

RPT*: Global Planning with Probabilistic Terminals for Target Search in Complex Environments

Yunpeng Lyu¹, Chao Cao² *Member, IEEE*, Ji Zhang², Howie Choset², *Fellow, IEEE*,
and Zhongqiang Ren^{1†}, *Member, IEEE*

Abstract—Routing problems such as Hamiltonian Path Problem (HPP), seek a path to visit all the vertices in a graph while minimizing the path cost. This paper studies a variant, HPP with Probabilistic Terminals (HPP-PT), where each vertex has a probability representing the likelihood that the robot's path terminates there, and the objective is to minimize the expected path cost. HPP-PT arises in target object search, where a mobile robot must visit all candidate locations to find an object, and prior knowledge of the object's location is expressed as vertex probabilities. While routing problems have been studied for decades, few of them consider uncertainty as required in this work. The challenge lies not only in optimally ordering the vertices, as in standard HPP, but also in handling history dependency: the expected path cost depends on the order in which vertices were previously visited. This makes many existing methods inefficient or inapplicable. To address the challenge, we propose a search-based approach RPT* with solution optimality guarantees, which leverages dynamic programming in a new state space to bypass the history dependency and novel heuristics to speed up the computation. Building on RPT*, we design a Hierarchical Autonomous Target Search (HATS) system that combines RPT* with either Bayesian filtering for lifelong target search with noisy sensors, or autonomous exploration to find targets in unknown environments. Experiments in both simulation and real robot show that our approach can naturally balance between exploitation and exploration, thereby finding targets more quickly on average than baseline methods.

Index Terms—Path Planning, Traveling salesman problems, Heuristic algorithms, Mobile robots

I. INTRODUCTION

The well-known Hamiltonian Path Problem (HPP) seeks the shortest path that starts at a depot (start vertex) and visits every vertex in a graph exactly once. In this paper, we study the Hamiltonian Path Problem with Probabilistic Terminals (HPP-PT), a stochastic generalization of HPP, where each vertex is assigned a probability representing the likelihood that the robot's path will terminate there, and the objective is to minimize the expected path length. Our motivation for HPP-PT comes from target search with a mobile robot. Imagine one or more target objects, such as a key, left somewhere in a building, with the owner only vaguely recalling which rooms he had visited. In this setting, each possible location

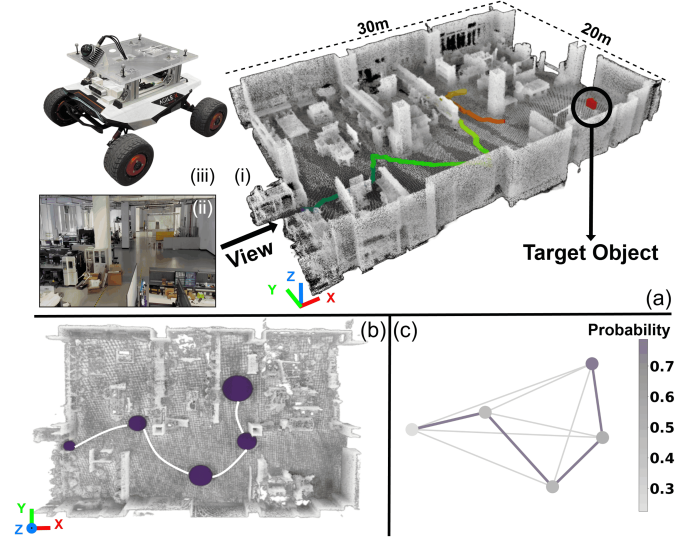


Fig. 1. Target search in unknown environments using HPP-PT. (a) shows the scenario of a real robot experiment, including (i) the point cloud map of the environment after finding the target object, where the colored line shows the trajectory of the robot; (ii) a photo of the test space; and (iii) the wheeled robot used in this experiment. (b) shows the top-down view of the environment, where purple dots represent candidate target locations, with marker sizes proportional to their corresponding probabilities of finding the target object there. (c) is the visualization of the generated solution path.

(e.g., a sofa or a table) is represented as a vertex in the HPP graph, with an associated probability indicating the likelihood of finding the target there. The robot then plans its route by optimizing the visiting order of the vertices to minimize the expected path length. If the target is found at a vertex, the robot can terminate its search early; otherwise, it continues until all vertices have been visited.

HPP-PT generalizes HPP in the sense that if the probability at each vertex is set to zero, HPP-PT reduces to the standard HPP. HPP is NP-hard [1] and so is HPP-PT. Although routing problems, such as HPP and Traveling Salesman Problem (TSP), have been studied for decades with many variants, none of them considers uncertainty similar to HPP-PT formulated in this paper. The difficulty of HPP-PT arises from two sources. First, it inherits the fundamental challenge of HPP: determining the optimal visiting order of the vertices. Second, unlike HPP, the objective in HPP-PT (i.e., the expected path length) depends on the visiting order of vertices in the path history. Different visiting orders of the same set of vertices can yield different expected costs, which greatly enlarges the planning state space. As a result, popular techniques for HPP

This work was supported by the Natural Science Foundation of China under Grant 62403313. [†]Corresponding author: Zhongqiang Ren. email: zhongqiang.ren@sjtu.edu.cn

¹Yunpeng Lyu and Zhongqiang Ren are with the Global College, Shanghai Jiao Tong University, Shanghai 200240, China (email: lv_hn_promising@sjtu.edu.cn; zhongqiang.ren@sjtu.edu.cn).

²Chao Cao, Ji Zhang, and Howie Choset are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA (email: ccao1@andrew.cmu.edu; zhangji@andrew.cmu.edu; choset@andrew.cmu.edu).

and TSP such as integer linear programming [2], [3] via branch-and-cut, or Held-Karp dynamic programming [4], [5] either cannot be readily applied to handle HPP-PT or can lead to poor runtime efficiency when being naively adapted.

To address these challenges, the first contribution of this paper is proposing algorithms for HPP-PT with solution optimality or bounded sub-optimality guarantees. Naively applying search or dynamic programming to HPP-PT can lead to a state space that includes all permutations of the vertices in the graph, which grows combinatorially with the number of vertices. To bypass such a curse of dimensionality, we develop a new approach called Routing with Probabilistic Terminals Star (RPT*) which leverages heuristic search to iteratively plan paths from the depot towards other vertices. RPT* considers a new state space to bypass the dependency on the visiting order of vertices in the path history. RPT* further introduces a fast-to-compute and informative relaxation of the problem, which is used as the heuristic function to guide the planning process. RPT* also leverages new pruning rules to speed up the computation. Compared to the well-known Held-Karp dynamic programming, RPT* saves computation due to the guidance from the heuristic function and the pruning rules during planning. Besides, we observe that, as the number of visited vertices increases, the expected path cost grows slowly, which inspires us to develop a bounded sub-optimal variant of RPT* by fusing focal search [6] to handle large graphs at the cost of finding solutions with small sub-optimality bounds.

We compared our RPT* against several baselines including integer programming, Held-Karp dynamic programming, greedy method, popular routing solver LKH [7], and adaptation from a recent learning method RL4CO [8], using both a public dataset on TSPLIB [9] and a synthetic dataset with randomly generated graphs in Euclidean spaces. Results show that, within the same runtime limit of a minute, RPT* can find optimal solutions in graphs with up to 40 vertices, and find 10% sub-optimal solutions in graphs with up to 200 vertices. For the solution quality, the expected path length found by the baselines is up to three times that of our RPT*.

To apply the proposed RPT* algorithms to a practical robotic system, the second major contribution is a Hierarchical planning system for Autonomous Target Search (HATS). HATS has three major components: (i) a module that converts the continuous workspace into discrete search regions, where each region corresponds to a vertex in the graph, and assigns the probability of finding targets at each vertex for HPP-PT based on sensor reading, (ii) the high-level planning using RPT* to determine the visiting order of target regions in the workspace, and (iii) the low-level planning for local trajectories satisfying kinodynamic constraints of the robot.

To evaluate the proposed HATS system, we implemented HATS for two tasks: lifelong target search in a known environment with noisy sensors (HATS-L, L for lifelong), and exploratory target search in an unknown environment (HATS-U, U for unknown). We compared HATS-L against baselines in Gazebo simulations and find that HATS-L tends to narrow down the likelihood about where target objects are with noisy sensors more quickly than the baselines. Besides, we compared HATS-U against baselines in both simulation and real robots

with YOLO software [10] for real-time object detection. Our HATS-U is more robust to misleading prior knowledge and can better balance between exploitation (moving toward regions with high target probability) and exploration (maintaining coverage over candidate regions), and thus find target objects with less time than the baselines on average.

The rest of the paper is organized as follows. Sec. II reviews the related work, and Sec. III formulates the HPP-PT problem. Sec. IV presents our RPT* algorithm for HPP-PT. Then, Sec. V describes the HATS system for lifelong target search and target search in unknown environments, which explains how the RPT* algorithm is embedded in the HATS system. Next, Sec. VI presents the experimental results and Sec. VII discusses the conclusion and future work. Since HPP-PT is a new problem, we detail our integer programming and Held-Karp baseline in Appendix A. Finally, Appendix B shows the system hyperparameters.

II. RELATED WORK

A. Hamiltonian Path Problem and its Variants

The Hamiltonian Path Problem (HPP) and Traveling Salesman Problem (TSP) are both well-known routing problems that are NP-hard to solve to optimality [1]. While TSP seeks a tour that visits all vertices in a graph and returns to the depot (start vertex), HPP does not require the robot to return to the depot. There are several variants of HPP based on whether the terminal vertex is fixed or not [11], [12], and this paper does not require a fixed terminal vertex.

To solve HPP and TSP, a variety of routing approaches have been developed, trading off solution quality against runtime. Among them, exact methods usually use dynamic programming such as the Held-Karp recursion for TSP and HPP [4], or formulate the problem as an integer linear program and then solve by branch-and-cut techniques [3], [13]. While being able to guarantee solution optimality, these exact approaches generally scale poorly to large graphs. Bounded sub-optimal methods typically first solve a simpler problem and then convert the result to a desired solution path or tour for TSP and HPP [14]. Besides, heuristic or learning-based methods [15], [16], including the Lin-Kernighan-Helsgaun (LKH) algorithm [7], the Attention Model [17], and learning-based combinatorial optimization frameworks such as RL4CO [8], solve HPP and TSP quickly but usually have no solution quality guarantees. This paper is interested in exact algorithms.

Although HPP, TSP and their variants have been studied a lot, few of them consider uncertainty as required in this work for target object search scenarios. Many of them consider uncertain edge costs. For example, the Canadian Traveler Problem [18], [19] considers edges with stochastic traversability, which is only revealed upon arrival. Uncertainty has also been considered in stochastic variants of the TSP with randomness in travel costs [20], [21], which focuses on uncertainty of cost rather than the vertex presence. Other literature considers vertex uncertainty, which is still different from ours. For example, Probabilistic Traveling Salesman Problem [22], [23] assigns each vertex a probability indicating the likelihood for the robot to skip that vertex during execution and seeks to

minimize the expected path length, which is different from our target search problem as we require all vertices to be visited. Stochastic Shortest Path problem [24], [25] seeks a policy to reach a pre-specified goal vertex subject to uncertainty when transiting from one vertex to another, while the HPP-PT problem seeks a path to visit all vertices with uncertainty of terminating the path at each vertex. Overall, although some routing problems with uncertainty have been studied from different perspectives, the challenge of ensuring guaranteed coverage with uncertainty of target object existence remains unaddressed by the literature.

B. Target Search

Target search is an important problem in robotics that arises in applications such as search and rescue [26], environmental monitoring [27], and hazardous operations [28]. Target search has been studied in different ways, including search for static or dynamic targets [29], [30], and search in a global or local fashion.

Seemingly related yet fundamentally different problems include target tracking [31] and ergodic search [32]. In target tracking, the target's location is assumed to be known or partially observed at all times, and the objective is to maintain persistent estimates of its state as it moves [33]. Similarly, ergodic search assumes access to a predefined spatial distribution encoding information density, and aims to generate trajectories whose visitation frequency matches this known distribution [34]. In both cases, the agent operates with explicit prior knowledge, and the challenge is to track targets rather than discovery of targets as in this work.

