i(v_k, v_{k+1})$.

[1]Here we abuse the notation A, which is previously used to denote the capability map in the problem definition in Sec. III-A. Both places share the common meaning that A maps either a pose $q \in \mathrm{SE}(3)$ or a node $v \in G$ to a subset of agents that can visit that pose or node.

Problem 2 (Min-Max mHPP)**.** *The Min-Max Multiple Hamiltonian Path Problem (mHPP) seeks a joint path of all agents while minimizing* $\max_{i \in I} C(\pi^i)$*, the maximum path cost among all agents (i.e. makespan).*

Remark 1. *In exploration, each node in G corresponds to a frontier $q \in \mathrm{SE}(3)$ to be visited, and an edge indicates the transition from one frontier to another. The edge cost $c^i(u, v)$ is estimated by solving a small path planning problem from u to v based on the kinodynamic constraints of the robot i. The min-max objective in Problem 2 is aligned with the exploration objective in Problem 1: since all robots explore simultaneously, the exploration is completed when the last robot finishes. Therefore, minimizing the maximum routing cost provides the global planner with an objective directly related to the overall exploration completion time.*

B. Integer Programming Formulation

Similar to many other routing problems, the Min-Max mHPP considered in this work can also be formulated as an integer program (IP). Let binary variables $x_{uv}^i \in \{0, 1\}$ indicate whether robot i traverses the edge (u, v). Let $V_T := V \setminus \{s_i, g_i : i \in I\}$ denote the set of target nodes to be visited, and let $N_T = |V_T|$. Let $u_v \in \{1, \ldots, N_T\}$, $v \in V_T$, be auxiliary variables that indicate the visiting order of the

target nodes. Problem 2 can be formulated as follows.

$$\min \quad \max_{i\in I}\left(\sum_{u,v\in V} c^i(u,v)x^i_{uv}\right) \tag{1}$$

$$\text{s.t.} \quad \sum_{i\in I}\sum_{u\in V} x^i_{uv} = 1, \quad \forall v \in V_T \tag{2}$$

$$\sum_{v\in V} x^i_{s_i v} = 1, \quad \sum_{u\in V} x^i_{u g_i} = 1, \quad \forall i \in I \tag{3}$$

$$\sum_{u\in V} x^i_{u s_j} = 0, \quad \sum_{v\in V} x^i_{g_j v} = 0, \quad \forall i,j \in I \tag{4}$$

$$\sum_{v\in V} x^i_{uv} = \sum_{v\in V} x^i_{vu}, \quad \forall i \in I, u \in V_T \tag{5}$$

$$x^i_{uv} = 0, \quad \text{if } i \notin A(u) \text{ or } i \notin A(v) \tag{6}$$

$$u_w - u_v + N_T \sum_{i\in I} x^i_{vw} \leq N_T - 1, \forall v,w \in V_T, v \neq w \tag{7}$$

$$x^i_{uv} \in \{0,1\}, \quad u_v \in \{1,\dots,N_T\} \tag{8}$$

Constraint 2 ensures each target vertex is visited exactly once by exactly one robot. Constraint 3 ensures each robot must start from its own starting point and end at its own ending point. Constraints 4 prevents any robot from entering a starting point or leaving a goal point, respectively. Constraint 5 guarantees that if a robot enters an intermediate vertex, it must also depart from it. Constraint 6 enforces the node-agent assignment constraints so that a robot i cannot traverse an edge incident to a vertex that it is not allowed to visit. Constraint 7 eliminates disconnected cycles. Constraint 8 states that the edge variables are binary and the order variables are integers. Finally, the objective function seeks to minimize the maximum path costs of all agents and each agent's path cost is the sum of edge costs along its path.

C. RCH Overview

To solve Problem 2, RCH runs a systematic A*-like search that incrementally extends partial solutions (joint paths), one vertex at a time, until all vertices in V are visited and all robots reach their goals. A direct implementation of this search is intractable, since the branching factor grows exponentially in the number of robots.

RCH therefore combines five components. Partial expansion (Sec. IV-E) advances a single robot per expansion, keeping the branching factor independent of N_a. A dominance rule (Sec. IV-D) prunes partial solutions that cannot lead to a better makespan than an already generated one. An MST-based heuristic (Sec. IV-F) lower bounds the makespan of any solution extending a given partial solution, and a focal list (Sec. IV-G) selects among the remaining partial solutions those closest to completion, within a user-specified suboptimality bound. Finally, a post-optimization (Sec. IV-H) procedure refines every solution found during the search.

Among these components, partial expansion [45], focal search [46] and 2-opt [14] are adapted from the literature, where they were developed for problem domains other than mHPP. The components that are new in RCH are (i) the heuristic, which lower bounds the cost-to-go of a partial solution for min-max mHPP under assignment constraints; (ii) the dominance rule, which compares partial solutions by both the cost of each robot and the set of visited vertices; and (iii) the post-optimization, which perturbs the paths within and across groups of robots of the same type.

D. Concepts and Notations

Let $\mathcal{G} = G^{N_a} = G \times G \times \cdots \times G$ denote the *joint graph* of all robots. Our RCH uses A*-like search to iteratively plan paths in $\mathcal{G}$ from the joint start vertex $\vec{v}_s = (v_s^1, v_s^2, \cdots, v_s^{N_a})$ of all robots towards their joint goals $\vec{v}_g = (v_g^1, v_g^2, \cdots, v_g^{N_a})$.

1) Label and Dominance: During the search, let $l = (\vec{v}, \vec{g}, B)$ denote a *label*, where

- $\vec{v} = (v^1, v^2, \cdots, v^{N_a})$ is a joint vertex that represents the current vertices of the robots in this label,
- the cost vector $\vec{g} = (g^1, g^2, \cdots, g^{N_a})$ denotes the current cumulative costs of the robots, and
- the set $B \subseteq V$ is the set of nodes that have been visited.

During planning, each label identifies a unique path in $\mathcal{G}$ that can be reconstructed by iteratively backtracking the *parent pointers* until reaching l_0. We use $\vec{v}(l), \vec{g}(l), B(l)$ to denote the corresponding component in label l. Let $l_0 = (\vec{v}_s, \vec{0}, \emptyset)$ denote the initial label.

There can be multiple paths in $\mathcal{G}$ from v_s to any other $v \in G$, and the planner needs to compare them and potentially discard a less promising one. We introduce the following dominance rule to compare and prune labels, which is similar to some recent work that uses A*-like search to solve (single-agent) TSP or HPP problems [47], [48].

Definition 1 (Label Dominance)**.** *Given two labels $l_1 = (\vec{v}_1, \vec{g}_1, B_1)$ and $l_2 = (\vec{v}_2, \vec{g}_2, B_2)$, we say that l_1 dominates l_2, denoted $l_1 \prec l_2$, if the following conditions hold: (i) $\vec{v}_1 = \vec{v}_2$; (ii) $\vec{g}_1$ is element-wise no greater than $\vec{g}_2$, which is denoted as $\vec{g}_1 \leq \vec{g}_2$; (iii) $B_2 \subseteq B_1$. We also write $l_1 \preceq l_2$ if $l_1 \prec l_2$ and $l_1 = l_2$.*

If label $l_1 \preceq l_2$, then l_2 cannot lead to a better solution path from v_s to v_g since any future path from l_2 can be cut-and-paste to l_1 without worsening the makespan. We prove the correctness of this dominance rule in Lemma 2 in Sec. IV-I.

For two labels l_1, l_2 with $\vec{v}(l_1) = \vec{v}(l_2)$, if neither of them dominates the other, these two labels are non-dominated by each other. At each joint vertex $\vec{v} \in \mathcal{G}$, there can be multiple labels that are non-dominated by each other. To store these non-dominated labels, let $L(\vec{v})$ denote a *frontier set* of non-dominated labels with $\vec{v}(l) = \vec{v}$. During planning, when a new label l' is generated, l' is checked for dominance against existing labels in $L(\vec{v}(l'))$ as explained later.

E. Partial Expansion

Each vertex in the joint graph $\mathcal{G}$ has $(|V|-1)^{N_a}$ successor vertices, and this number is referred to as the branching factor of the search. Naively expanding a label l by generating all labels for those successor vertices is computationally prohibitive as the branching factor grows exponentially in N_a. To bypass this curse of dimensionality, our RCH thus takes a partial

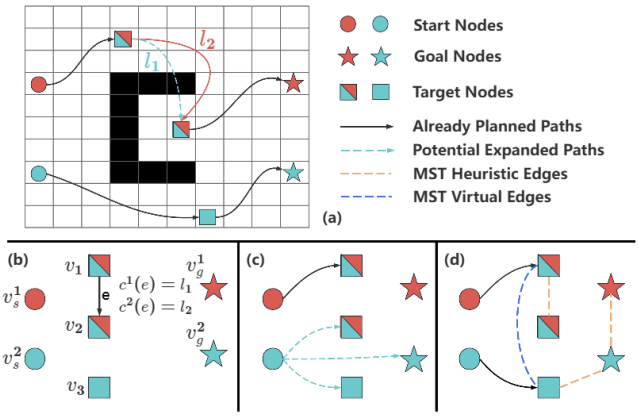


Fig. 3. (a) An example of global planning, where a path in the workspace for a robot corresponds to an edge in the target graph in (b), and the edge cost in (b) is same as the corresponding path length in (a). The color of a target node indicates the assignment constraints. The objective is to find a joint path in the mHPP graph, as in (b), for all robots that minimizes the maximum path cost. (b) The corresponding complete target graph, where v_s^i, v_g^i are the start and goal nodes of robots $i = 1, 2$ and v_1, v_2, v_3 are the target nodes to be visited. (c) Visualization of a search label, which represents a partial solution and its possible successors during expansion. Here, to expand this label, the next moves of robot 2 are considered as shown by the dashed lines, which correspond to a feasible target or goal node. (d) The proposed MST-based heuristic estimates a lower bound of the remaining cost to visit all unvisited targets, where current robot positions are connected by zero-cost virtual edges.

expansion strategy [45], which has been used in Multi-Agent Path Finding problems but not yet been investigated under the mTSP or mHPP settings. Intuitively, partial expansion here plans the next actions of the robots one by one during the search, as opposed to planning a next joint action of all robots together, which helps reduce the branching factor. Such reduction of branching factor comes at the cost of moving one agent each time as opposed to move multiple agents each time in $\mathcal{G}$, which therefore leads to more iterations of search, which is addressed by dominance pruning and focal search as explained next.