Target search has been investigated in a “local” planning fashion. Existing work mostly focuses on how to extract information from human [35] or the environment [36] to guide the search of the target, and employs simple planning strategy such as greedy search [37] to iteratively navigate the robot to the most informative region. Besides, recent work has incorporated semantics [38] and contextual priors [39] to guide target search. Active Visual Search [40] approaches use probabilistic models relating objects, rooms, and scene context to compute a belief distribution over the target's location and select views accordingly. However, these approaches are usually limited to a relatively small region and the focus is to navigate in semantically complex environments [36] or interaction with human during object search [35]. In contrast, our work complement these existing work by addressing the problem from a global perspective, and achieving efficient target search during the exploration of unknown environments or lifelong search.

C. Exploration of Unknown Environments

Another related topic to this work is the exploration of unknown environments, where the objective is to efficiently build a map or maximize coverage of unexplored space. These methods seek to explore the environment as quickly as possible, by selecting frontier points [41], maximizing information theoretic metrics [42], or optimizing exploration

paths via TSP [43] and receding horizon strategies [44] to ensure complete coverage of the environment.

Our HATS-U is inspired by recent hierarchical approaches [45], [46] that decompose exploration into high-level region selection and low-level motion planning, enabling efficient navigation and coverage in unknown spaces. Target search differs fundamentally from exploration as the goal is to locate a specific target as quickly as possible rather than the coverage of an unknown environment. For target search, exploring every region is neither necessary nor desirable.

III. PROBLEM STATEMENT OF HPP-PT

Let $G = (V, E)$ denote a finite directed complete graph with N vertices, where the vertex set V represents possible locations, and edge set $E \subseteq V \times V$ represents possible transitions between locations. Each edge is associated with a finite transition cost $c(e) \in \mathbb{R}^+$, which is the cost for the robot to go through the edge. The edge costs satisfy the triangle inequality. Each vertex $v \in V$ is associated with a real number $p(v) \in [0, 1]$, denoting the probability that v is used as the terminal, i.e., the robot ends its path at v . In other words, $p(v)$ is the probability that the robot finds a target object at v , and the robot stops at v thereafter.

Definition 1 (Expected Path Cost). *Let $\pi = \pi(v_1, v_N) = (v_1, v_2, \dots, v_N)$ denote a path from vertex v_1 to v_N in G . Let $C(\pi) = \sum_{k=1}^{N-1} c(v_k, v_{k+1})$ denote the accumulated edge cost (i.e., path cost) along a path π before finding a target. Then, $C(\pi)$ is a random variable, and let $\xi(\pi)$ denote the expectation of path cost $C(\pi)$, which is defined as follows:*

$$\begin{aligned} \xi(\pi) := & (1 - p_1)p_2c_{1,2} + \\ & (1 - p_1)(1 - p_2)p_3(c_{1,2} + c_{2,3}) + \dots + \\ & (1 - p_1)(1 - p_2) \dots (1 - p_{N-1})(c_{1,2} + c_{2,3} + \dots + c_{N-1,N}) \end{aligned} \quad (1)$$

where $c_{k,k+1}$ denotes the edge cost $c(e)$, $e = (v_k, v_{k+1})$ in π .

Here, the k -th ($1 \leq k \leq N - 1$) term corresponds to the expected path cost of finding the target at the $(k+1)$ -th vertex.

Remark 1. *Note that, the last term in $\xi(\pi)$ does not have a p_N in the middle of that term, since if the robot fails to find a target at the previous $N - 1$ vertices, the robot has to go to the last vertex v_N anyway, regardless of the probability of finding a target at v_N .*

Definition 2 (HPP-PT). *Let v_s denote the start vertex, and the goal of the Hamiltonian Path Problem with Probabilistic Terminals is to find a path π in G starting from v_s such that, (i) π visits all vertices in G exactly once, and (ii) the expected cost $\xi(\pi)$ reaches the minimum.*

Remark 2. *When applying HPP-PT to autonomous target search, there are different cases which may impose additional constraints to the problem formulation. If there is only one target object (such as an ID card) in G and it is known that the object must exist in G , then an additional constraint $\sum_{v \in V} p(v) = 1$ should be imposed. If there are multiple target objects (such as trash bins) in G and the robot needs to find any one of them, then there is no additional constraint on*

$p(v), v \in V$. Our approaches that will be presented next do not rely on those constraints and are applicable to all these problem variants.

IV. RPT* ALGORITHM

A*-like algorithms [47] are widely used to solve a variety of path planning problems, and our presentation assumes readers are familiar with A* [47]. However, A* cannot be directly applied to solve HPP-PT since the expected path cost in Def. 1 to be minimized depends on the visiting order of vertices. A naive extension of A* for HPP-PT would have to search in a state space that includes all permutations of vertices in G , which grows combinatorially with respect to the size of the graph, and will soon become computationally intractable. To address this difficulty, we identify a transformation of the expected path cost in Def. 1 by introducing an additional variable, which helps bypass this dependency on the visiting order of vertices and allows the planner to search a new state space efficiently.

A. Concepts and Notations

1) *New State Definition*: Along a path $\pi = (v_1, v_2, \dots, v_N)$, let q_k denote the probability of not finding a target at any vertex among the first k vertices in π :

$$q_k = (1 - p(v_1))(1 - p(v_2)) \cdots (1 - p(v_k)) \quad (2)$$

With q_k , the expected cost $\xi(\pi)$ can be reformulated.

Proposition 1. *The expected path cost in Eq. (1) is the same as the following expression:*

$$\xi(\pi) = q_1 c_{1,2} + q_2 c_{2,3} + \cdots + q_{N-1} c_{N-1,N} = \sum_{i=1}^{N-1} q_i c_{i,i+1}$$

Proof. Note that $q_{k-1} p_k + q_k = q_{k-1} (p_k + (1 - p_k)) = q_{k-1}$. Hence for any $k = 1, 2, \dots, N-1$ in Eq. (1), all terms related to $c_{k,k+1}$ can be written as

$$\begin{aligned} & (q_k p_{k+1} + q_{k+1} p_{k+2} + \cdots + q_{N-2} p_{N-1} + q_{N-1}) c_{k,k+1} \\ &= (q_k p_{k+1} + q_{k+1} p_{k+2} + \cdots + q_{N-2}) c_{k,k+1} \\ &= \cdots = q_k c_{k,k+1}. \end{aligned}$$

This can be applied to all terms $c_{k,k+1}, k = 1, 2, \dots, N-1$, thereby proving this proposition. $\square$

We now define search states based on q and show that the new state definition is Markovian, i.e., the future path depends only on the current state, independent from the past history including the set of visited vertices and their visiting orders. Let $s = (v, g, q, A)$ denote a state, which consists of a vertex $v \in V$, a g -cost as commonly used in A* $g \in \mathbb{R}^+$, a probability value $q \in [0, 1)$, and $A \subseteq V$ representing the unordered set of vertices already visited by that partial path. We use $v(s), g(s), q(s), A(s)$ to denote the respective component of state s . Later, we will use this state definition to search for paths in G and use parent pointers to record the path. Intuitively, each state s identifies a path from the start vertex v_s that has visited the vertices in A , and reaches

the vertex $v(s)$ with the cost-to-come $g(s)$ and the probability $q(s)$ of not finding a target along that path yet.

Remark 3. Here, for a state s and the corresponding path π from the start vertex to $v(s)$ as recorded by the parent pointers, the g -cost $g(s)$ is same as $\xi(\pi)$ as defined in Proposition 1. We use two different notations to highlight that, ξ is the expected cost of a solution path that visits all vertices as introduced in this new HPP-PT, while $g(s)$ is the g -cost as commonly used in A* and is the expected cost of a (partial) path from the start vertex to some other vertex in G before visiting all vertices.

2) *State Expansion*: When expanding a state s , the path identified by s is extended to all successor vertices in G that have not yet been visited. Let s' denote a new successor state that is generated when expanding s . Then, the components of s' can be computed as follows: $v(s')$ is a successor vertex of $v(s)$, $v(s') \notin A(s)$ and

$$g(s') = g(s) + q(s) \cdot c(v(s), v(s')) \quad (3)$$

$$q(s') = q(s)(1 - p(v(s'))) \quad (4)$$

$$A(s') = A(s) \cup \{v(s')\} \quad (5)$$

Eq. (3) holds because of Proposition 1. Eq. (4) follows the definition of q , and

Proposition 2. *A state s is Markovian when generating successor states by expanding s .*

Proof. When expanding s to any successor s' with $v(s') \notin A(s)$, Eq. (3)(4)(5) shows that the successor state s' depends on s only and there is no dependency on the history, i.e., the parents or ancestor states of s . $\square$

Due to Proposition 2, we can iteratively extend paths in G by expanding states based solely on their current g, q, A variables, without considering their history. This eliminates the need to consider all possible permutations (i.e., visiting orders) of vertices during planning.

3) *State Comparison and Pruning*: With the new state definition, each state s identifies a path from v_s to $v(s)$ while visiting the vertices in $A(s)$ along the path. For two paths represented by states s_1, s_2 that reach the same vertex v , (i.e., $v(s_1) = v(s_2) = v$), we need to compare them and potentially prune one of them to reduce the planning effort. To compare these two states, we introduce the following dominance rule.

Definition 3 (State Dominance). *Given two states s_1, s_2 with the same vertex (i.e., $v(s_1) = v(s_2) = v$), we say s_1 dominates s_2 , if all of the following conditions hold: (i) $g(s_1) \leq g(s_2)$ and (ii) $A(s_1) \supseteq A(s_2)$.*

Proposition 3. *In Def. 3, the condition $A(s_1) \supseteq A(s_2)$ implies that $q(s_1) \leq q(s_2)$.*

Proof. For any state s , the value $q(s)$ is the product of $(1 - p(v)), \forall v \in A(s)$ and $(1 - p(v)) \in (0, 1]$. Given that $A(s_1)$ is a superset of $A(s_2)$, then $q(s_1)$ is equal to $q(s_2)$ multiplied by possibly more terms $(1 - p(v)), v \in A(s_1), v \notin A(s_2)$, which makes $q(s_1) \leq q(s_2)$. $\square$

If s_1 dominates s_2 , then s_2 cannot lead to a better solution than s_1 and can thus be discarded. We prove the correctness

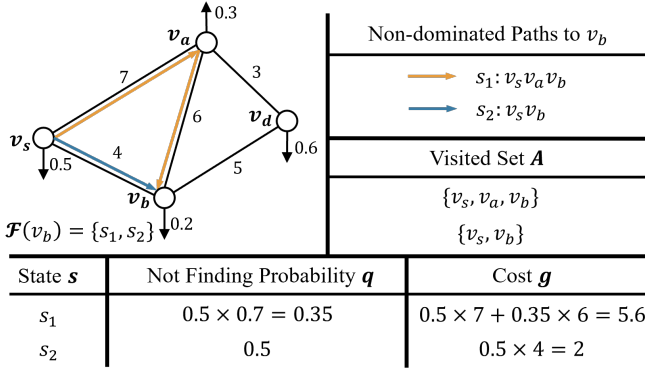


Fig. 2. Visualization of related concepts.

of this dominance rule later in Proposition 6, which involves some technical details.

If s_1 does not dominate s_2 and s_2 does not dominate s_1 , then s_1 and s_2 are non-dominated by each other, and both of them need to be kept for planning. Let $\mathcal{F}(v), \forall v \in V$ denote a *frontier set* defined at each vertex, which is a set of non-dominated states at the same vertex (Fig. 2).

Finally, let $h(s)$ denote the heuristic value of state s , which estimates the expected cost-to-go. We provide a heuristic function in Sec. IV-C. The f -value of s is defined as $f(s) := g(s) + h(s)$, where $g(s)$ is the accumulated expected cost from the start vertex to s , and $h(s)$ is the estimated remaining expected cost. $f(s)$ estimates the total expected cost of any solution that uses state s .

B. Planning Algorithm

To initialize, RPT* (Alg. 1) first creates an initial state $s_o = (v_s, 0, 1 - p(v_s), \{v_s\})$, which is added to OPEN, a priority queue where states are prioritized based on their f -values from the minimum to the maximum. RPT* maintains parent pointers of the states during planning to easily reconstruct a path when a solution is found, and initializes the parent pointer of the initial state to be *null*. The frontier sets $\mathcal{F}(v), \forall v \in V$ are initialized as empty sets.

In each search iteration (Line 6-23), a state with the smallest f -value is extracted from OPEN for processing. Then, a procedure *IsPruned* is called to check if s should be discarded based on dominance. Specifically, *IsPruned* iterates all existing states $s' \in \mathcal{F}(v(s))$ and checks if s' dominates s . If so, s cannot lead to a better solution than s' and is thus discarded. The current search iteration ends.

If s is not pruned, s is then added to $\mathcal{F}(v)$ in procedure *FilterAndAddFront*, where any existing states in $s' \in \mathcal{F}(v(s))$ is removed if it is dominated by s . Notably, *FilterAndAddFront* differs from *IsPruned* as *FilterAndAddFront* seeks to remove the existing states in $\mathcal{F}(v(s))$ while *IsPruned* seeks to remove s . Subsequently, the algorithm checks if s visits all vertices in G . If $A(s) = V$, all vertices are visited, and a solution is found. The path is then reconstructed by tracking the parent pointers of s iteratively until reaching the initial state s_o .

If s does not visit all vertices ($A(s) \neq V$), s is expanded. The set U of all unvisited successor vertices of s is obtained.

Algorithm 1 Pseudocode for RPT*

```

1:  $s_o \leftarrow (v = v_s, g = 0, q = 1 - p(v_s), A = \{v_s\})$ 
2:  $h(s_o) \leftarrow \text{GetHeuValue}(s_o), f(s_o) \leftarrow g(s_o) + h(s_o)$ 
3: Add  $s_o$  to OPEN (and FOCAL)
4:  $\text{parent}(s_o) \leftarrow \text{null}$ 
5:  $\mathcal{F}(v) \leftarrow \emptyset, \forall v \in V$ 
6: while OPEN  $\neq \emptyset$  do
7:   extract  $s$  from OPEN (or FOCAL)
8:   if IsPruned( $s$ ) then
9:     continue
10:  FilterAndAddFront( $s$ )
11:  if  $A(s) = V$  then
12:    return Reconstruct( $s$ )
13:   $U \leftarrow \text{GetSuccessors}(s)$ 
14:  for  $v' \in U$  do
15:     $g' \leftarrow g(s) + q(s) \cdot c(v(s), v')$ 
16:     $q' \leftarrow q(s)(1 - p(v'))$ 
17:     $A' \leftarrow A(s) \cup \{v'\}$ 
18:     $s' \leftarrow (v', g', q', A')$ 
19:    if IsPruned( $s'$ ) then
20:      continue
21:     $h(s') \leftarrow \text{GetHeuValue}(s'), f(s') \leftarrow g(s') + h(s')$ 
22:     $\text{parent}(s') \leftarrow s$ 
23:    Add  $s'$  to OPEN (and FOCAL)
24: return failure

IsPruned( $s$ )
1: for  $s' \in \mathcal{F}(v(s))$  do
2:   if  $s'$  dominates  $s$  then
3:     return true
4: return false

FilterAndAddFront( $s$ )
1: for  $s' \in \mathcal{F}(v(s))$  do
2:   if  $s$  dominates  $s'$  then
3:     remove  $s'$  from  $\mathcal{F}(v(s))$ 
4: add  $s$  to  $\mathcal{F}(v(s))$ 

```

For each of these successor vertices $v' \in U$, a corresponding new state s' is generated as described in Eq. (3)(4)(5). Then, *IsPruned* is called to check if the new state s' should be pruned. Finally, if s' is not pruned, the f -value of s' is computed, and s' is added to OPEN for future expansion.

Remark 4. The proposed RPT* is related to recent multi-objective search problems [48]–[50] and their variants [51]–[56]. HPP-PT is not a conventional multi-objective search problem, as its objective is a scalar value indicating the expected path cost rather than an additive cost vector. As a result, the heuristic design and pruning rules, which will be presented next, are very different from those existing work.

C. Heuristic Computation

This section presents a relaxation of the HPP-PT problem, which results in a heuristic function that can be computed in polynomial-time. Our heuristic requires the pre-computation of a table γ as follows. For any $v \in V$, let $\gamma(v, k)$ denote an estimated cost-to-go from v while visiting other remaining k vertices where k is an integer ranging from 0 to $|V| - 1$. For the base case $k = 0$, let $\gamma(v, 0) = 0$, which means there is no additional vertices to be visited from v . Then, for integers $i \in [0, k - 1]$,

$$\gamma(v, i + 1) = \min_{u, u \neq v} (1 - p(v)) \cdot (c(v, u) + \gamma(u, i)), \quad (6)$$

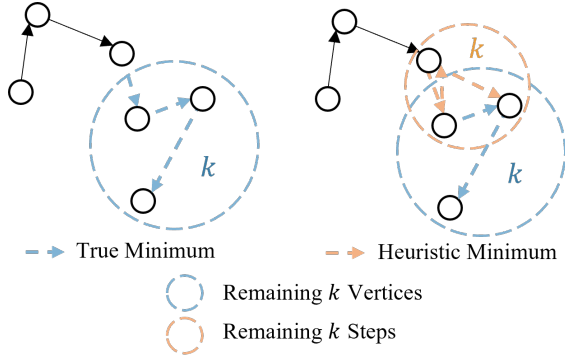


Fig. 3. Heuristic function of our RPT* algorithm: for the remaining k unvisited vertices, the heuristic models the process as k steps, explicitly allowing for repeated visits to vertices.

which is minimum incremental cost to visit any other vertex u that is different from v multiplied by the probability $(1-p(v))$ of not finding the target at v .

Proposition 4. *The pre-computation of table γ has the worst case runtime complexity $O(|V|^2)$.*

Proof. The table γ involves both $v \in V$ and $i \in [0, |V| - 1]$, and has $|V|^2$ items in total. Eq. (6) allows the use of dynamic programming to compute the entire table from $i = 0$ to $|V| - 1$, and for each i , the computation needs to loop over all $v \in V$, which has runtime complexity $O(|V|^2)$ in the worst case. $\square$

Then, the heuristic function $h(s)$ of any state s can be computed during the search with the help of this pre-computed table γ as follows.

$$\begin{aligned} h(s) &= (q(s)/(1-p(v(s)))) \cdot \gamma(v(s), k(s)), \\ k(s) &= |V| - |A(s)| \end{aligned} \quad (7)$$

Here, the term $q(s)/(1-p(v(s)))$ is basically the q -value of the parent state of s if s is not the initial state ($s \neq s_o$). When $s = s_o$, $q(s_o) = 1$ and the term is still well-defined. The term $k(s)$ is the number of unvisited vertices of state s .

We will prove that the heuristic $h(s)$ is admissible (Proposition 5). Intuitively, this heuristic allows a vertex with the cheapest visiting cost to be used many times in the future, i.e., it only requires that v is not the same as its immediate successor u , and ignores the constraints that u should not be the same as any other vertices that will be visited in the future.

D. Bounded Sub-optimal Variant

Finding an optimal solution for HPP-PT is computationally burdensome for large graphs. In large graphs, as the path length grows during planning, the q -values of states (Eq. (4)) keep decreasing, which makes the g -values of states plateau (Eq. (3)). As a result, an optimal solution path may have negligibly smaller expected path cost than many other sub-optimal solutions, especially in large graphs. We therefore develop a bounded sub-optimal variant of RPT* using focal search [57] technique, and call this variant Focal Routing with Probabilistic Terminals Star (F-RPT*).

Focal search is a bounded sub-optimal variant [57] of A*, which further introduces a FOCAL list in addition to the

OPEN list to store candidate states. Let f_{min} denote the minimum f -value of states in OPEN, and let $\epsilon \in [0, \infty)$ denote a sub-optimality bound. FOCAL list is a priority queue containing states in OPEN whose f -value are within range $[f_{min}, (1+\epsilon)f_{min}]$, and those states are prioritized based on a secondary heuristic function h_F . In each search iteration, instead of selecting the first state from OPEN, focal search selects the first state from FOCAL for processing.

In F-RPT*, we define $h_F(s) := |V| - |A(s)|$ as the number of remaining unvisited vertices of a state, and FOCAL sorts the states based on h_F from the minimum to the maximum. In each search iteration, a state with the minimum h_F value in FOCAL is selected for expansion. By doing so, F-RPT* prefers states that have visited more vertices, and tends to find a solution earlier. Since FOCAL is always a fraction of OPEN with f -value ranging between $[f_{min}, (1+\epsilon)f_{min}]$, the resulting solution is guaranteed to be bounded sub-optimal (Theorem 3), with the expected path cost g no greater than $(1+\epsilon)g^*$, where g^* is the true optimum of HPP-PT.

E. Analysis

This section proves the solution optimality and bounded sub-optimality of RPT* and F-RPT*. To this end, we first prove the admissibility of the heuristic function in Proposition 5, and then the correctness of dominance pruning in Proposition 6. Putting these together within the A* search framework, we show that the completeness and solution optimality of RPT*.

Proposition 5. (Admissible Heuristic). *The heuristic in Eq. (7) is admissible.*

Proof. Let $h^*(s)$ denote the true minimum cost of any path from a given state s to a goal state. then, for any state s , the condition for admissibility is $h(s) \leq h^*(s)$. For $k \geq 1$,

$$\begin{aligned} \gamma(v(s), k) &= (1-p(v)) \cdot \min_{u, u \neq v} (c(v, u) + \gamma(u, k-1)) \\ &\leq (1-p(v)) \cdot \min_{u, u \notin A(s)} (c(v, u) + \gamma(u, k-1)). \end{aligned}$$

Along path π , let q_m and q_{m-1} denote the probability of not finding a target at the first m and $(m-1)$ vertices in π respectively. Then,

$$\begin{aligned} h(s) &= q_{m-1} \cdot \gamma(v(s), k) \\ &\leq q_m \cdot \min_{u, u \notin A(s)} (c(v, u) + \gamma(u, k-1)) \\ &\leq q_m \cdot \min_{u, u \notin A(s)} \{c(v, u) + (1-p(u)) \cdot \\ &\quad \min_{u', u' \notin A(s) \cup \{u\}} (c(u, u') + \dots)\} = h^*(s) \end{aligned}$$

Since the condition $h(s) \leq h^*(s)$ holds for all states s , the heuristic function is admissible. $\square$

To prove the correctness of the dominance rule in Def. 3, we need the following assumption. This assumption is only needed to complete the proof and is discussed in Remark 5.

Assumption 1. *For a path π that visits a vertex $v \in V$ more than once, when computing the g -cost, the probability of finding a target for the first visit $p_{first}(v)$ is the same as the*

probability value $p(v)$ as defined in the problem statement, and the probability of finding a target at v after the first visit $p_{\text{after}}(v)$ is always zero.

To prove the correctness of the dominance rule, we use the following proof-level convention for paths that arise only through conceptual concatenation in the proof. Such repeated-visit paths are not generated by RPT*.

Proposition 6. *For two states s_1, s_2 with $v(s_1) = v(s_2)$, if s_1 dominates s_2 , s_2 cannot lead to a better solution than s_1 .*

Proof. As illustrated in Fig. 4, let $\pi_a(s_1), \pi_a(s_2)$ denote the path from v_s to $v(s_1)$ and $v(s_2)$ respectively, and let $\pi_b(s_1)$ denote an arbitrary path from $v(s_1)$ that visits all unvisited vertices not in $A(s_1)$. In other words, $\pi_b(s_1)$ completes the path $\pi_a(s_1)$. Similarly, let $\pi_b(s_2)$ denote an arbitrary path from $v(s_2)$ that traverses all unvisited vertices not in the set $A(s_2)$, and the concatenated path $\pi = \pi_a(s_2) + \pi_b(s_2)$ is a solution. Let $\pi' = \pi_a(s_1) + \pi_b(s_2)$ denote a path that concatenates $\pi_a(s_1)$ and $\pi_b(s_2)$ together. Since s_1 dominates s_2 , we know $g_1 \leq g_2$ and $\xi(\pi') \leq \xi(\pi)$. We need to show that for any possible π' , there exists a no-worse path $\pi'' = \pi_a(s_1) + \pi_b(s_1)$ than π' (i.e., $\xi(\pi'') \leq \xi(\pi')$), and as a result, there is no need to search π since $\xi(\pi'') \leq \xi(\pi') \leq \xi(\pi)$.