Specifically, for partial expansion, let $A_{\text{active}}(l) \subseteq I$ denote the set of robots that have not yet reached their goal nodes. In other words, any "inactive" robot $j \notin A_{\text{active}}(l)$ must have reached their goal, i.e., $\vec{v}^j(l) = v_g^j$. Let $i(l)$ denote the robot in $A_{\text{active}}(l)$ with the smallest cost value in $\vec{g}(l)$. To expand a label l, RCH only considers the possible movement of robot i from $\vec{v}^i(l)$ to another unvisited vertex $u \in V, u \notin B(l)$, while keeping the vertices of all other robots the same. For each generated successor label l', the edge cost value traversed by robot i is added to the corresponding component (in $\vec{g}$, $g^i(l') \leftarrow c^i(u, v) + g^i(l)$, and the vertex u is marked as visited, i.e., $B(l') \leftarrow B(l) \cup \{u\}$).

F. Heuristics

For any label l, let $g_{\max}(l) = \max_{i \in I} \vec{g}^i(l)$ denote the current maximum cost among all robots, which resembles the g-value in regular A* search. To estimate the remaining cost-to-go, we introduce a heuristic function $h(l)$ based on the minimum spanning tree (MST) among all unvisited nodes $V \setminus B(l)$, which is computed with the following steps.

First, for label l, the current nodes of all robots $\vec{v}(l)$ are connected with zero-cost virtual edges, and then a MST is computed to connect all unvisited nodes $V \setminus B(l)$. With these zero-cost edges, the MST computed as explained next will connect all currently unvisited nodes $V \setminus B(l)$ together with the current robot locations.

Second, when computing this MST, for each edge $e \in E$, the edge cost value is selected to be the minimum among all robots $c(e) = \min_{i \in I} c^i(e)$ so that the total cost of the resulting MST is a lower bound on the cost-to-go. If a robot cannot traverse the edge due to assignment constraints, the corresponding cost is infinity. By doing so, the resulting edge cost represents the most "optimistic" traversal cost achievable by any robot.

Third, let $c_{\text{MST}}(l)$ denote the total edge cost of this MST. Recall that our planner minimizes the maximum path costs of the agents. Then, the cost of this MST is divided by the number of active agents $h(l) := \frac{c_{\text{MST}}(l)}{|A_{\text{active}}(l)|}$, which is a lower bound on the cost-to-go from l.

Fourth, to tighten the estimated lower bound, we introduce $f_{\text{est}}(l) = \min_{i \in A_{\text{active}}}(\vec{g}^i(l) + h(l))$, which estimates the smallest total path cost of any active robots. $f_{\text{est}}(l)$ is a lower bound on the makespan of robots in A_{active}.

Finally, we define the f-value of label l as $f(l) = \max\{g_{\max}(l), f_{\text{est}}(l)\}$, which considers both active and inactive robots. This $f(l)$ is a lower bound on the makespan of any solution building upon l. We summarize the properties of this MST heuristic with the following lemma.

Lemma 1. *The proposed heuristic is admissible: the f-value $f(l) = \max\{g_{max}(l), f_{est}(l)\}$ of any label l is a lower bound on any solution that can be obtained by extending l.*

Proof. Consider any feasible complete solution obtained by extending label l, and let $M^\star(l)$ denote its makespan. For each active robot $i \in A_{\text{active}}(l)$, let r_i denote its remaining cost from label l to its goal in this solution. As stated before, the MST's total cost $c_{\text{MST}}(l)$ is a lower bound on the total remaining traversal cost of all active robots: $\sum_{i \in A_{\text{active}}(l)} r_i \geq c_{\text{MST}}(l)$. It follows that

$$\max_{i \in A_{\text{active}}(l)} r_i \geq \frac{1}{|A_{\text{active}}(l)|} \sum_{i \in A_{\text{active}}(l)} r_i \geq \frac{c_{\text{MST}}(l)}{|A_{\text{active}}(l)|} = h(l)$$

Thus, $h(l)$ lower bounds the remaining makespan from l. Let $j \in A_{\text{active}}(l)$ be an active robot with the maximum remaining cost in the above future expansions, i.e., $r_j = \max_{i \in A_{\text{active}}(l)} r_i$. Since $g^j(l) \geq \min_{i \in A_{\text{active}}(l)} g^i(l)$,

$$g^j(l) + r_j \geq \min_{i \in A_{\text{active}}(l)} g^i(l) + h(l) = f_{\text{est}}(l).$$

The final makespan must be at least the final path cost of robot j. It follows that $M^\star(l) \geq f_{\text{est}}(l)$. Moreover, the current maximum cost cannot decrease in any extension, and thus $M^\star(l) \geq g_{\max}(l)$. Combining the above inequalities, we obtain

$$M^\star(l) \geq \max\{g_{\max}(l), f_{\text{est}}(l)\} = f(l).$$

Hence, $f(l)$ lower bounds the makespan of any feasible solution extending l, and the proposed heuristic is admissible. □

Algorithm 1: RCH

```
Initialize l_0 and compute f_0;
Add l_0 to L_O and L_F // L_O, L_F are open and focal lists;
while within time limit do
    while L_F is not empty do
        l ← L_F.pop();
        Remove l from L_O and L_F;
        if IsDom(l) then
            continue
        UpdateAndFilter(l)
        if IsCompleteAndBetter(l) then
            π ← PostOptimize(l);
            break;
        foreach l' in SeqExpand(l) do
            if NotFeasible(l') or IsDom (l') then
                Continue;
            parent(l')← l
            Add l' to L_O and update L_F
    Decrease ε;
return π
```

Algorithm 2: PostOptimize(l)

```
pathgroups ← SortPathGroup(l);
foreach group ∈ pathgroups do
    InnerGroupOpt(group);
improved ← true;
while improved do
    improved ← InterGroupOpt(pathgroups);
foreach group ∈ pathgroups do
    InnerGroupOpt(group);
l' ← GetOptimizedLabel(pathgroups);
return l';
```

G. Focal List

RCH uses focal search to speed up the computation while ensuring bounded sub-optimal solution. Focal search maintains an open list L_O and a focal list L_F. L_O contains all labels to be expanded, where labels are prioritized based on their f-values from the minimum to the maximum. L_F is a subset of L_O containing labels with f-value within $[f_{\text{min}}, (1+\epsilon)f_{\text{min}}]$, where f_{min} is the smallest f-value of labels in L_O. L_F prioritizes labels based on another criterion as explained next, and the search also selects the label with the highest priority in L_F for expansion. L_F prioritizes labels based on the following three values in lexicographic order:

- (i) the number of visited nodes $|B(l)|$,
- (ii) the f-value $f(l)$, and
- (iii) $g_{\text{sum}}(l) := \sum_{i \in I} \vec{g}^i(l)$ the total path cost of all robots in label l.

In other words, for any two labels l_1 and l_2, l_1 has higher priority than l_2 in L_F, if (i) $|B(l_1)| > |B(l_2)|$, or (ii) $|B(l_1)| = |B(l_2)|$ and $f(l_1) < f(l_2)$, or (iii) $|B(l_1)| = |B(l_2)|$, $f(l_1) = f(l_2)$ and $g_{\text{sum}}(l_1) < g_{\text{sum}}(l_2)$. If all three values are equal, ties are broken arbitrarily. Such a focal prefers earlier expansion of labels with more visited nodes.

H. Search Process and Post-Optimization

As shown in Alg. 1, RCH adds the initial label l_0 to the open list L_O and focal list L_F and begins the search. In each iteration, the label l with the highest priority in L_F is selected for expansion and is removed from both L_O and L_F. Before expanding l, l is checked for dominance against labels in $L(\vec{v}(l))$. If l is dominated, l is discarded and the current while iteration ends. Otherwise, l is used to filter $L(\vec{v}(l))$ and then added to $L(\vec{v}(l))$, where filtering means removing any existing labels in $L(\vec{v}(l))$ that are dominated by l. If l visits all nodes and all robots reach their goals $\vec{v}_g$, then a solution is found, which is then post-optimized as explained in the next subsection. After post-optimization, the inner while loop is broken and ϵ decreases so as to find a better solution until time out. Then, l is expanded using the partial expansion as described in Sec. IV-E and a set of successor labels is created. For each of those successor labels l', l' is checked for dominance. Additionally, l' is also checked for feasibility in `NotFeasible` by validating:

- (i) whether l' obeys the assignment constraints and
- (ii) when all robots reach their goals ($\vec{v}(l) = \vec{v}_g$), if there are still any unvisited nodes.

If l' survives these checks, l' is added to L_O for future expansion. The focal list L_F is updated correspondingly so that L_F contains all labels l in L_O such that $f(l) \leq (1+\epsilon) \cdot f_{\text{min}}$, with f_{min} being the minimum f-value of any label in L_O. RCH terminates when the runtime limit is reached.

After finding a feasible solution, RCH runs local optimization (Alg. 2) to further reduce the makespan. Here, the robots are classified into a finite set of types, and robots are grouped by their types. Specifically, if two robots $i, j \in I, i \neq j$ satisfy $i \in A(v), j \in A(v)$ for all vertices $v \in V$, then robots i, j are of the same type. For example, two robots are both legged, then they can visit the same subset of vertices in the graph G.

Each robot's path is then improved as follows. RCH first applies an *inner-group* 2-opt optimization follows to possibly further reduce the makespan. After that, an additional *inter-group* optimization, where the nodes in the longest path are iteratively selected and transferred to the shortest path in another group with the smallest inner-group makespan. Finally, another round of *inner-group* optimization is applied.

I. Analysis

Lemma 2. *For two labels l_1, l_2, if l_1 dominates or is equal to l_2, l_2 cannot lead to a better solution than l_1.*

Proof. We prove this lemma by showing that for any solution path π_2 that extends l_2, a corresponding solution path π_1 that extends l_1 can be constructed and is no worse than π_2. Let $\pi_a(l_1), \pi_a(l_2)$ denote the joint paths of all robots from $\vec{v}_s$ to $\vec{v}(l_1)$ and $\vec{v}(l_2)$, respectively. Let $\pi_b(l_2)$ denote an arbitrary joint path starting from $\vec{v}(l_2)$ that visits all unvisited vertices in $V \setminus B(l_2)$. Thus, the concatenated joint path $\pi_2 = \pi_a(l_2) + \pi_b(l_2)$ is a solution extending l_2. Since $\vec{v}(l_1) = \vec{v}(l_2)$ (i.e., each robot is at the same vertex in l_1 as in l_2), joint paths $\pi_a(l_1)$ and $\pi_b(l_2)$ can be concatenated. Let $\pi_{12} = \pi_a(l_1) + \pi_b(l_2)$ denote the resulting joint path. Since the accumulated cost of each

robot in l_1 is no greater than that of the corresponding robot in l_2, and π_{12} and π_2 share the same continuation $\pi_b(l_2)$, we have $g_{\max}(\pi_{12}) \leq g_{\max}(\pi_2)$. However, since $B(l_1) \supseteq B(l_2)$, some vertices visited in $\pi_b(l_2)$ may have already been visited in $\pi_a(l_1)$, and therefore may be visited more than once in π_{12}.