To show that, since $A(s_1) \supseteq A(s_2)$, there can be vertices that are visited more than once in π' . For each of those vertices u_k visited more than once in π' , if u_k is the last vertex in π , then u_k can be simply removed from π' with a strictly lower expected cost than $\xi(\pi')$. Consequently, the new expected cost is less than $\xi(\pi)$. Otherwise (u_k is not the last vertex in π'), let u_{k-1} and u_{k+1} denote the previous and the next vertex of u_k in π . Then, a new path π'' can be obtained by shortcutting u_k , i.e., in π'' , the vertices u_{k-1} and u_{k+1} are directly connected and u_k is skipped (Fig. 4). The resulting shortcut path π'' must have no-larger expected path cost than $\xi(\pi')$ because of Assumption 1 and that the graph edge costs satisfy the triangle inequality. Specifically, using the expression in Def. 1,

$$\xi(\pi') = q_1 c_{1,2} + q_2 c_{2,3} + \cdots + q_{k-1} c_{k-1,k} + q_k c_{k,k+1} + \cdots + q_{N-1} c_{N-1,N} \quad (8)$$

$$= q_1 c_{1,2} + q_2 c_{2,3} + \cdots + q_{k-1} (c_{k-1,k} + c_{k,k+1}) + \cdots + q_{N-1} c_{N-1,N} \quad (9)$$

$$\geq q_1 c_{1,2} + q_2 c_{2,3} + \cdots + q_{k-1} (c_{k,k+1}) + \cdots + q_{N-1} c_{N-1,N} = \xi(\pi'') \quad (10)$$

Here, Eq. (8) is equal to Eq. (9) since u_k is visited more than once and $p_k = p_{\text{after}} = 0$. As a result, $q_k = q_{k-1}(1 - p_k) = q_{k-1}$. The inequality from Eq. (9) to Eq. (10) follows from the triangle inequality of the edge costs $c_{k-1,k} + c_{k,k+1} \geq c_{k,k+1}$. We can keep doing so until all the vertices that are visited more than once in π are shortcut. As a result, the resulting path π'' after shortcut has $\xi(\pi'') \leq \xi(\pi')$, which means π' can be discarded. $\square$

Remark 5. *The purpose of introducing Assumption 1 is to complete the proof, since the proof involves paths that repeatedly visit the same vertices by construction. The path cost with repeatedly visited vertices is undefined in the problem*

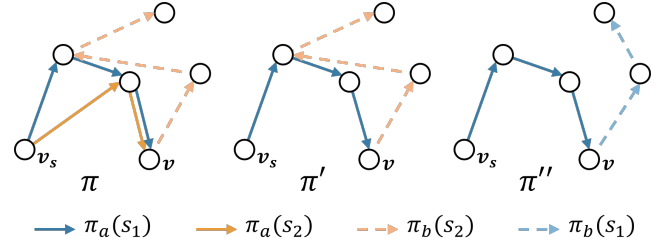


Fig. 4. Illustration of shortcutting in the proof.

statement and Assumption 1 therefore defines such costs. In other words, paths with repeatedly visited vertices are not feasible solutions for the HPP-PT problem and are never returned by RPT* in practice. For the practical HATS system for lifelong target search as presented later in Sec. V-B, the robot can repeatedly visit the same location as many times as needed. Every time when RPT* is called, the produced path has no repeated visit. However, when RPT* is called many times, it is possible that a newer path directs the robot to visit a vertex that was previously visited by an older path.

The following theorems summarize the main properties of RPT* and F-RPT*, whose correctness relies on the general best-first search framework (such as A* and its variants [57]). We show the main ideas in the following proofs.

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

Proof. RPT* conducts systematic best-first search as A* [57], where in each search iteration a state with the minimum f -value in OPEN is selected for expansion, and when expanding a state, all possible successor states are considered. During the search, a state s is pruned only when there is another state s' that dominates s in either IsPruned or FilterAndAddFront. By the definition of dominance (Def. 3) and Proposition 6, such pruning only removes sub-optimal solutions since for any future solution from s , there must exist a corresponding (better) solution from s' . As a result, for a solvable problem instance, RPT* never eliminates the only path to any vertex $v \in G$ 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 G is finite, there is a finite number of possible paths. RPT* terminates after enumerating all paths and reports failure. $\square$

Theorem 2. *RPT* finds an optimal solution for HPP-PT.*

Proof. RPT* systematically enumerates all paths in the graph G , and by Def. 3 and Proposition 6, dominance pruning never eliminates a path along an optimal solution. With an admissible heuristic (Proposition 5), the f -value of any state s popped from OPEN is a lower bound of the true optimal solution cost [57], i.e., $f(s) \leq \xi_*$ with ξ_* being the true optimal cost of an optimal solution. At termination, a state s that has visited all vertices is popped. Since all vertices are visited and there is no additional vertex to be visited, $h(s) = 0$

and $g(s) = f(s) \leq \xi_*$. As ξ_* is the true optimum, it must be the case $g(s) = \xi_*$ and RPT* finds an optimal solution. $\square$

Theorem 3. Let ξ_* denote the true optimal cost value for any solvable HPP-PT problem instance, F-RPT* finds a $(1 + \epsilon)$ bounded sub-optimal solution π for HPP-PT, i.e., $\xi(\pi) \leq (1 + \epsilon)\xi_*$.

Proof. The use of focal search [57] allows F-RPT* to select a state s with f -value within range $[f_{min}, (1 + \epsilon)f_{min}]$ with f_{min} being the minimum f -value of any state in OPEN. At termination, the popped state s is guaranteed to have g -cost $g(s) = f(s) \leq (1 + \epsilon)f_{min} \leq (1 + \epsilon)\xi_*$, which ensures the $(1 + \epsilon)$ bounded sub-optimality of the returned solution. $\square$

V. SYSTEMS AND APPLICATIONS

The presented HPP-PT planner operates on a finite graph, whereas a practical target search problem is defined in a continuous space with many viewpoints to be checked. To apply our RPT* and F-RPT* in practical target search applications, we propose the Hierarchical planning system for Autonomous Target Search (HATS), which is a hierarchical system that uses HPP-PT to iteratively plan global paths to direct the robot, and integrates perception, mapping and other modules for real robot deployment. We showcase the use of HATS framework in two distinct application scenarios.

Section V-B details HATS-L for *lifelong* target search with unreliable sensors in *known* environments, which combines the Bayesian estimation approach [58] to iteratively improve the belief about where the target object is during navigation. Specifically, a prior map about the environment is known and an initial target probability distribution is available.¹ The robot is equipped with a noisy sensor that is subject to false positive and false negative detection and is described by a probabilistic sensor model. As a result, HATS needs to iteratively generate new HPP-PT instances with the updated probability every time after a sensor detection, and calls RPT* to re-plan the path.

Section V-C describes HATS-U for integrated exploration and target search, where the robot needs to simultaneously explore an *unknown* environment and search for a target, terminating when the target is found. Specifically, when no prior map is available, the robot has to autonomously explore the environment to incrementally build a map using SLAM algorithms while searching for the target.

A. HATS System Overview

HATS is a hierarchical system (Fig. 5) consisting of four modules: (i) Perception, (ii) Mapping, (iii) Global Planning, and (iv) Local Planning.

1) *Perception*: The Perception module provides sensory data for target detection and for the subsequent Mapping module. We use a LiDAR and a camera as the primary sensors on the robot. The LiDAR provides real-time point cloud data for mapping, localization, and navigation, while the camera captures images within a fixed Field Of View (FOV) for target detection. The Perception module continuously executes

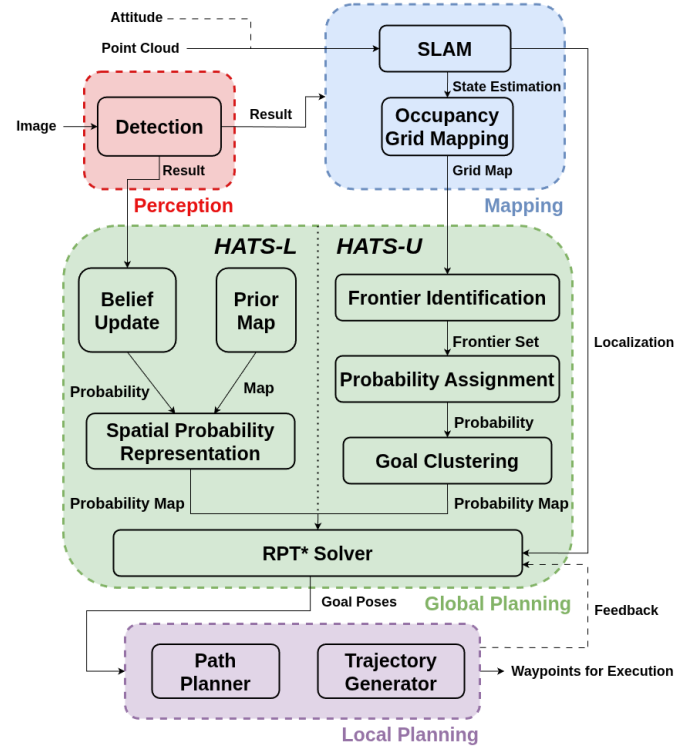


Fig. 5. The architecture and workflow of the Hierarchical planning system for Autonomous Target Search (HATS). HATS iteratively runs the workflow that begins by collecting sensory data and eventually outputs waypoints for execution. When local planning fails, HATS has a feedback loop that calls global planning to update the global path and search in different regions. The two variants HATS-L and HATS-U differ in their global planning.

onboard target detection and verification using real-time image data from the camera. In HATS-L, the perception module constantly updates the likelihood of finding a target at each location after each detection. In HATS-U, this constant detection has an interruption mechanism: when the confidence score of the detected target exceeds a predefined threshold value, the robot immediately stops its motion and claims success.

2) *Mapping*: This module processes LiDAR point cloud data to generate both state estimations and map representations for the subsequent planning modules. It runs a SLAM algorithm which produces both a point cloud map of the surrounding environment and an SE(2) pose estimate by using LiDAR (and IMU) data. We adopt LIO-mapping [59] as the underlying SLAM algorithm, and other SLAM methods can also be applied within HATS.

We employ SLAM to construct an occupancy grid map from point clouds where each cell is marked as one of the following three labels: “unknown”, “occupied” or “free”, and “occupied” and “free” both belong to the category of “known”, indicating whether the cell is occupied by an obstacle or not. With the help of SLAM, we update the occupancy grid by projecting sensor measurements into the map’s coordinate frame and updating each cell’s label through ray tracing. To handle point cloud noise, the occupancy status of each cell is determined by evaluating the density or ratio of points contained within its volume in a similar way as in [60].

This occupancy grid is the input to the subsequent Global

¹A uniform distribution is used when no clue where the target might be.

Planning module. As the robot explores, the occupancy grid grows incrementally until a target is found or the environment is fully explored without finding any target.

3) *Global Planning*: This module generates a graph based on the occupancy grid map and computes the probability of each vertex in the graph to formulate an HPP-PT problem, which is then solved to plan a global path π for the robot. Each vertex in this path π corresponds to a SE(2) pose in the environment, which will be a goal pose for the subsequent local planning. In other words, this global path determines the visiting order of goal poses in the environment. The Global Planning module in HATS-L and HATS-U differs (Fig. 5).

In HATS-L, an environment occupancy grid is built in advance. A fixed graph G embedded in the grid is provided, as well as the probability distribution of the vertices, which are used to formulate an HPP-PT problem. Every time the robot reaches a vertex, based on the sensor reading, the probability of that vertex is updated using the Bayesian rule (Sec. V-B) and a new HPP-PT is generated and solved to determine the next goal pose of the robot.

In HATS-U, the environment is unknown, and the robot has to incrementally build a point cloud map using a SLAM algorithm to generate the corresponding occupancy grid during navigation. Based on the grid, the frontier points from the boundary between known and unknown spaces are extracted. Then, as elaborated in Sec. V-C, these frontiers are used to compute the probability distribution and clustered to generate a graph G , which leads to an HPP-PT problem. Solving this problem yields the goal pose for the subsequent module.

4) *Local Planning*: While the Global Planning determines the sequence of goal poses (x, y, θ) to be visited, the Local Planning is responsible for both (i) planning the detailed geometric path to connect those goal poses and (ii) generating control command to follow the geometric path. Here, we use A*-based search in the occupancy grid to plan the geometric path connecting the goal poses from the global planning, and we use the Dynamic Window Approach [61] for local path following, which allows avoiding unexpected dynamic obstacles (such as pedestrians) in a local and reactive fashion.

Furthermore, if the Local Planning module fails to generate or execute a feasible trajectory to the assigned goal pose, it reports the failure to the Global Planning module. Consequently, the corresponding edge or vertex in the graph of HPP-PT is then removed from the graph, and the global planner is subsequently invoked again to update the global path. This feedback mechanism allows HATS to recover from local infeasibility rather than assuming that all graph edges remain traversable throughout execution.

B. Lifelong Target Search with Bayesian Update

For lifelong search with noisy sensors, as Fig. 5 shows, HATS-L employs a Bayesian filter [58] for probabilistic map update. We first introduce the general form of the Bayesian filter and demonstrate its specific application within our experiment. Let $b_t(v) \in [0, 1]$ denote the estimated probability values of vertex $v \in V$ in the graph G at time step t , which is the posterior belief of the target object being at v , i.e.,

$b_t(v) = b_t(\text{object is at } v)$. Let $\bar{b}_t(v)$ denote the corresponding prior belief at time t before incorporating the sensor measurement. At the beginning of a global planning cycle, the vertex probability used by HPP-PT is instantiated from the current belief, $p_t(v) = b_t(v)$, and remains fixed within that planning call. After the robot moves to the next target location, and receives a sensor measurement, the belief is updated and a new HPP-PT instance is generated. Note that $p(v)$ in Secs. III and IV denotes the vertex probability used within one HPP-PT instance, whereas $b_t(v)$ denotes the physical belief that evolves across closed-loop execution. Furthermore, let u_t and z_t denote the control input and measurement at time t , respectively. In our experiment, u_t represents the robot's transition between vertices, which is an edge in the graph traversed by the robot. The belief refinement process involves two steps:

1) *Prediction Step*: As formulated in Eq. (11), the prior belief $\bar{b}_t(v)$ is computed by incorporating the control input u_t using the discrete Chapman-Kolmogorov equation [58]:

$$\bar{b}_t(v) = \sum_{v' \in V} p(v | u_t, v') \cdot b_{t-1}(v'), \forall v \in G. \quad (11)$$

Here, $p(v | u_t, v')$ denotes the transition probability for the target object to move from v' to v when the robot takes control u_t . Since the target object is static in our setting, the transition probability has a simple form:

$$\begin{aligned} p(v | u_t, v') &= 1, \text{ if } v = v' \\ p(v | u_t, v') &= 0, \text{ otherwise, i.e., } v \neq v', \end{aligned} \quad (12)$$

regardless of the control u_t taken by the robot.² Consequently, the prediction step has no effect on the belief, and the prior belief at time t is the same as the posterior belief from the previous time step $t - 1$:

$$\bar{b}_t(v) = b_{t-1}(v), \forall v \in G. \quad (13)$$

2) *Update Step*: The posterior belief $b_t(v)$ corrects the prior $\bar{b}_t(v)$ with the current measurement z_t via Bayes rule, when the robot reaches vertex v after taking the control u_t :

$$b_t(v) = \eta P(z_t | v) \cdot \bar{b}_t(v). \quad (14)$$

Here, $P(z_t | v)$ denotes the observation model, and η is the normalization factor so that the posterior belief sums to one. Specifically, the sensor measurement $z_t \in \{0, 1\}$ is binary and tells whether the target object is found ($z_t = 1$) or not ($z_t = 0$). Let $\neg v$ denote that the target object is not at v , then the observation model is

$$P(z_t = 1 | v) = \alpha_1, \quad \forall v \in G \quad (15)$$

$$P(z_t = 0 | v) = 1 - \alpha_1, \quad \forall v \in G \quad (16)$$

$$P(z_t = 1 | \neg v) = \alpha_2, \quad \forall v \in G \quad (17)$$

$$P(z_t = 0 | \neg v) = 1 - \alpha_2, \quad \forall v \in G, \quad (18)$$

where $\alpha_1, \alpha_2 \in [0, 1]$ are real numbers indicating the true positive rate and false positive rate of the sensor, and the normalization factor is:

$$\eta = \frac{1}{P(z_t | v) \bar{b}_t(v) + P(z_t | \neg v) (1 - \bar{b}_t(v))}. \quad (19)$$

²A possible future work is to consider more complicated transition probability and extend our HATS to find dynamic target objects.

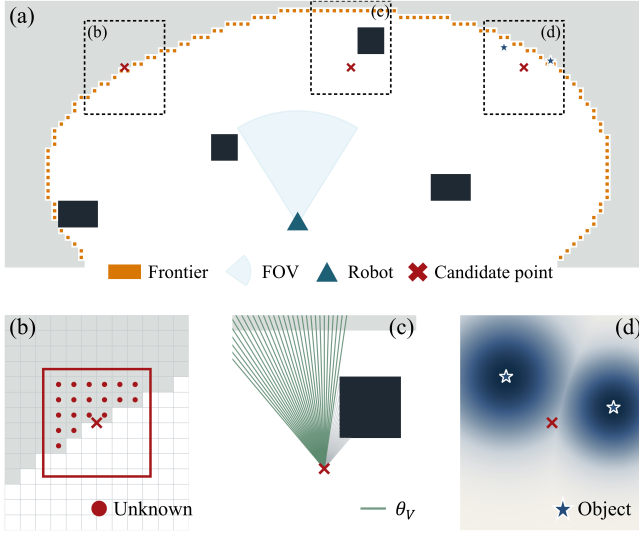


Fig. 6. Probability calculation for candidate points in the HATS-U. For each candidate point x , we compute (b) the density of unknown space $\phi_u(x)$, (c) the geometric visibility $\phi_g(x)$, and (d) the prior knowledge of object $\phi_o(x)$. These terms are linearly combined and utilized as the vertex probability within an HPP-PT instance.

This observation model indicates the likelihood for the sensor to return 0 (or 1) at any vertex $v \in G$ given that the object is at v (or not at v). After the update step, $b_t(v)$ denotes the probability of the target object being at v , i.e., $b_t(v) = b_t(\text{object is at } v)$.

Remark 6. If it is known that there is only one target, the update step (Eq. (14)) needs to update the probability of all vertices. Intuitively, detecting the target object at one vertex v will increase the probability value at v while lowering the probability value at other vertices $u \neq v, u \in G$. In our setting, we don't know how many targets are there. As a result, the update step (Eq. (14)) only needs to be applied to the vertex v that is reached by the robot after taking control u_t .

C. Target Search in Unknown Environments

As shown in Fig. 5, HATS-U dynamically constructs the graph G and updates probability distributions via the following three steps:

1) *Frontier Identification*: Based on the occupancy grid built by the Mapping module (Sec. V-A2), HATS-U extracts frontiers, i.e., the boundary cells between unknown and free cells. Note that the boundary cells between unknown and occupied cells are ignored since occupied space are obstacles and cannot be accessed by the robot.

2) *Probability Assignment*: HATS-U then prioritizes these frontier cells by computing the “information gain” for each of them using a weighted average of three factors. As depicted in Fig. 6, these three factors are: the density of unknown space ϕ_u , the geometric visibility ϕ_g , and prior knowledge of object locations ϕ_o (o for object), and the final probability is:

$$p(x) = w_u \phi_u(x) + w_g \phi_g(x) + w_o \phi_o(x), \quad (20)$$

where w_u, w_g, w_o are non-negative weighting factor so that $p(x)$ is within range $[0, 1]$. Specifically:

The density of unknown space $\phi_u(x)$ quantifies the ratio of unknown cells to the total number of cells in a neighborhood around a frontier cell x . For example, we use a square neighborhood of fixed size centered on x in our experiments (Fig. 6 (b)).

For any cell x in the occupancy grid, the geometric visibility $\phi_g(x)$ describes the proportion of unknown cells that are visible from x after considering the occlusion by static obstacles. As Fig. 6 (c) shows, let θ_{FOV} denote the sensor's Field Of View (FOV), and let θ_V denote the total angle of the visible unknown space from x (that are not blocked by obstacles), which can be computed by ray casting. Then, $\phi_g(x)$ is the ratio of θ_V over θ_{FOV} , within a range between 0 and 1. As the robot moves, we compute the θ_V based on the robot's current pose.

The prior knowledge ϕ_o is a potential field over the entire workspace, and the value $\phi_o(x)$ for any cell x in the occupancy grid describes the likelihood for the object to be at x . As shown in Fig. 6 (d), we use a mixture of Gaussians to describe the prior knowledge of object locations. Let X_o denote a set of possible object locations. In other words, for each location $x_k \in X_o$, $p_k(x) \sim \mathcal{N}(\mu_k, \sigma_k)$ denotes a Gaussian with mean value $\mu_k = x_k$ and a covariance matrix σ_k describing the uncertainty, where $p_k(x)$ is the probability value of any cell x in the workspace based on that Gaussian. Then, $\phi_o(x) = \max_{x_k \in X_o} p_k(x)$ is equal to the maximum probability value among these Gaussians.

3) *Goal Clustering*: During target search in an unknown environment, there can be many spatially adjacent frontier cells, and directly treating each frontier cell as a graph vertex would unnecessarily enlarge the resulting HPP-PT instance. We therefore use a probability-weighted Mean Shift algorithm [62] to cluster frontier cells into a compact set of candidate search regions. Compared with the standard Mean Shift, the modified algorithm will prioritize regions of high target probability. Specifically, let $\hat{c}(x)$ denote the cluster center for each frontier cell x , with its initial value set to the coordinates of x itself. Then, for a given neighborhood N of $\hat{c}(x)$, a new cluster center $\hat{c}_{new}(x)$ is calculated as the weighted mean of the neighboring coordinates $x_i \in N$ $\hat{c}_{new}(x) = \frac{\sum_{i \in N} x_i \cdot \omega_i}{\sum_{i \in N} \omega_i}$, where the weighting coefficient ω_i accounts for both the probability value and spatial proximity of the frontier cell, given by $\omega_i = p(x_i) \cdot \exp\left(-\frac{|\hat{c}(x) - x_i|^2}{2\delta^2}\right)$, and δ is the Mean Shift bandwidth.

The clustering process continues iteratively until convergence, which is determined by a stopping condition. Specifically, the update of the cluster center $\hat{c}_{new}(x)$ is considered complete when the change between successive cluster centers is below a predefined threshold δ_c , satisfying $|\hat{c}_{new}(x) - \hat{c}(x)| < \delta_c$. After convergence, a post-processing merging step is applied. It merges cluster centers that are within a defined distance threshold δ_{dist} , which reduces the final dimension of the graph G for the HPP-PT instance.

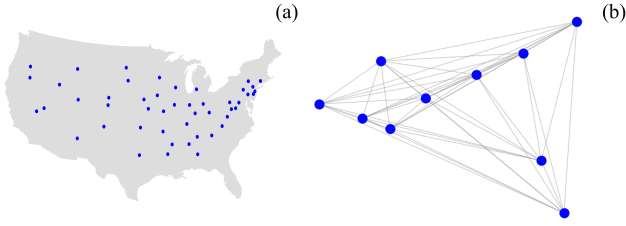


Fig. 7. Two types of datasets for experiment. (a) is the att48 instance from public TSPLIB dataset: 48 U.S. capital cities. (b) is an example of the synthetic dataset: a randomly generated graph with 10 vertices.

VI. EXPERIMENTAL RESULTS

Our experiments include evaluation of both algorithms for HPP-PT and systems for target search.

- To evaluate our algorithms (Sec. IV), we tested the proposed RPT* algorithm against two exact baselines: the integer programming (IP) [13] and Held-Karp dynamic programming (DP) [4], both adapted to HPP-PT and detailed in Appendix A. We also conducted an ablation study of RPT* to understand the impact of heuristic and focal search. Finally, we compared our approach with several baseline methods in terms of the runtime and solution quality.
- To evaluate our HATS systems (Sec. V), we compared HATS-L and HATS-U against several baselines in both Gazebo simulations and real robot experiments.

As shown in Fig. 7, we used two types of datasets including (i) a public dataset of city-like networks from TSPLIB [9], a popular library for TSP-related problems, and (ii) a synthetic dataset. The synthetic dataset consists of graphs with varying number of vertices that are randomly sampled from a 2D plane in two separate modes. For experiments whose graph sizes do not exceed 40, vertices are sampled from a 500×500 region, and for experiments whose graph-size range extends beyond 40, vertices are sampled from a larger 5000×5000 region for all tested instances. In all cases, the distance between any two vertices must exceed 5. Unless otherwise stated, for each fixed number of vertices N , 50 instances are generated. All methods were implemented in C++ and run on a laptop with an Intel Core i7-10875H 2.30 GHz CPU and 16 GB of RAM.