For each vertex u_k that is visited more than once in π_{12}, if u_k is the last vertex of some robot i's path π^i, then this occurrence of u_k can be removed without increasing $g_{\max}(\pi_{12})$, since π^i is shortened while all other robot paths remain unchanged. Otherwise, let u_{k-1} and u_{k+1} denote the previous and next vertices of u_k in robot i's path π^i. The occurrence of u_k can then be shortcut by directly connecting u_{k-1} and u_{k+1} and skipping u_k. By the triangle inequality, this shortcut does not increase the length of π^i, while all other robot paths remain unchanged. Therefore, the resulting joint path has a $g_{\max}$ value no greater than that before the shortcut. We repeatedly apply this operation until all redundant visits in π_{12} are removed. Let π_1 denote the resulting joint path after all such shortcuts. Since only redundant visits are removed, π_1 still visits all required vertices and is therefore a solution extending l_1. Moreover, $g_{\max}(\pi_1) \leq g_{\max}(\pi_{12}) \leq g_{\max}(\pi_2)$. Thus, for any solution extending l_2, there exists a solution extending l_1 that is no worse. l_2 thus cannot lead to a better solution than l_1 and is pruned. □

Theorem 1. *RCH is complete in a sense that, (i) for an unsolvable mHPP instance, RCH terminates in finite time and reports failure; (ii) for a solvable mHPP instance, RCH returns a feasible solution in finite time.*

Proof. RCH conducts systematic best-first search as A* [46], where in each search iteration the most promising label in FOCAL is selected for expansion, and when expanding a state, *all* possible successor states are considered. During the search, a label l is pruned only when there is another label l' that dominates l. By the definition of dominance (Def. 1) and Lemma 2, such pruning only removes sub-optimal solutions since for any future solution from l, there must exist a corresponding (better) solution from l'. As a result, for a solvable problem instance, RCH never eliminates the only solution joint path if a better path has not been found yet, and terminates the search with a feasible solution. For an unsolvable problem instance, since the graph is finite, there is a finite number of possible joint paths. RCH terminates after enumerating all joint paths and reports failure. □

Theorem 2. *Let g_* denote the cost of the true optimal solution, RCH finds a $(1+\epsilon)$ bounded sub-optimal solution π for any solvable Min-Max mHPP as defined in Problem 2, i.e., $g_{max}(\pi) \leq (1+\epsilon)g_*$.*

Proof. RCH systematically enumerates all paths in the joint graph $\mathcal{G}$, and by Lemma 2, dominance pruning never eliminates a path along an optimal solution. With an admissible heuristic (Lemma 1), the f-value of any label l in OPEN is a lower bound of the true optimal solution cost [46], i.e., $f(l) \leq g_*$ with g_* being the true optimal cost of an optimal solution. The use of focal search [46] allows RCH to select a label l with f-value within range $[f_{\min}, (1+\epsilon)f_{\min}]$ with $f_{\min}$ being the minimum f-value of any label in OPEN. At termination, the popped label l is guaranteed to have $g_{\max}(l) = f(l) \leq (1+\epsilon)f_{\min} \leq (1+\epsilon)g_*$, which ensures the $(1+\epsilon)$ bounded sub-optimality of the returned solution. The use of `PostOptimize` only further improves the solution quality of the popped label and will never increase the cost. □

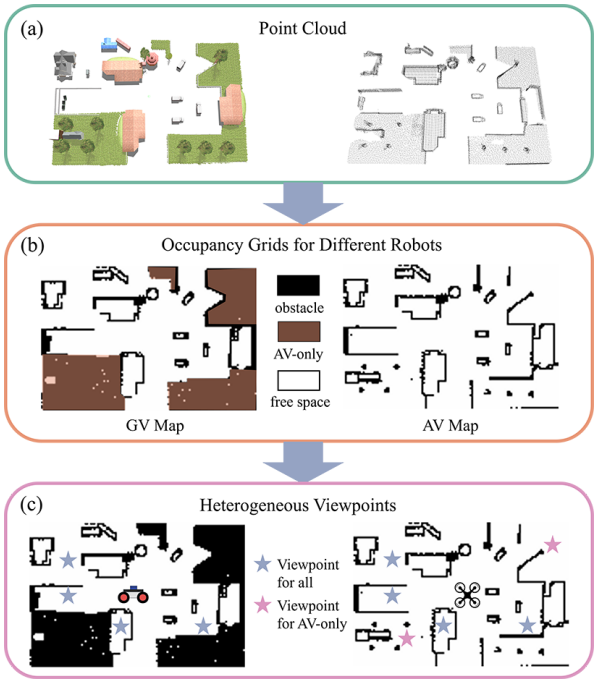


Fig. 4. A visualization of the terrain analysis process in HEHA. Multiple occupancy grids are generated for different types of robots, and viewpoints are sampled for different robots with different traversability, which are then used for the global planning.

V. Other Components in HEHA

A. Terrain Traversability

1) Terrain Analysis: In HEHA, due to the heterogeneous terrain traversability of different robot platforms, different occupancy grids are maintained for different robot types, as shown in Fig. 4. For ground robots, including wheeled and legged robots, these grids encode terrain-dependent traversability, while for UAVs, the corresponding grid mainly reflects the availability of free flying space based on obstacle and height information. All these occupancy grids share the same spatial information about known and unknown cells at any time during exploration. To build the occupancy grids for ground robots, we use a plane-fitting method [49]. To summarize the main ideas, point cloud data is first acquired through SLAM software and uniformly sampled with a fixed spatial interval. For each sampled block, singular value decomposition (SVD) is performed to obtain the dominant direction vectors that describe the local geometric structure of the point cloud. By evaluating the consistency of these normal vectors, points corresponding to the surfaces of the ground or vertical walls are filtered out. Subsequently, the remaining point cloud is analyzed based on its normal distribution to extract regions exhibiting significant inclination or step-like geometry, such as slopes and staircases.

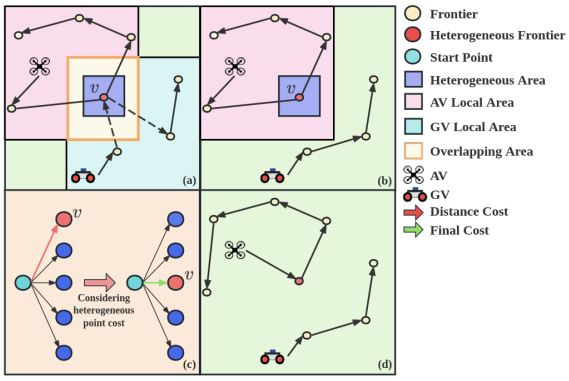


Fig. 5. An illustration of two tricks in Local Planning. (a) shows the set of local viewpoints and the baseline path generated without specialized planning constraints; a specific node v in heterogeneous region (painted as violet) is identified within the overlapping local areas of both robots. (b) shows that v is assigned to the AV due to the high priority of AV. (c) details the initial phase of local planning, where the TSP cost matrix is adjusted to incorporate hetero-frontier cost, encouraging an optimal approach to the viewpoints. (d) displays the final optimized local plan resulting from the cost-adjusted TSP.

The identified regions are then projected onto the respective grid maps of each type of robot. For the wheeled robot, overly steep or uneven areas are marked as occupied cells, representing non-traversable regions for the wheeled robots. In contrast, the same areas are labeled as traversable in the legged robot's grid, since the legged platform is capable of passing through them. We call these areas *heterogeneous regions*, which are areas that are traversable for the legged robot but not for the wheeled robots. Viewpoint sampling is then performed within these heterogeneous regions to generate *heterogeneous viewpoints*, which are poses that are only accessible for certain types of robots. For example, some of those heterogeneous viewpoints can be explored by the legged robot and are inaccessible to the wheeled robots.

2) Failure Handling: During exploration, if no terrain analysis result is received by the planner, then there is no way for HEHA to determine which type of robot can traverse that region. Therefore, the region is treated as an obstacle (i.e., non-traversable region) for all robots. In this region, no free space will be created in the resulting map, and no viewpoints will be generated in that region.

If a non-traversable region is mistakenly classified as traversable, the robot will usually stop its motion as it approaches that non-traversable region due to the local obstacle collision avoidance mechanism, and be able to find no feasible path exists in that region in the subsequent detection-planning cycle. In that case, the viewpoints in that non-traversable region are discarded. In the worst case, if the local collision avoidance fails, then it indicates that there is no way for the robot to identify that non-traversable region and the robot will have to collide or get trapped.

B. Local Planning

Let $D^i \subseteq \mathcal{W}$ denote a bounded space around the robot $i \in I$ for local planning. During exploration, there are often many frontiers near each other; we therefore down-sample these frontiers. Specifically, for a set of frontiers $Q(D^i)$ to be visited by robot i (i.e., $i \in A(q), q \in Q$), HEHA clusters them into subsets $Q(D^i) = Q_1 \cup Q_2 \cup \cdots \cup Q_K$ based on the position of each frontier $q \in Q(D^i)$, so that the frontiers in each $Q_k, k = 1, 2, \cdots, K$ are close to each other. Within each Q_k, a representative frontier q'_k near all other frontiers in Q_k is selected for the robot to visit. Then, similar to [5], a TSP is formulated and solved to obtain a local path π^i_{local} for robot i to visit all these representative frontiers.

Specifically, let $G_L = (V_L, E_L)$, L stands for local, denote a complete graph where V_L is a set of vertices, one for each representative frontier $q'_k, k = 1, 2, \cdots, K$, and $E_L = V_L \times V_L$ is a set of edges that represent the transition from one vertex to another. In G_L, each edge $e \in E_L$ is associated with a non-negative cost value $c(e)$ representing the transition path cost from one representative frontier to another, which is estimated by using Hybrid-state A* [50] to find a kinodynamically feasible trajectory from one frontier to another. Then, a TSP problem can be defined on G_L which seeks a cycle in G_L that visits all vertices and returns to the starting vertex.

When exploring the frontiers in D^i, new frontiers may occur and the robot iteratively replans path locally. Thus, the robot only executes the sub-path to reach the first frontier in π^i_{local} and then a new TSP is formulated and solved, which includes the new frontier if any.

Besides these similar components to [5], HEHA has two additional tricks to handle the heterogeneous robots in the local planning as detailed next.

1) Hetero-Frontier Cost: In the graph G_L for local planning, each vertex in G_L corresponds to either the current pose of the robot (denoted as v_0), or a representative frontier q'_k as aforementioned. Some of the edge costs are adjusted to encourage early exploration of frontiers that can only be visited by a subset of agents due to the terrain and traversability of the robots. A frontier q is a *hetero-frontier* for a robot $i \in I$, if $A(q) \neq I$ and $i \in A(q)$, i.e., q can only be visited by a subset of robots and the current robot i being planned can visit it. When down-sampling the frontiers in each subset Q_k as aforementioned, HEHA also counts the percentage $\alpha \in [0, 1]$ of hetero-frontiers in Q_k. For each edge e from v_0 to some other vertex $v \in G_L$, its edge cost is changed to

$$c'(e) := \max\{c(e) - \alpha \frac{n_h}{n_0}, 0\} \tag{9}$$

where $c(e)$ is the original edge cost of the edge $e = (v_0, v)$, n_h is the number of hetero-frontiers within a certain range (e.g. a circle of radius $r_h \geq 0$ and $r_h = 2$ in our tests) from v, and n_0 is a user-defined non-negative number, which is set to 25 in our tests. By doing so, the edges to the hetero-frontier tend to have lower cost, which thus encourages the robot to visit them as the next step in a soft manner when solving the TSP problem. Here, Eq. (9) is just an empirical way to realize the idea of encouraging early exploration of hetero-frontier and other expressions are also possible.

2) Priority Assignment: When two robots' local planning spaces intersect (i.e., $D^{i,j} := D^i \cap D^j \neq \emptyset$) and there are hetero-frontiers in $D^{i,j}$ (i.e., $\exists q \in Q(D^{i,j})$), then the local planner needs to determine which robot should visit

those hetero-frontiers. HEHA introduces additional priorities $\xi^i, i \in I$ to all robots, and those hetero-frontiers are assigned to the robots with higher priority. In practice, for example, we set the priority of drones to be higher than legged and legged to be higher than wheeled, since drones can traverse all terrains while legged and wheeled can only access a subset of terrains. These priorities help reduce redundant and duplicated exploration by different robots in overlapping local regions and therefore improve the exploration efficiency.

C. Planning Cycles in HEHA

The global and local planners in HEHA operate as follows. First, the local planner is invoked with a fixed rate (once per 2.5 seconds, $0.4-0.5$ Hz). Each invocation of the local planner forms a $L \times L$ (e.g. $L = 7m$) local squared region centered at the current position of the robot and plans a local path for the robot to visit all frontier viewpoints in this region. In addition to this periodic invocation, HEHA also supports condition-triggered replanning through a 2 Hz checking mechanism. For example, if frontier viewpoints in the local region disappear rapidly due to exploration progress or map updates, replanning can be triggered earlier without waiting for the next fixed-period cycle.

Second, when the number of frontier viewpoints in the local square region is below a certain threshold (e.g. when the robots run into a dead-end or when surrounding areas are already explored), or when the local planner fails to plan a path (e.g. due to non-traversable terrains), the global planner is invoked, which formulates a mHPP problem and uses the proposed RCH to quickly solve it. The resulting path can provide the robot with global guidance.

With such designs, in practice, we observed that, at the beginning of the exploration, the local planner is often able to quickly generate paths while locally coordinating with other robots of the same or different types to avoid duplicated exploration of overlapped areas. As the exploration progresses into the middle and late stages, the distribution of frontiers becomes increasingly sparse and distant. Consequently, the local planner fails more frequently, which calls the global planner more frequently. Sometimes, the frequency of calling the global planner gradually increases until it aligns with the local planning frequency (approximately $0.4 - 0.5$ Hz).

In each global planning, an mHPP is formulated based on the current map and frontiers, and the solution to this mHPP is computed from scratch. The fast running speed of our RCH makes such online "real-time" global planning on robots possible.

VI. Experimental Results

We compare our global planner RCH against baselines for mHPP, compare our HEHA against baselines in simulations for heterogeneous multi-robot exploration, and test the robustness and sensitivity of HEHA in simulation. Then, we conduct an ablation study to verify the benefits of the two tricks for the local planning, and finally demonstrate our HEHA with real robots as shown in the video attachment.

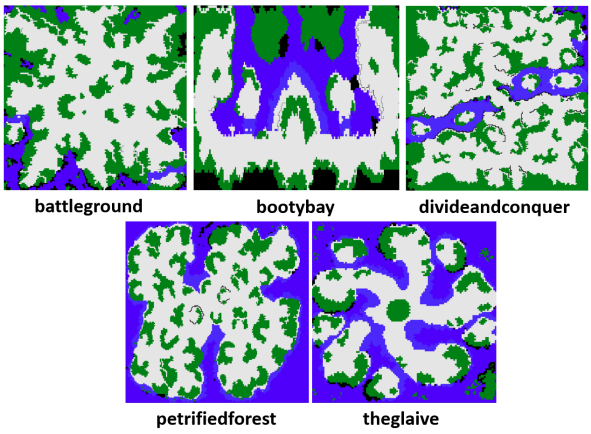


Fig. 6. Five grid maps of size 512×512 from a public dataset, which is used in our experiments to generate mHPP instances.

A. Global Planning and mHPP

1) Maps and Test Settings: We use five grid maps (Fig. 6) of size 512×512 cells from a public dataset [51], where each map is a 2D grid with heterogeneous terrain types, including ground, forest, swamp, water, and obstacle. For simplicity, we denote the five grid maps as M1–M5 in the same order as shown in Fig. 6. We consider two types of robots, ground vehicle (GV) and aerial vehicle (AV). To generate an instance, we sample a few nodes from these grid maps to be visited by the robots, and generate a complete graph, where the edge costs are the shortest path cost between these sampled nodes using A* search in the grid map. These sampled nodes may be located on different terrains, and thus lead to assignment constraints. We classify the nodes to be visited by the robots into two classes: (i) *Normal nodes*, which are sampled from *ground* grids so that both GVs and AVs can visit, (ii) *AV-only nodes*, which are sampled from *swamp* and *water* grids that can only be visited by AVs. The ratio of normal nodes and AV-only nodes is 2:1.

The experiments involve two different settings: *Setting A* has 60 nodes, 3 GVs and 3 AVs; *Setting B* has 150 nodes, 10 GVs and 10 AVs. Both Setting A and B are repeated 30 times, with nodes randomly sampled each time. In our tests, we set a runtime limit of 30 seconds for each planner for each instance.

2) Baselines: We use six baselines (B1-B6).

- B1 is a greedy method that incrementally assigns nodes to the robots subject to assignment constraints while minimizing the current makespan.
- B2 builds upon B1 and further uses the `PostOptimize` (Alg. 2) to refine the solution returned by B1. The comparison between B2 and B1 will show the improvement on solution quality provided by the local optimization idea in the proposed `PostOptimize`.
- B3 uses the transformation method in [41] to convert mHPP to a single-agent TSP on a larger graph G' and then invokes LKH [14], a popular solver for routing problems, to find a solution that minimizes the sum of costs for mHPP. In this transformation method, the transformed graph G' depends on both the number of

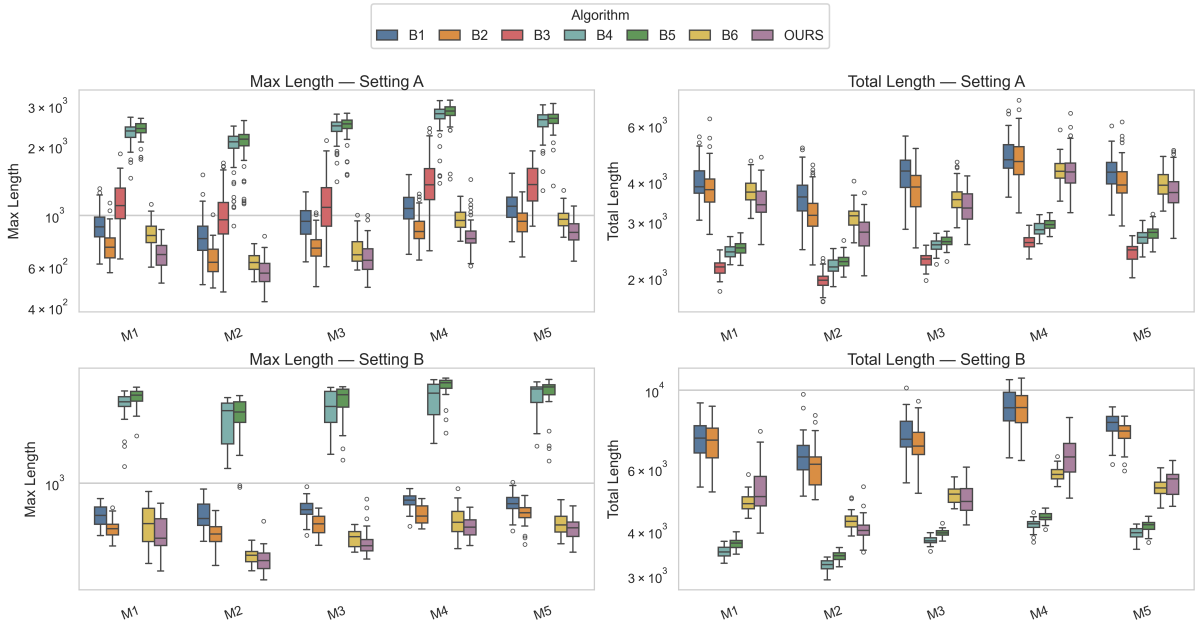


Fig. 7. Makespan (i.e., maximum length of all robots' path) and Sum of costs (i.e., total length of all robots' path) comparison under Setting A and B in five maps. Colors of algorithms (B1–B6 and OURS) are consistent across both settings.

agents and the size of the graph. In our experiments, this method is only able to handle Setting A. For Setting B the graph G' is too big for this method to return any solution within the runtime limit, which is thus omitted from the results as shown next.