A. Comparison Against Exact Baselines

This subsection compares RPT* with two exact baselines for HPP-PT: IP and DP, as elaborated in Appendix A.

1) *IP Baseline*: The IP baseline can also take advantage of the proposed transformation method in Eq. (1). We denote IP and IP_{tf} as the naive IP formulation and the IP formulation that leverages the transformed formulation, respectively. We tested on five TSPLIB instances [9] with different graph sizes: gr17, gr21, gr24, fri26, and bays29, where the numbers of vertices are 17, 21, 24, 26, and 29, respectively. Gurobi [63] is used as the solver for both IP formulations.

As shown in Fig. 8, IP requires hundreds of seconds, while IP_{tf} requires a few seconds, which is obviously faster. Furthermore, our RPT* requires several milliseconds, about

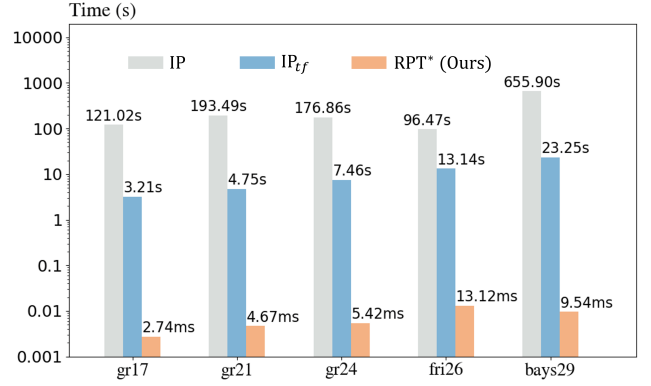


Fig. 8. Average runtime comparison of IP baselines and our RPT*.

TABLE I
IP VS. RPT* SOLUTION COST COMPARISON

Instance	IP Cost	RPT* Cost	RPT* Savings (vs. IP)
gr17	16.1545	16.1473	0.045%
gr21	13.9029	13.9024	0.004%
gr24	9.4408	9.4400	0.008%
fri26	13.2027	13.1946	0.061%
bays29	12.9606	12.9526	0.062%

three orders of magnitude faster than the transformed IP formulation.

Besides, we observe that both IP formulations often suffer from numerical errors of floating numbers and result in errors in the returned optimal solutions. As shown in Table I, the IP formulations often find slightly more expensive solutions than RPT* although the solver claims to solve the problem to optimality (when we configured the optimality gap to be zero). Recall that, in HPP-PT, due to the accumulated probability and products of floating numbers, such small errors can lead to big differences in the resulting solution paths. For this reason, the IP formulations are used only as exact-optimization baselines for small instances and are omitted from the remaining comparisons.

2) *DP Baseline*: After the expected cost is transformed into the additive form in Sec. IV, the application of Held-Karp dynamic programming to the HPP-PT becomes possible. The DP baseline takes advantage of the transformation in Eq. (1) since otherwise, the state space involves all permutations of vertices which is prohibitively large. The recurrence of this DP baseline is detailed in Appendix A-B. We ran tests on our randomly generated dataset.

As Fig. 9 shows, while RPT* can solve instances with up to approximately 40 vertices to optimality, the DP baseline solves instances with up to only approximately 24 vertices within the 60-second runtime limit. Furthermore, to achieve these solutions, the DP baseline processed more than an order of magnitude more states than RPT* for the same graph size, and as the number of vertices grows, the gap increases rapidly. These results confirm the computational advantage of the proposed RPT* over DP.

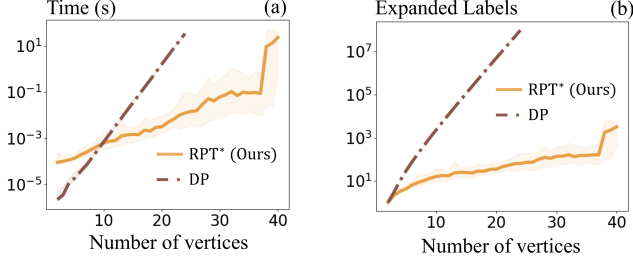


Fig. 9. Comparison of the DP baseline with RPT*. (a) Runtime comparison. DP solves instances with up to approximately 24 vertices within the runtime limit, whereas RPT* solves instances with up to approximately 40 vertices to optimality. (b) Number of processed states. The DP baseline computes an order of magnitude more states than RPT*, and the gap increases rapidly as the number of vertices grows.

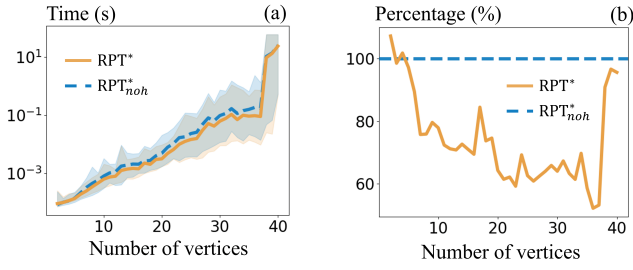


Fig. 10. Impact of heuristic. (a) Runtime comparison. (b) Normalized runtime ratio. For graphs with up to 40 vertices, both methods have almost the same success rates, while heuristic helps save 30%-50% runtime than the uninformed RPT*.

B. Ablation Study of RPT* and F-RPT*

This section presents the impact of the heuristic and focal search on RPT*, evaluated on the randomly generated dataset.

1) *Impact of Heuristic*: We first set the h -value to 0 (denoted as RPT^*_{noh}) which is used as an uninformed baseline, and then compared its performance against the full algorithm (RPT*) with the heuristic as described in Sec. IV-C.

The results show that, for very small graphs, RPT* with the heuristic can be slightly slower because of the $O(|V|^2)$ preprocessing step for the heuristic table and the additional heuristic evaluation overhead during search. As the graph size increases, the reduction in search effort becomes more evident, and the normalized runtime ratio is generally below 100%. For nontrivial graph sizes before timeout effects dominate, RPT* often requires approximately 50%–70% of the runtime of RPT^*_{noh} , corresponding to an empirical runtime reduction of roughly 30%–50%. When N exceeds approximately 37, the ratio increases because the success rate of RPT* starts to decrease and the average runtime becomes dominated by runs close to the 60 s time limit. Therefore, this increase near the largest tested graph sizes should be interpreted as a timeout effect rather than as evidence that the heuristic becomes ineffective. Overall, the results indicate that the heuristic is useful for reducing search effort on nontrivial HPP-PT instances where both methods terminate reliably. Fig. 10 shows the runtime and the normalized runtime ratio of RPT* over RPT^*_{noh} respectively, where the values below 100%

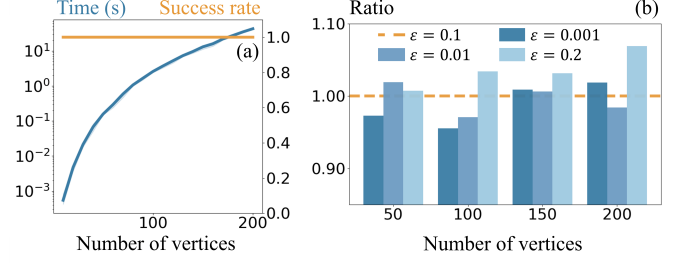


Fig. 11. Impact of focal search. (a) Runtime and success rates of F-RPT* ($\epsilon = 0.01$). (b) Comparison of different ϵ . Focal search helps RPT* to solve problems with up to 200 vertices. A small ϵ can speed up the search and further increasing ϵ does not provide much additional reduction in the runtime.

indicate that RPT* with the heuristic is faster.

The result shows that, for very small graphs, RPT* with the heuristic can be slightly slower because of the $O(|V|^2)$ preprocessing step for the heuristic table. As the size of graph increases, the reduction in search effort becomes more clear, and the normalized runtime ratio is generally below 100%. For the nontrivial instance, RPT* often requires approximately 50%–70% of the runtime of RPT^*_{noh} , corresponding to a runtime reduction of roughly 30%–50%, which verifies the advantage of the proposed heuristic. When $N > 38$, the ratio increases because the success rate of RPT* starts to decrease and the average runtime of both planners becomes close to the 60 s time limit.

2) *Runtime of F-RPT**: RPT* solves the problem to optimality, but can only handle graphs with around 40 vertices as we observed in the results. In contrast, F-RPT* can handle much larger graphs. As shown in Fig. 11 (a), we assessed F-RPT* on graph sizes from 10 to 200 vertices with a step size of 10, with 10 instances generated for each size. The runtime of F-RPT* slightly increases as the number of vertices increases. To summarize, F-RPT* can handle graphs with up to about 200 vertices within a runtime limit of 60 seconds.

3) *Varying ϵ of F-RPT**: We then tested F-RPT* with varying $\epsilon \in \{0.001, 0.01, 0.1, 0.2\}$. As shown in Fig. 11 (b), varying ϵ does not affect the runtime very much. The reason is that even if using a small epsilon (such as $\epsilon = 0.01$), there can be many states that enter the focal list, which can speed up the search.

C. Comparison Against Non-optimal Baselines

This subsection compares RPT* and F-RPT* with non-optimal baselines. We first introduce the baselines, and then present the comparison results.

1) *Baselines*: The **first** baseline Greedy on Probability (GP), uses a greedy strategy to solve HPP-PT by letting the robot always move to a vertex v in the graph G with the largest probability value $p(v)$ among the unvisited vertices. The **second** baseline Greedy on Distance (GD), always selects an unvisited vertex v in the graph with the nearest edge cost from the current vertex. The **third** baseline ignores the probability by treating the probability of all vertices simply as one. As a result, HPP-PT becomes a regular HPP and we use a popular solver Lin-Kernighan-Helsgaun (LKH) [7] to solve the

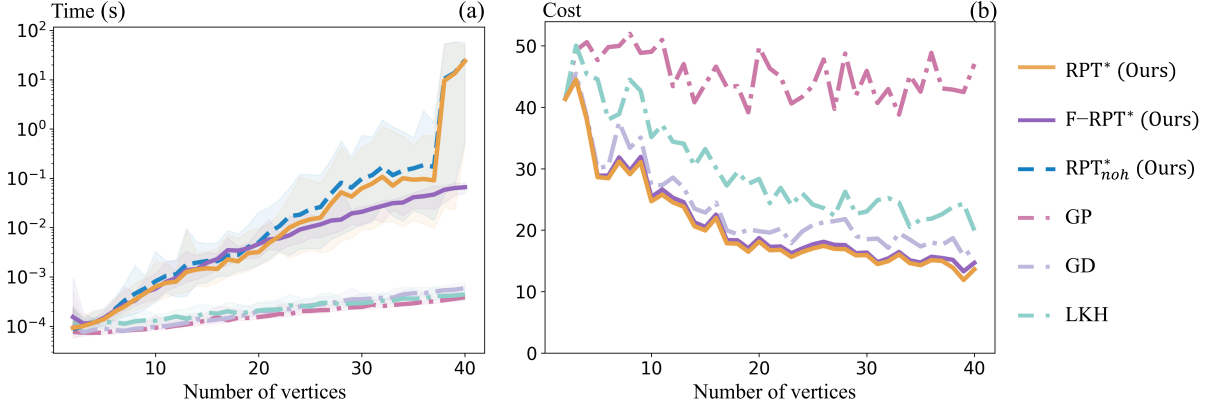


Fig. 12. Comparison against baselines. (a) Runtime comparison. (b) Solution cost comparison. In both (a) and (b), solid lines denote the proposed methods, RPT* and F-RPT*; the dashed line denotes the uninformed ablation baseline RPT*_{noh}; and dash-dotted lines denote the non-optimal baselines GP, GD, and LKH. Our RPT* and F-RPT* run slower than the baseline but find better quality solutions.

resulting HPP.³ We refer to this baseline as LKH. The **fourth** baseline utilizes the RL4CO framework [8], a recent learning-based combinatorial optimization method, where each vertex is encoded by its coordinates and probability of finding the target, and the method returns a visiting order permutation. We refer to this as RL4CO.

2) *Runtime and Solution Costs*: All methods, except for RL4CO, are tested in the synthetic dataset with up to 40 vertices. Fig. 12 (a) reports the runtime on a logarithmic scale, while Fig. 12 (b) compares the expected solution costs based on Eq. (1).