- B4 uses another popular solver PyVRP [16] to solve mHPP with assignment constraints while minimizing the sum of costs. In our tests, we set the termination condition of B4 as follows: B4 terminates after 1000 iterations without additional improvement on the incumbent solution.
- B5 uses Google OR-Tools [15] with default parameters, which optimizes the sum of costs.
- B6 is a fine-tuned version of B5, including an augmented objective that encourages balanced workloads and a smaller makespan. Specifically, it adds a penalty term proportional to the difference between the longest and shortest path lengths: Objective $= \sum_{i\in I} L_i + 100 \cdot (\max_{i\in I} L_i - \min_{i\in I} L_i)$, where L_i denotes the path length of robot i. This penalty encourages more balanced workloads among robots and leads to smaller makespan.

The six baselines above can be grouped into three categories according to their optimization objectives and roles in the evaluation. First, B1 and B2 optimize makespan, sharing the same min-max objective as RCH. Second, B3–B5 retain the native min-sum objectives of representative and widely used routing solvers. These baselines are included to assess the consequence of applying existing high-performance min-sum solvers to our problem, where the makespan is the focus. Note that, B3–B5 are not intended as strictly same-objective comparisons with our approach, but serve as practical references for solution quality and computational efficiency. Third, B6 represents adaptation of an existing min-sum solver to the min-max objective. Its comparison with RCH evaluates whether such an adaptation can achieve performance comparable to a solver specifically designed for constrained min-max mHPP.

3) Solution Makespan Comparison: For the experimental results, among the instances, we report their average total cost (avg sum), average makespan (avg max), and average search time (avg time), i.e., the last time point when an algorithm updates the solution within the runtime limit.

As shown in Fig. 7, our RCH always finds solutions with better makespan on average (avg max). Detailed distributions of solution makespans for both settings are shown with box plots in Fig. 7. These results verify that RCH is able to find solutions with a better makespan than the existing methods, which, as shown later, helps improve the completion time of exploration in various environments.

4) Solution Sum of Costs Comparison: We also report the sum of costs (i.e., total length) returned by different planners in Fig. 7. It is not a surprise that the existing planners B3 (LKH transformation) and B4 (PyVRP) find solutions with much smaller sum of costs than our RCH, since they are designed to optimize sum of costs. We visualize the solutions for an instance in Fig. 8. We observe that the solutions with small sum of costs returned by B3 and B4 can lead to very unbalanced path lengths among the robots, resulting in a large makespan. When applied to exploration, these unbalanced solutions can increase the completion time of exploration.

5) Effects of `PostOptimize`*:* We observe that B2 finds a better solution than B1, due to the local optimization, which verifies the benefits of applying `PostOptimize`. Our RCH is better than B2, since RCH conducts systematic search while employing `PostOptimize`, and B2 can get stuck in local minima that is near the initial greedy solution.

6) Runtime Comparison: As shown in Figure 9, B1 and B2 runs the fastest due to their simple greedy strategy. Our RCH is slower than the greedy methods B1 and B2, but is

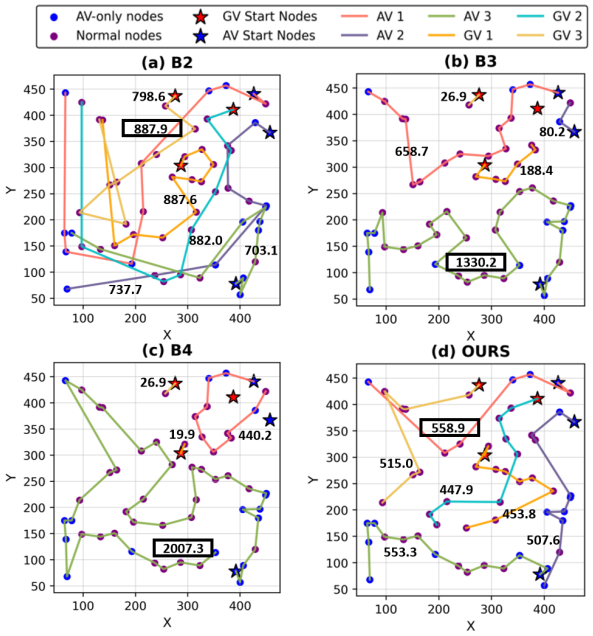


Fig. 8. Visualization the solution paths of baseline B2, B3, B4 and OURS on an instance with Setting A. The boxed number is the makespan. Our RCH can distribute the nodes in a more balanced fashion among the robots and thus achieves smaller makespan.

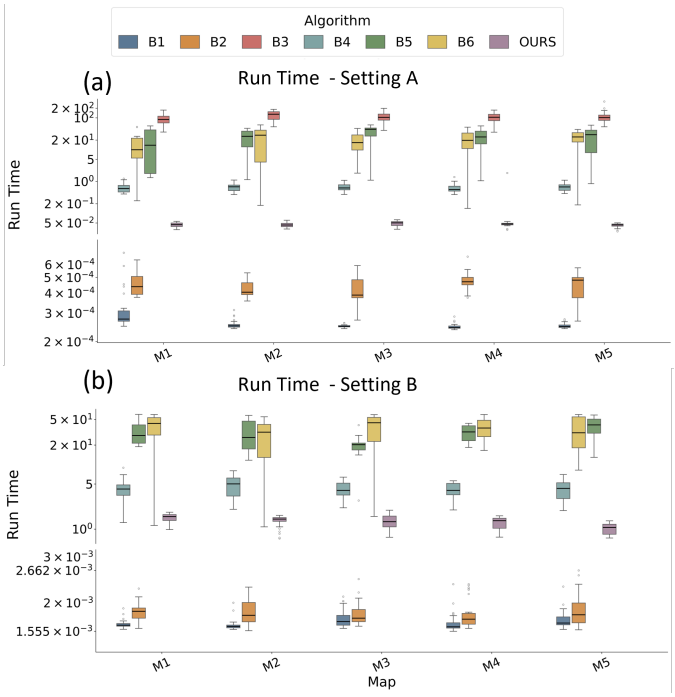


Fig. 9. Time to get final solution under Setting A and B in five maps. Colors of algorithms (B1–B6 and OURS) are consistent across both settings.

often obviously faster than B4, B5 and B6. Recall that the reported runtime of RCH here is the last time point when an algorithm updates the solution within the runtime limit. It indicates that, in practice, RCH is able to find high-quality solutions very quickly. Note that, the runtime reported in Figure 9 is the computational time of the routing solver, rather than the exploration completion time. This metric is

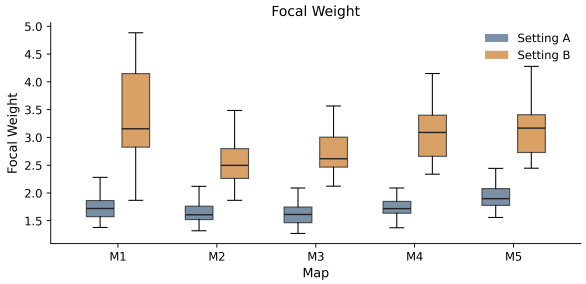


Fig. 10. Focal weight ϵ when RCH terminates under Setting A and B in five maps, which provides a sub-optimality bound of the returned solutions.

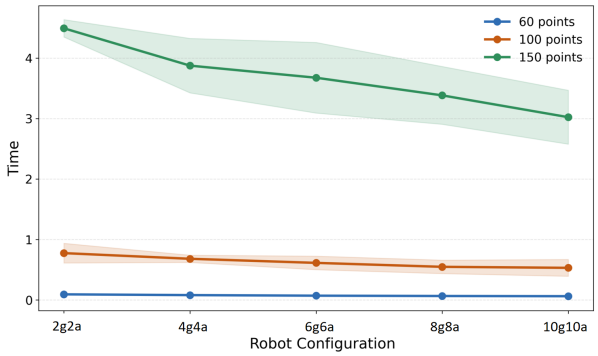


Fig. 11. Average solving time on Battleground under varying problem sizes and robot configurations.

relevant because the global mHPP is repeatedly solved online during exploration, requiring the planner to return high-quality routing solutions quickly. The practical exploration completion time is evaluated separately in Sec VI-B1.

7) Solution Sub-Optimality Bounds: Since RCH is an anytime algorithm where the focal weight ϵ decreases iteratively, we set the initial focal weight ϵ to a large number and report the ending ϵ when RCH terminates in Fig. 10. Note that this weight ϵ bounds the solution sub-optimality. The relatively large values observed are mainly due to the scale of the problem instances in Setting B, where the number of nodes is large, while in Setting A, the bound is relatively tighter.

8) Scalability: To evaluate the scalability of our method, we fix the map as M1 Battleground and vary the number of robots from 2 GVs and 2 AVs (2g2a) to 10 GVs and 10 AVs (10g10a). As shown in Fig. 11, the solving time generally increases with the number of target nodes. However, for a fixed number of nodes, the solving time does not increase (and sometimes decreases slightly) as the number of robots increases. The possible reason for this trend is due to the design of our algorithm: at each step, the search expands the robot with the currently minimum cost. As a result, increasing the number of robots allows the path of one robot to finish early, which then reduces the number of branches and iterations for the future search, resulting in fewer search iterations in total and reducing the solving time.