As depicted in Fig. 12, while GP runs orders of magnitude faster than our RPT*, its solution cost often doubles or triples the optimal cost returned by RPT* (Fig. 12 (b)). Similarly, GD and LKH are faster than RPT* but find solutions that are more expensive than the optimal solution found by RPT*. Overall, the results show a clear trade-off: non-optimal baselines are faster but produce higher expected costs, whereas RPT* provides optimal solutions and F-RPT* offers a practical compromise between runtime and solution quality. The next subsection evaluates how such a gain in solution quality affects the robot's performance in target search.

3) *Comparison Against RL4CO*: In this RL4CO baseline [8], the policy network uses an Attention Model [17] to output a visiting permutation from graph features, and each vertex is represented by geometric coordinates and probability of finding the target. The reward is defined as $R(\pi) = -\xi(\pi)$, where $\xi(\pi)$ is the HPP-PT expected cost in Eq. (1) so that maximizing the reward encourages minimizing the expected search cost during the training process.

The model is trained offline on synthetic instances of $N = 40$ vertices, where the coordinates and the corresponding probabilities are randomly generated. We conducted training on an NVIDIA RTX 4080 GPU with mixed precision using 100,000 training instances and 10,000 validation instances, with a batch size of 512, a learning rate of 10^{-4} , a rollout

³LKH solves a TSP problem. An HPP can be converted to a TSP by making a copy v'_s of the start vertex v_s and adding a zero-cost edge between v'_s and all other vertices. After obtaining the solution tour from LKH, one can remove v'_s to obtain a solution to HPP.

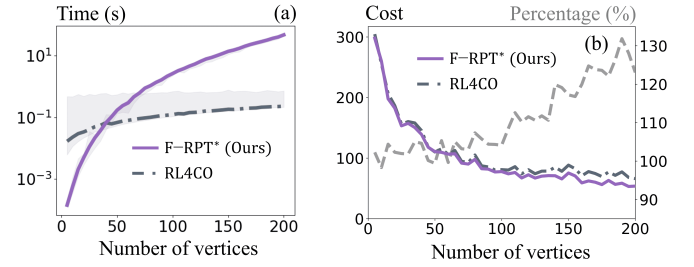


Fig. 13. Comparison between F-RPT* and the RL4CO-based learning baseline. (a) Runtime comparison. (b) Solution cost comparison and normalized ratio of RL4CO with respect to F-RPT*. RL4CO provides fast online inference and competitive solutions, while F-RPT* retains the bounded-suboptimality guarantee and achieves lower expected cost in the tested instances.

baseline, and 100 training epochs. During evaluation, a path is built by iteratively selecting the best next vertex as predicted by the network (i.e., greedy decoding).

We compared RL4CO with F-RPT* on the synthetic dataset with up to 200 vertices and the reported runtime of RL4CO includes only online inference and decoding, excluding offline training. As shown in Fig. 13, for small graphs, RL4CO obtains expected costs comparable to those of F-RPT*, but the fixed neural-network inference overhead makes it slower than F-RPT*. As the graph size increases beyond $N = 40$, the runtime of RL4CO grows more slowly than that of F-RPT*, and outperforms F-RPT*. However, the expected cost obtained by RL4CO becomes noticeably higher than that of F-RPT*, up to approximately 30% higher than the cost of F-RPT* at around $N = 200$. This result indicates that the learned policy has limited generalization capability to unseen scenarios, when evaluated on graphs substantially larger than those used for training. In contrast, F-RPT* finds better quality solutions for graphs with up to 200 vertices.

D. Lifelong Target Search with Bayesian Update

We evaluated HATS-L in an 80×80 m Gazebo environment with 20 candidate target locations. In each trial, one of the locations is selected as the true target location at random, and

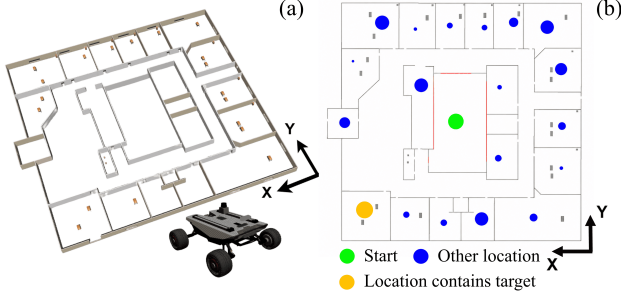


Fig. 14. Experimental setup for lifelong target search. (a) The 80×80 m simulation environment and robot model in Gazebo. (b) Pre-constructed map with 20 candidate target locations. In each trial, one location is randomly selected as the true target location and is marked in orange, while the remaining locations are target-absent and are marked in blue. The size of the blue or orange circle indicates the sampled probability values at each location. A larger probability corresponds to a larger circle.

the remaining locations are treated as target-absent. The initial belief distribution over all candidate locations is randomly sampled once per trial and kept identical for all compared planners. We pre-constructed a graph where edge weights are set as the path length returned by A* search between each pair of target locations in an occupancy grid. We used the parameters $\alpha_1 = 0.8, \alpha_2 = 0.4$ for the sensor observation model as described in Sec. V-B. We used the following termination condition for this “lifelong” test: a location is confirmed as “target-present” if its probability exceeds $p_H = 0.98$, or “target-absent” if it falls below $p_L = 0.02$. Upon reaching either threshold, the vertex is removed from the search graph, and the search process terminates when the robot finds the target. We use “mission duration” to denote the total runtime until the robot stops. To isolate the effect of the Global Planning, we replaced only the global solver with the aforementioned non-optimal baselines while keeping other modules identical across all methods.

As shown in Fig. 15, RPT* achieves the lowest median mission duration with small variance. GP can visit high-probability locations early, but it may incur unnecessary travel because it does not account for travel distance. GD and LKH reduce travel distance, but they ignore the probability of finding the target and therefore can delay the visits to locations with high probability. RL4CO achieves better mission time than GD and LKH but suffers from large variance in this randomized setting. These results show that RPT* achieves better mission time for lifelong target search with Bayesian updates due to the optimization of the expected cost in Eq. (1).

E. Simulation of Target Search in Unknown Environments

1) *Simulation Setup*: As shown in Fig. 16 (a), we first evaluated the HATS-U system within a simulated unknown forest environment of size 101×101 m, where the robot autonomously explores and searches for the target following the procedures detailed in Sec. V-C. In this experiment, there are 20 candidate target locations, and for each trial we consider only one target. The prior knowledge of its location is modeled as a Gaussian-based distribution via ϕ_o , as described in Sec. V-C3.

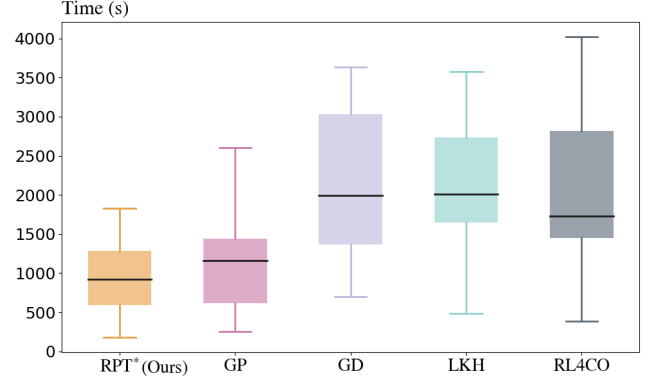


Fig. 15. Mission duration of HATS-L with different global solvers. The boxplot reports the mission duration of RPT*, GP, GD, LKH, and the RL4CO-based baseline over 20 randomized trials. Each box shows the interquartile range, the center line shows the median, and the whiskers indicate the empirical minimum and maximum. RPT* achieves the lowest median mission duration and a relatively compact distribution, demonstrating the benefit of optimizing the HPP-PT expected cost under Bayesian belief updates.

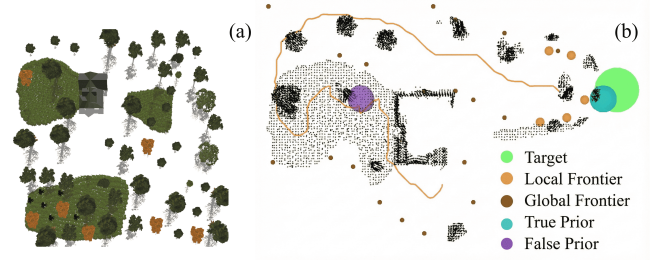


Fig. 16. Experimental setup for target search in unknown environments. (a) The 101×101 m forest environment. (b) Visualization of the search process.

The exploration process terminates when the distance between the robot and the target is less than a threshold of 5 m, and we record the time as “mission duration”. Otherwise, the mission is terminated at a time limit of 600 s and is recorded as a timeout. We replaced our RPT*, GP, GD, LKH and RL4CO for the planner in Global Planning module, and evaluated over 10 randomized trials. All other settings remain identical across all trials.

2) *Accurate Prior Knowledge*: We first tested a scenario with accurate prior knowledge, where ϕ_o assigns a weight of 0.4 to the true target, and distributes the remaining 0.6 weight uniformly over the other candidate locations.

As Table II shows, GP obtains the shortest mission duration in the accurate-prior scenario. This result is expected, because when the prior knowledge matches the true target location, directly moving toward the high-probability region can lead to shortest mission time. In this scenario, RPT* achieves the second-shortest mission duration and, together with GP, substantially outperforms the remaining baselines.

3) *Misleading Prior Knowledge*: We then designed a second scenario with misleading prior knowledge, where ϕ_o assigns a weight of 0.4 to both the true target and an additional decoy location that is far away from the true target, and the remaining 0.2 weight is uniformly distributed over the other candidates.

TABLE II
ACCURATE-PRIOR RESULTS OVER 10 RANDOMIZED TRIALS.

Planner	Success (/10)	Mission Duration (s)
RPT* (Ours)	10	77.2
GP	10	49.5
GD	10	170.6
LKH	5	439.2
RL4CO	3	420.6

TABLE III
MISLEADING-PRIOR RESULTS OVER 10 RANDOMIZED TRIALS.

Planner	Success (/10)	Mission Duration (s)
RPT* (Ours)	10	90.7
GP	9	101.4
GD	9	180.6
LKH	6	484.8
RL4CO	2	414.0

As Table III shows, in the misleading-prior scenario, GP is attracted to the decoy location, and therefore increases the mission duration and decreases the success rate. In contrast, RPT* shows only mild degradation, with a slight increase of mission duration, and is the *only* planner that succeeds in all 10 trials. Therefore, under misleading prior information, RPT* shows the highest reliability.

These results indicate that our RPT* can balance exploitation (moving toward regions with high target probability as GP does), with exploration (maintaining coverage over alternative candidate regions such as LKH does). This balance allows RPT* to locate the target more quickly when the prior knowledge can be misleading.

4) *Hyperparameter Sensitivity Study*: Our RPT* finds an optimal solution within a given graph G and has no hyperparameter. We therefore investigate the two hyper-parameters that affect the generation of G : the bandwidth δ and the post-processing merge threshold δ_{dist} , as described in Sec. V-C3. We varied one parameter at a time while holding the other at its nominal value, where nominal values mean the one we used in the previous experiments. Specifically, we varied $\delta \in \{1, 2, 3, 4, 5\}$ m while fixing $\delta_{\text{dist}} = 5$ m, and varied $\delta_{\text{dist}} \in \{4, 5, 6\}$ m while fixing $\delta = 3$ m. We tested on the aforementioned misleading-prior scenario and repeated the same 10 trials used in the previous experiments.

As shown in Table IV and V, overall, the performance in terms of mission duration does not fluctuate much when the parameters vary, with a median percentage of 2.35%. However, there are also cases where the mission duration varies a lot, such as $\delta = 4$ in Table IV. The possible reason behind is that the clustering process induces an abrupt change in the structure of the planning graph: when δ reaches certain values, the merging of frontier clusters can alter the connectivity of the HPP-PT graph, which then affects the ordering of candidate search regions and the planned path, resulting in big deviation in the mission duration.

F. Real-Robot Target Search in Unknown Environments

We further validated the HATS-U system through deployment in real-world scenarios. To demonstrate the system's

TABLE IV
SENSITIVITY TO THE BANDWIDTH PARAMETER δ WITH A FIXED $\delta_{\text{dist}} = 5$ IN THE MISLEADING-PRIOR SCENARIO. MISSION DURATION IS AVERAGED OVER SUCCESSFUL TRIALS.