B. Simulation Experiments on Exploration

1) Comparison with Baselines: We compare our HEHA against two baselines for exploration. The first baseline *Near-*

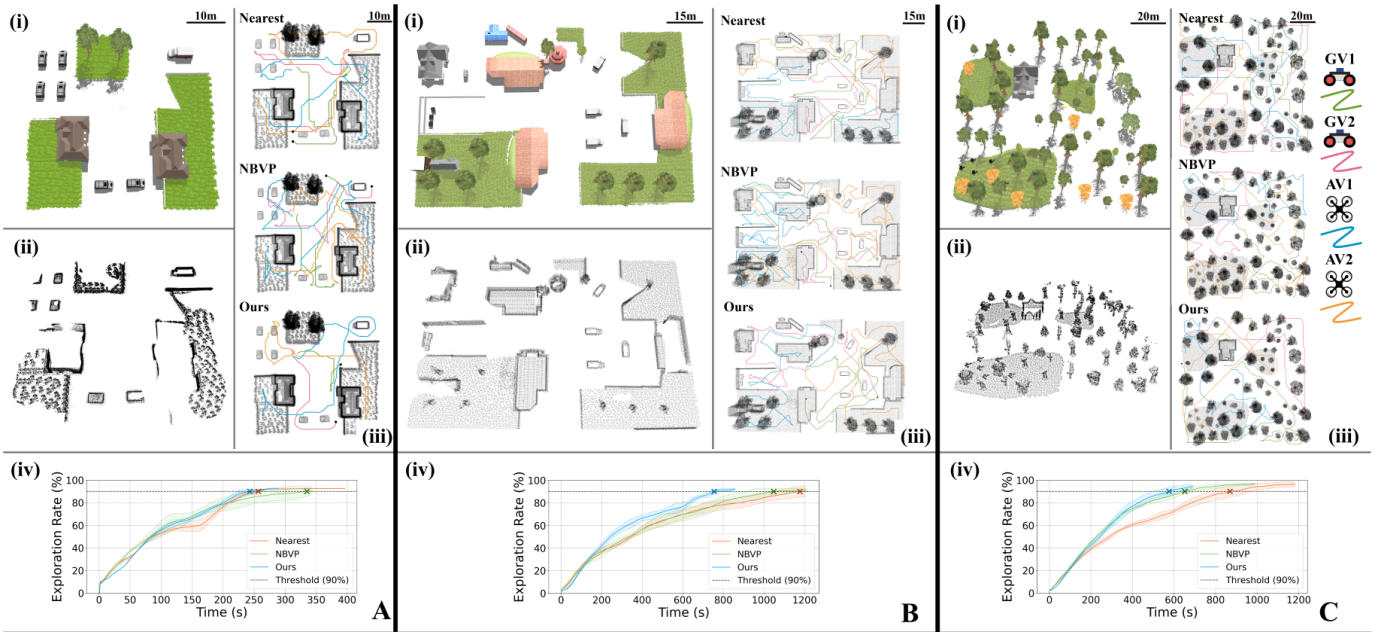


Fig. 12. Exploration in three environments: Garden (A), Village (B), Forest (C). (i) shows the overview of simulation environment. (ii) shows the point cloud map after exploration. (iii) shows the trajectories of the robots using *Nearest*, *NBVP* and Ours. (iv) shows the exploration rates over time.

TABLE I
RESULTS OF EXPLORATION IN SIMULATION IN THREE MAPS SHOWN IN FIG. 12. THE LAST COLUMN MARGIN SHOWS THE PERCENTAGE OF REDUCTION IN EXPLORATION TIME AND PATH LENGTH OF OUR HEHA OVER THE BEST RESULTS AMONG THE BASELINES (NEAREST, NBVP).

Map and Metric	Nearest	NBVP	Ours	Margin
Garden (51m × 51m)				
Time Avg (s)	255.2	284.0	**249.4**	2.3%
Length Avg (m)	480.5	494.1	**309.1**	35.7%
Village (81m × 121m)				
Time Avg (s)	1189.4	1024.6	**715.0**	30.2%
Length Avg (m)	2095.8	1766.4	**1204.5**	31.8%
Forest (101m × 101m)				
Time Avg (s)	880.8	652.4	**572.8**	12.2%
Length Avg (m)	1733.5	1276.6	**1089.3**	14.7%

est adapts the approach in [6] by greedily assigning a robot to the closest frontier it can visit. The second baseline *NBVP* extends [18] to multi-robot, where a Rapidly-exploring Random Tree is built in the free space for each robot with view pose selected to maximize the information gain. For our HEHA, we set $\alpha = 0.6$ for large maps and $\alpha = 0.4$ for small map. We use the same path planning and trajectory tracking modules across all methods. All methods terminate after the exploration rate is over 90% in a similar way to [5], where the exploration rate is the ratio of explored volume over the total volume of the space. In simulation, we have access to the ground truth environment and can thus compute the volume. We stop the exploration with a 90% exploration rate instead of 100% since some space may be inaccessible to any robots.

The experiments are conducted in a customized ROS-based simulation environment where large-scale point cloud maps represent the static world. Rather than using a generic physics engine, we implement a kinematic simulator that performs ray-casting on the point cloud to simulate 3D LiDAR sensing with an effective range of 8m. The robot team consists of two GVs and two AVs. GVs are modeled with differential-drive kinematics, constrained to the ground-level traversable surfaces extracted from the point cloud. AVs are modeled with 3D kinematic mobility, allowing them to fly at a fixed altitude of 3m to bypass ground obstacles. This setup ensures that the heterogeneous traversability and sensing constraints are accurately reflected.

We evaluate all methods in three environments (Fig. 12): (i) a small *Garden* with little heterogeneous terrain, (ii) a large *Village* with moderate heterogeneous terrain like lawn, and (iii) a large *Forest* with more heterogeneous terrain such as scattered trees. For each map, we run five independent trials and report the average exploration time and average total path length over these 5 trials.

As summarized in Table I, our HEHA consistently outperforms the baseline approaches (*Nearest* and *NBVP*) with reductions of 2.3%-35.7% of the exploration time and total path length over the best results among the baselines.

As shown in Fig. 12, our approach explores faster than the other two baselines in all maps. As the proportion of heterogeneous regions and complexity increases, the advantage of our method becomes more obvious as shown in the Village (B). Finally, as shown in the trajectory figures, the trajectories of the baseline approaches are longer and have more intersections than ours, which indicates our approach leads to less repeated exploration and thus shortens the exploration time.

2) Sensitivity of Terrain Analysis: We then run an experiment with five agents out of three different types with different sensing capabilities, using the same simulator described in Sec. VI-B1. The environment is of size $153m \times 153m$ with four floors connected by staircases (Fig. 13). The team consists of two Legged Vehicles (LV), two GVs and one AV. Two

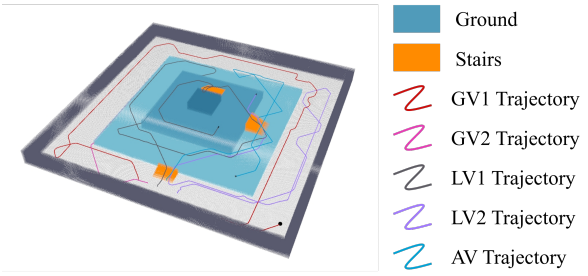


Fig. 13. Exploration in a four-floored environment.

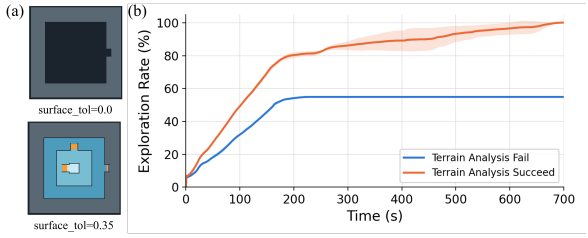


Fig. 14. (a) shows the terrain analysis result with varing `surface_tol`. (b) shows the exploration efficiency with different terrain analysis result.

LVs share the same sensing capability (360° FOV) and speed limit of $0.5m/s$. Two GVs share the same sensing range (360° FOV) while having different maximum speeds ($0.5m/s$ and $1m/s$). The AV has a different FOV (180°). Only the LVs run terrain analysis to detect staircases to possibly go across floors. GVs are limited to the first floor only. The AV can only fly on the 1st and 2nd floors and is not allowed to fly on the 3rd or 4th floor (e.g. due to the height of the ceiling).

We run two experiments. First, we vary a parameter `surface_tol` in terrain analysis for stair detection used by LV. If the height values of two neighboring cells are above this parameter, these two cells are regarded as at different floors and LV cannot traverse. Otherwise, LV regards them as traversable. All other parameters are fixed. The stairs in this environment have a rise of $0.25m$ per step.

As shown in Fig. 14(a), when `surface_tol` is set to $0.0m$, below the actual step size, the stair becomes indistinguishable from a wall, and no robot can explore the 3rd and 4th floor, resulting in incomplete exploration, shown by the blue curve in Fig. 14(b). When `surface_tol` is set to 0.35, all stairs are correctly detected by the LVs. Our global planner can then direct the LV to the 3rd and 4th floors, resulting in complete exploration as shown by the red curve in Fig. 14(b).

The second experiment we run is to consider the uncertainty in terrain analysis. We fix `surface_tol` at $0.35m$ and perturb the estimated height difference of each cell by an additive Gaussian noise with zero mean and standard deviation σ, with $\sigma \in \{0.05, 0.10, 0.15\}m$. With a larger σ, terrain analysis is more likely to regard stairs as walls, resulting in inaccessible space for the LVs. Note that, since the terrain analysis runs iteratively during exploration, whenever the height values of cells are correctly estimated in one iteration, stairs are correctly detected and LVs are directed to higher floors. On the flip side, a correctly detected stair may be regarded inaccessible if the detection goes wrong in the next iteration. As shown in Fig. 15, the exploration rate decreases, as the noise in the height estimation increases (as σ increases). For small noise ($\sigma = 0.05m$), HEHA still explores the whole environment. As σ increases to 0.10 and 0.15, the exploration rates decrease to 93% and 80% respectively.

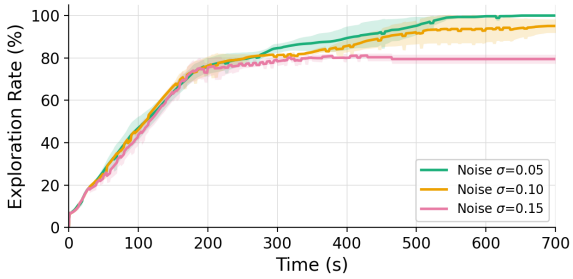


Fig. 15. Exploration efficiency with different noise in terrain analysis module.

TABLE II
RESULTS WITH VARYING α IN HETERO-FRONTIER COSTS.

Village (81m × 121m)	Time (s)	Length (m)	Reduction of Time
$\alpha = 0$	843.7	1209.8	0
$\alpha = 0.2$	821.3	1288.0	2.7%
$\alpha = 0.4$	743.3	1189.4	11.9%
$\alpha = 0.6$	**704.3**	1189.7	**16.5%**
$\alpha = 0.8$	727.0	**1185.2**	13.8%
$\alpha = 1$	781.3	1222.1	7.4%

To summarize, with the help of the proposed planners that consider heterogeneous agents, HEHA can resist small noise in terrain analysis by properly setting the parameters, but can result in incomplete exploration with incorrect or highly noisy terrain analysis.

C. Local Planning

We run an ablation study to verify the two proposed techniques in local planning.

1) Hetero-Frontier Cost: To evaluate the impact of hetero-frontier cost in our local planning, we vary the parameter α in Eq. 9 from 0 to 1 with a step size of 0.2, where $\alpha = 0$ means no use of the proposed hetero-frontier cost, and compare the exploration time taken by our HEHA in a large *village* environment (Fig. 1B(ii)), which is explained later.

As shown in Table II, with an intermediate $\alpha = 0.6$, HEHA achieves the shortest exploration time while $\alpha = 0.8$ produces the shortest path. A moderate α can direct the capable robot to those hetero-frontiers more proactively, which helps avoid

TABLE III
ABLATION STUDY OF LOCAL PLANNING. NOPR MEANS NO PRIORITY ASSIGNMENT. NOHE MEANS NO HETERO-FRONTIER COST. NOLO MEANS NEITHER OF THE TWO TRICKS IS INCLUDED. FULL MEANS USING BOTH.

Map and Metric	*NoPr*	*NoHe*	*NoLo*	*Full*
Village (81m × 121m)				
Time Avg (s)	758.0	843.7	896.7	**704.3**
Length Avg (m)	1398.6	1209.8	1405.3	**1189.7**

having this robot take a detour to go back to those hetero-frontiers at the end. In addition, too large an α (e.g. $\alpha = 1$) can lead to myopic and greedy behaviour, which sometimes results in zig-zag paths, and thus prolongs the exploration time and path length.

2) Priority Assignment: To evaluate the effectiveness of priority assignment and hetero-frontier cost, we compare HEHA (*Full*) against HEHA without priority assignment (*NoPr*), HEHA without hetero-frontier cost (*NoHe*), and HEHA without both tricks (*NoLo*). We fix $\alpha = 0.6$ in this test. As shown in Table III, HEHA has the shortest exploration time and path length, demonstrating the benefits of both proposed tricks for local planning.

D. Demonstration on Real Robots

As shown in Fig. 1 and Fig. 16, we deploy our HEHA system to explore a space of size around 35m×35m with both flat ground and stairs. We use a legged robot (Unitree Go2) and a wheeled robot (AgileX Scout Mini), both equipped with a NUC13 i7-1360P and a Livox MID360 Lidar. We used FAST-LIO2 [52] for localization and mapping on both robots. We set the LiDAR field of view to 7 meters with 360 degrees coverage, and the local exploration range was configured as a 10-meter square. The maximum speed of all robots was set to 0.7 m/s. The exploration finishes in about 205 seconds, with the process shown in the snapshots in Fig. 16. In (a–c), both robots initiate exploration from their starting poses on the ground level. During this stage, the local planner coordinates the robots' trajectories to ensure efficient coverage, as shown by the green line and yellow line.

As shown in (d), upon encountering a dead end, the legged robot is redirected (shown by the violet line) by the global planner toward unexplored areas, as illustrated in (e). Finally in (f), the legged vehicle ascends the stairs and stops at the second floor.

VII. Conclusion and Future Work

A. Conclusion

This paper considers the exploration of an unknown environment using multiple heterogeneous robots such as drones, wheeled, and legged robots, which have different capabilities to traverse complex terrains. We propose HEHA which includes a new routing algorithm RCH for its global planning that can quickly find bounded sub-optimal solutions for mHPP to minimize the maximum path length among the agents subject to traversability constraints. Additionally, the local planner in HEHA also considers heterogeneity to avoid repeated and duplicated exploration among the robots. The experimental results verify the advantage of RCH over existing approaches for mHPP, and that our HEHA can reduce the exploration time by up to 30% compared with the baseline methods

B. Future Work

For future work, one can consider developing a distributed version of RCH that allows running on various robots, and extending HEHA to its decentralized counterpart to handle communication-denied environments. A possible way to handle communication limit is to run HEHA on each robot independently, which is possible as the entire system does not require much computational resources as verified in our experiments, and then fuse the subset of robots together when these robots are within the communication range of each other. When fusing these robots together, the one among them with the strongest computational resources can be selected as the leading robot, which then runs RCH to plan paths for all robots, which is then dispatched to the other robots. This is possible due to the fast running speed of RCH and that it requires only modest computational resources (e.g. without any requirement on powerful GPU).

References

[1] R. R. Murphy, J. Kravitz, S. L. Stover, and R. Shoureshi, "Mobile robots in mine rescue and recovery," *IEEE Robotics & Automation Magazine*, vol. 16, no. 2, pp. 91–103, 2009.

[2] J. P. Grotzinger, J. Crisp, A. R. Vasavada, R. C. Anderson, C. J. Baker, R. Barry, D. F. Blake, P. Conrad, K. S. Edgett, B. Ferdowski *et al.*, "Mars science laboratory mission and science investigation," *Space science reviews*, vol. 170, pp. 5–56, 2012.

[3] H. Qin, Z. Meng, W. Meng, X. Chen, H. Sun, F. Lin, and M. H. Ang, "Autonomous exploration and mapping system using heterogeneous uavs and ugvs in gps-denied environments," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1339–1350, 2019.

[4] J. Yan, X. Lin, Z. Ren, S. Zhao, J. Yu, C. Cao, P. Yin, J. Zhang, and S. Scherer, "Mui-tare: Cooperative multi-agent exploration with unknown initial position," *IEEE Robotics and Automation Letters*, vol. 8, no. 7, pp. 4299–4306, 2023.

[5] C. Cao, H. Zhu, Z. Ren, H. Choset, and J. Zhang, "Representation granularity enables time-efficient autonomous exploration in large, complex worlds," *Science Robotics*, vol. 8, no. 80, p. eadf0970, 2023.

[6] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97.'Towards New Computational Principles for Robotics and Automation'*. IEEE, 1997, pp. 146–151.

[7] B. Zhou, H. Xu, and S. Shen, "Racer: Rapid collaborative exploration with a decentralized multi-uav system," *IEEE Transactions on Robotics*, vol. 39, no. 3, pp. 1816–1835, 2023.

[8] Y. Gao, Y. Wang, X. Zhong, T. Yang, M. Wang, Z. Xu, Y. Wang, Y. Lin, C. Xu, and F. Gao, "Meeting-merging-mission: A multi-robot coordinate framework for large-scale communication-limited exploration," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 13 700–13 707.

[9] B. Yamauchi, "Frontier-based exploration using multiple robots," in *Proceedings of the second international conference on Autonomous agents*, 1998, pp. 47–53.

[10] G. Laporte, "The vehicle routing problem: An overview of exact and approximate algorithms," *European journal of operational research*, vol. 59, no. 3, pp. 345–358, 1992.

[11] A. G. Philip, Z. Ren, S. Rathinam, and H. Choset, "A mixed-integer conic program for the multi-agent moving-target traveling salesman problem," in *2025 IEEE 21st International Conference on Automation Science and Engineering (CASE)*. IEEE, 2025, pp. 492–498.

[12] R. Doshi, S. Yadlapalli, S. Rathinam, and S. Darbha, "Approximation algorithms and heuristics for a 2-depot, heterogeneous hamiltonian path problem," *International Journal of Robust and Nonlinear Control*, vol. 21, no. 12, pp. 1434–1451, 2011.

[13] Y. Yang, Z. Liu, and W. Yu, "Approximation algorithms for the k-depots hamiltonian path problem," *Optimization Letters*, vol. 16, no. 4, pp. 1215–1234, 2022.

[14] K. Helsgaun, "General k-opt submoves for the lin–kernighan tsp heuristic," *Mathematical Programming Computation*, vol. 1, no. 2, pp. 119–163, 2009.

[15] V. Furnon and L. Perron, "Or-tools routing library," Google. [Online]. Available: https://developers.google.com/optimization/routing/

[16] N. A. Wouda, L. Lan, and W. Kool, "PyVRP: a high-performance VRP solver package," *INFORMS Journal on Computing*, vol. 36, no. 4, pp. 943–955, 2024.

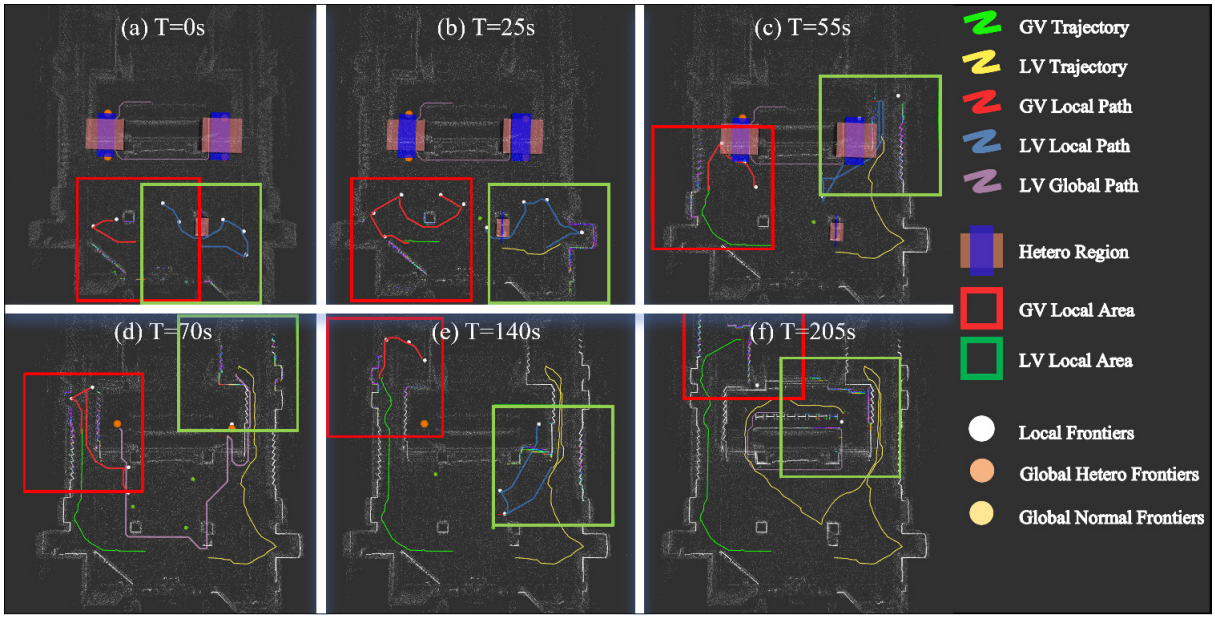


Fig. 16. Real-world exploration using HEHA. The green and yellow lines show the executed trajectories of the ground vehicle (GV) and the legged vehicle (LV) respectively. The red and blue lines represent the local path of the GV and LV respectively. The red and green boxes indicate the local planning range of the GV and LV respectively. The violet line denotes the global path of the LV. Due to the size of the space, GV only explores locally and there is no global path for the GV. The white points represent the robots' local viewpoints, the orange points represent global heterogeneous points, and the green points represent global normal viewpoints. The hetero-regions are the areas with stairs where only LV can traverse.

[17] Z. Ren, A. Nandy, S. Rathinam, and H. Choset, "Dms*: Towards minimizing makespan for multi-agent combinatorial path finding," *IEEE Robotics and Automation Letters*, vol. 9, no. 9, pp. 7987–7994, 2024.

[18] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon" next-best-view" planner for 3d exploration," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 1462–1468.

[19] T. Cieslewski, E. Kaufmann, and D. Scaramuzza, "Rapid exploration with multi-rotors: A frontier selection method for high speed flight," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 2135–2142.

[20] W. Tabib, M. Corah, N. Michael, and R. Whittaker, "Computationally efficient information-theoretic exploration of pits and caves," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 3722–3727.

[21] S. Bai, J. Wang, F. Chen, and B. Englot, "Information-theoretic exploration with bayesian optimization," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 1816–1822.

[22] C. Cao, H. Zhu, H. Choset, and J. Zhang, "Tare: A hierarchical framework for efficiently exploring complex 3d environments." in *Robotics: Science and Systems*, vol. 5, 2021, p. 2.

[23] D. Zhu, T. Li, D. Ho, C. Wang, and M. Q.-H. Meng, "Deep reinforcement learning supervised autonomous exploration in office environments," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 7548–7555.

[24] H. Li, Q. Zhang, and D. Zhao, "Deep reinforcement learning-based automatic exploration for navigation in unknown environment," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 6, pp. 2064–2076, 2019.

[25] Y. Cao, R. Zhao, Y. Wang, B. Xiang, and G. Sartoretti, "Deep reinforcement learning-based large-scale robot exploration," *IEEE Robotics and Automation Letters*, vol. 9, no. 5, pp. 4631–4638, 2024.

[26] Z. Zhang, J. Yu, J. Tang, Y. Xu, and Y. Wang, "Mr-topomap: Multi-robot exploration based on topological map in communication restricted environment," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 794–10 801, 2022.

[27] Y. Liu and Z. Ren, "A probabilistic measure of multi-robot connectivity and ergodic optimal control," *Proceedings of Robotics: Science and Systems, Los Angeles, CA, USA*, 2025.

[28] K. Masaba and A. Q. Li, "Gvgexp: Communication-constrained multi-robot exploration system based on generalized voronoi graphs," in *2021 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. IEEE, 2021, pp. 146–154.

[29] G. S. Kashyap, D. Mahajan, O. C. Phukan, A. Kumar, A. E. Brownlee, and J. Gao, "From simulations to reality: enhancing multi-robot exploration for urban search and rescue," *International Journal of Information Technology*, pp. 1–12, 2025.

[30] M. Kulkarni, M. Dharmadhikari, M. Tranzatto, S. Zimmermann, V. Reijgwart, P. De Petris, H. Nguyen, N. Khedekar, C. Papachristos, L. Ott *et al.*, "Autonomous teamed exploration of subterranean environments using legged and aerial robots," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 3306–3313.

[31] Q. Bi, X. Zhang, J. Wen, Z. Pan, S. Zhang, R. Wang, and J. Yuan, "Cure: A hierarchical framework for multi-robot autonomous exploration inspired by centroids of unknown regions," *IEEE Transactions on Automation Science and Engineering*, 2023.

[32] I. D. Miller, F. Cladera, A. Cowley, S. S. Shivakumar, E. S. Lee, L. Jarin-Lipschitz, A. Bhat, N. Rodrigues, A. Zhou, A. Cohen *et al.*, "Mine tunnel exploration using multiple quadrupedal robots," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2840–2847, 2020.

[33] J. Delmerico, E. Mueggler, J. Nitsch, and D. Scaramuzza, "Active autonomous aerial exploration for ground robot path planning," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 664–671, 2017.

[34] L. Zheng, M. Wei, R. Mei, K. Xu, J. Huang, and H. Cheng, "Aage: Air-assisted ground robotic autonomous exploration in large-scale unknown environments," *IEEE Transactions on Robotics*, 2025.

[35] T. Sakamoto, S. Bonardi, and T. Kubota, "A routing framework for heterogeneous multi-robot teams in exploration tasks," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6662–6669, 2020.

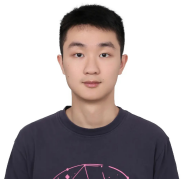
[36] N. Kottege, J. Williams, B. Tidd, F. Talbot, R. Steindl, M. Cox, D. Frousheger, T. Hines, A. Pitt, B. Tam *et al.*, "Heterogeneous robot teams with unified perception and autonomy: How team csiro data61 tied for the top score at the darpa subterranean challenge," *Field Robotics*, vol. 4, pp. 313–359, 2024.

[37] Y. Zhou, L. Quang, C. Nieto-Granda, and G. Loianno, "Coped-advancing multi-robot collaborative perception: A comprehensive dataset in real-world environments," *IEEE Robotics and Automation Letters*, vol. 9, no. 7, pp. 6416–6423, 2024.

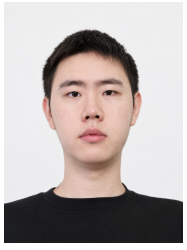
[38] J. Banfi, A. Messing, C. Kroninger, E. Stump, S. Hutchinson, and N. Roy, "Hierarchical planning for heterogeneous multi-robot routing problems via learned subteam performance," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4464–4471, 2022.

[39] S. Dong, K. Xu, Q. Zhou, A. Tagliasacchi, S. Xin, M. Nießner, and

B. Chen, "Multi-robot collaborative dense scene reconstruction," *ACM Trans. Graph.*, vol. 38, no. 4, Jul. 2019.
[40] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook, *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton, NJ, USA: Princeton University Press, 2007.
[41] P. Oberlin, S. Rathinam, and S. Darbha, "Today's traveling salesman problem," *IEEE robotics & automation magazine*, vol. 17, no. 4, pp. 70–77, 2010.
[42] Y. Ba, R. Zheng, Z. Wang, and G. Li, "Multi-objective heterogeneous fleet vehicle routing problem: Formulation and algorithm," *IEEE Transactions on Intelligent Transportation Systems*, vol. 27, no. 1, pp. 1666–1680, 2026.
[43] Z. Ren, S. Rathinam, and H. Choset, "Cbss: A new approach for multiagent combinatorial path finding," *IEEE Transactions on Robotics*, vol. 39, no. 4, pp. 2669–2683, 2023.
[44] K. Sundar and S. Rathinam, "Algorithms for heterogeneous, multiple depot, multiple unmanned vehicle path planning problems," *Journal of Intelligent & Robotic Systems*, vol. 88, pp. 513–526, 2017.
[45] M. Goldenberg, A. Felner, R. Stern, G. Sharon, N. Sturtevant, R. C. Holte, and J. Schaeffer, "Enhanced partial expansion a," *Journal of Artificial Intelligence Research*, vol. 50, pp. 141–187, 2014.
[46] J. Pearl and J. H. Kim, "Studies in semi-admissible heuristics," *IEEE transactions on pattern analysis and machine intelligence*, no. 4, pp. 392–399, 1982.
[47] C. Cao, J. Xu, J. Zhang, H. Choset, and Z. Ren, "Heuristic search for the orienteering problem with time-varying reward," in *Proceedings of the international symposium on combinatorial search*, vol. 17, 2024, pp. 11–19.
[48] S. Zhao, Y. Wu, and Z. Ren, "Bi-objective search for the traveling salesman problem with time windows and vacant penalties," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 18, no. 1, Jul. 2025, pp. 171–179.
[49] Z. Jian, Z. Lu, X. Zhou, B. Lan, A. Xiao, X. Wang, and B. Liang, "Putn: A plane-fitting based uneven terrain navigation framework," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 7160–7166.
[50] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *The international journal of robotics research*, vol. 29, no. 5, pp. 485–501, 2010.
[51] N. Sturtevant, "Benchmarks for grid-based pathfinding," *Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 2, pp. 144 – 148, 2012.
[52] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "Fast-lio2: Fast direct lidar-inertial odometry," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2053–2073, 2022.



Longrui Yang (Student Member, IEEE) received the B.S. degree from Lanzhou University of Technology, Lanzhou, China, in 2024. He is currently pursuing the master's degree at Nanyang Technological University (NTU), Singapore. His research interests include robot planning, autonomous exploration, embodied intelligence.



Yiyu Wang (Student Member, IEEE) is working toward bachelor's degrees in mechanical engineering at Shanghai Jiao Tong University, Shanghai, China, and in computer science at the University of Michigan, Ann Arbor, MI, USA. His research interests include embodied artificial intelligence, agentic robotics, and planning and search algorithms.



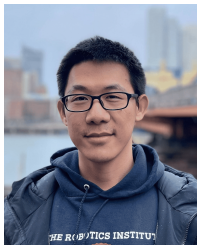
Jingfan Tang (Student Member, IEEE) is currently working toward the B.Eng. degree in electrical and computer engineering with Shanghai Jiao Tong University, Shanghai, China. His research interests include embodied artificial intelligence, world models, and planning and search algorithms.



Yunpeng Lyu received the B.S. degree in mechanical engineering from Huazhong University of Science and Technology, Wuhan, China, in 2024. He is currently working toward the Ph.D. degree in control science and engineering with Shanghai Jiao Tong University, Shanghai, China. His research interests include robotics, target search, and embodied artificial intelligence.



Shizhe Zhao (Member, IEEE) received the B.S. degree from Xidian University in 2014, and the M.S. and Ph.D. degrees from Monash University in 2018 and 2023, respectively. He is currently a postdoctoral researcher at Shanghai Jiao Tong University. His research interests include heuristic search and multi-agent motion planning.



Chao Cao (Member, IEEE) received the B.S. degree in Computer Science from The University of Hong Kong, Hong Kong, in 2017, and the M.S. degree in robotics from the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA, in 2019. He is currently working toward the Ph.D. degree in robotics with the Robotics Institute, Carnegie Mellon University. His research interests include path and motion planning, mobile manipulation, autonomous exploration, and multi-robot coordination.



Zhongqiang (Richard) Ren (Member, IEEE) received the dual B.E. degree in mechatronics from Tongji University, Shanghai, China, and FH Aachen University of Applied Sciences, Aachen, Germany, in 2015, and the M.S. degree in mechanical engineering and the Ph.D. degree in mechanical engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 2017 and 2022, respectively. He is currently an Associate Professor with the Global College, Shanghai Jiao Tong University, Shanghai, China. His research interests include robotics, path and motion planning, and multi-robot systems. He is an Associate Editor for IEEE Transactions on Robotics, Autonomous Robots, ICRA and IROS.