δ (m)	Success (/10)	Mission Duration (s)	Deviation (%)
1	10	94.7	+4.4
2	10	95.7	+5.5
3	10	90.7	0.0
4	10	77.7	-14.3
5	10	91.8	+1.2

TABLE V
SENSITIVITY TO THE CLUSTER-MERGE DISTANCE δ_{dist} WITH A FIXED $\delta = 3$ IN THE MISLEADING-PRIOR SCENARIO. MISSION DURATION IS AVERAGED OVER SUCCESSFUL TRIALS.

δ_{dist} (m)	Success (/10)	Mission Duration (s)	Deviation (%)
4	10	93.9	+3.5
5	10	90.7	0.0
6	10	91.7	+1.1

robustness across different platforms, experiments were conducted in two distinct environments:

- **Corridor Setting**: A spacious corridor area of size 40×30 m, whose point cloud map was shown in Fig. 17. The target was a bag, and the mission was executed by a quadrupedal robot.
- **Laboratory Setting**: A 30×20 m confined, cluttered laboratory environment, whose point cloud map is shown in Fig. 1. The designated target was a trash bin, and the search task was executed by a wheeled mobile robot.

For perception, we employed the YOLO [10] object detector. The model was trained on a custom dataset that was manually collected and annotated.

In both scenarios, the robot successfully completed the search tasks with mission durations of 152 and 104 seconds, respectively. The search processes are visualized in Fig. 17 and Fig. 18. These snapshots illustrate the real-time evolution of the occupancy grid and the corresponding camera views during the target search in the Corridor and Laboratory settings, respectively. Different from exploration which often uses a 360° Field Of View (FOV) Lidar for mapping, our target search is constrained by the limited FOV of a camera. As a result, the robot has to swing left and right to find the target during its motion using the camera.

Additionally, the YOLO software we use can return a confidence score between $[0, 1]$ for each object detection. We denote the sliding window duration as $T_{\text{YOLO}} = 2.0$ s, the required fraction of confident detections as $P_{\text{YOLO}} = 0.6$, and the detection confidence threshold as $\tau_{\text{YOLO}} = 0.6$. The robot stops if, within a sliding time window of T_{YOLO} , more than P_{YOLO} of the detections have confidence scores exceeding τ_{YOLO} . By doing so, the robot can reliably detect the object in practice. Too small a threshold or too short a sliding window duration may lead to false detection and terminate the robot motion earlier. On the flip side, YOLO may never detect the object and the robot may never terminate if we set too high a threshold or too long a period. More details can be found in our video attachment.

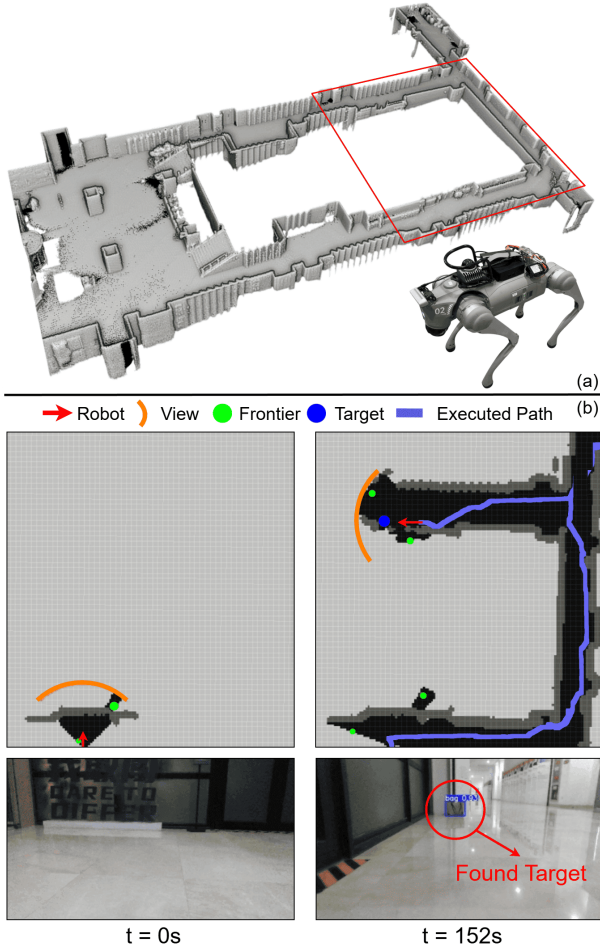


Fig. 17. Experiment of HATS-U in a corridor setting using a quadruped robot. (a) Panoramic view of the environment and the robot. (b) Snapshots of the occupancy grid and camera image during the search process.

VII. CONCLUSION AND FUTURE WORK

In this paper, we introduced the Hamiltonian Path Problem with Probabilistic Terminals (HPP-PT), a new variant of the HPP motivated by autonomous robot target search task. To address the problem, we proposed several algorithms including Integer Programming, and heuristic search Routing with Probabilistic Terminals Star (RPT*) and its bounded sub-optimal version. Extensive tests on TSPLIB and synthetic datasets demonstrated that RPT* outperforms baseline methods. Furthermore, we integrated these algorithms into a system, Hierarchical Autonomous Target Search (HATS), capable of handling both lifelong search in known environments (HATS-L) and exploratory search in unknown environments (HATS-U). The system was validated through simulations and physical deployment on both wheeled and quadrupedal robots. Our RPT* planner is able to balance between exploitation and exploration during the target object search.

For future work, we plan to extend this research from single-robot to multi-robot. Besides, we can leverage semantic mapping techniques [64] to extract probability of vertices based on semantic information. Finally, our RPT* has the potential to be extended to track moving targets by incorporating a probability

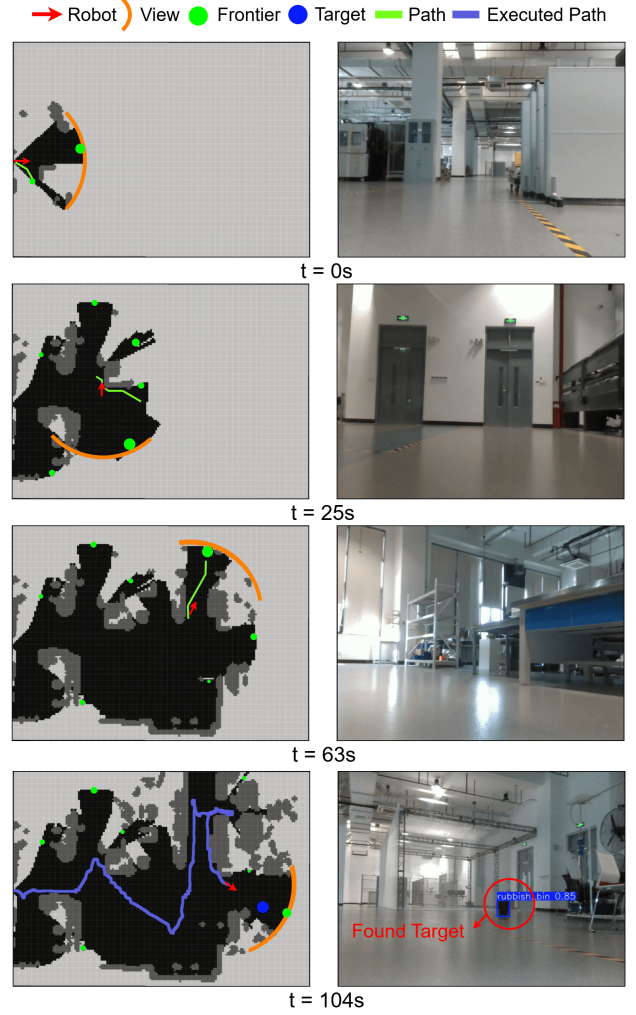


Fig. 18. Snapshots of the occupancy grid and camera image during the target search process using the wheeled robot in a lab setting.

transition model in the Bayesian filter framework when targets can move.

APPENDIX A EXACT BASELINES FOR HPP-PT

A. Integer Programming

We begin with a straightforward naive formulation and then improve it using auxiliary variables.

1) *Naive Formulation*: Let the vertices in V be organized in a fixed order. Let $x_{k,i} \in \{0, 1\}$ denote a binary variable, where $k = 1, \dots, N$ stands for the visiting order of vertex $v_i \in V$ in the solution. There are in total N^2 binary variables, and let $X \in \{0, 1\}^{N \times N}$ denote a square binary matrix consisting of $x_{k,i}$, with the k -th column being $x_k = [x_{k,1}, x_{k,2}, \dots, x_{k,N}]$. Each x_k has only one element taking value 1 with all other elements taking value 0, indicating which vertex is visited as the k -th step in the robot's path. We now define a few column vectors. Let $p = [p(v_1), p(v_2), \dots, p(v_N)]$ denote the probability vector. Let e denote a vector of length N with all elements being one. Let $C \in \mathbb{R}^{N \times N}$ denote a cost matrix with the entry in the i -th row and the j -th column being the edge cost $c(v_i, v_j)$.

To formulate the expected path cost, we introduce the following variable l_k to describe the path length when the robot visits the $(k+1)$ -th vertex in its path.

$$l_k = \sum_{i=1}^k x_i^T C x_{i+1}, k = 1, 2, \dots, N-1 \quad (21)$$

Note that $x_i^T C x_{i+1}$ selects the edge that is traversed by the robot when moving from the i -th vertex to the $(i+1)$ -th vertex in its path. Then, the objective function, i.e., the expected path cost, to be minimized is:

$$\begin{aligned} \xi(x) &= (1 - p^T x_1) p^T x_2 l_1 + (1 - p^T x_1)(1 - p^T x_2) p^T x_3 l_2 \\ &+ \dots + \prod_{j=1}^{N-1} (1 - p^T x_j) l_{N-1} \\ &= \sum_{k=1}^{N-2} \left(\prod_{j=1}^k (1 - p^T x_j) p^T x_{k+1} l_k \right) + \prod_{j=1}^{N-1} (1 - p^T x_j) l_{N-1} \end{aligned} \quad (22)$$

The constraints are:

$$e^T X = e^T, k = 1, 2, \dots, N \quad (23)$$

$$X e = e, k = 1, 2, \dots, N \quad (24)$$

$$x_1 = x_s. \quad (25)$$

Here, Constraint (23) ensures that only one vertex is visited as the k -th step $k = 1, 2, \dots, N$, and Constraint (24) ensures all selected vertices are unique and there is no repeated visit of any vertex. Finally, x_s is a binary vector of length N with only one component being 1 corresponding to the start vertex v_s , and Constraint (25) requires the first selected vertex must be the start vertex. The entire integer program is:

$$\begin{aligned} \min_x \quad & \xi(x) \\ \text{subject to} \quad & (23)(24)(25), \end{aligned}$$

which is a nonlinear program due to the product terms.

2) *Transformed Formulation*: This naive formulation has a complicated objective function $\xi(x)$ with long product terms. We propose the following transformed method to simplify these terms using an idea similar to the one in Sec. IV. We introduce a variable q_k to denote the probability of not finding a target at any vertex among the first k vertices along the path.

$$q_k = \prod_{j=1}^k (1 - p^T x_j) \quad (26)$$

With q_k , the expected path cost can be transformed into:

$$\begin{aligned} \xi(x) &= q_1 x_1^T C x_2 + q_2 x_2^T C x_3 + \dots + q_{N-1} x_{N-1}^T C x_N \\ &= \sum_{i=1}^{N-1} q_i x_i^T C x_{i+1} \end{aligned} \quad (27)$$

The constraints for this new formulation remain the same as those defined for the naive formulation (i.e., Constraints (23), (24), and (25)). The program is still nonlinear.

B. Held-Karp Dynamic Programming

The additive form of the expected cost in Eq. (1) allows Held-Karp dynamic programming to be applied for HPP-PT. Let $S \subseteq V$ be a subset of visited vertices satisfying $v_s \in S$, and let $v \in S$ be the current endpoint. Define

$$q(S) := \prod_{w \in S} (1 - p(w)), \quad (28)$$

which is the probability that the robot has not terminated at any vertex in S . Let $D(S, v)$ denote the minimum expected cost of any path that starts at v_s , visits all the vertices in S , and ends at v . The base case is

$$D(\{v_s\}, v_s) = 0. \quad (29)$$

All infeasible states are assigned value $+\infty$. For $|S| > 1$ and $v \in S \setminus \{v_s\}$, the recurrence is

$$\begin{aligned} D(S, v) &= \min_{u \in S \setminus \{v\}} [D(S \setminus \{v\}, u) + q(S \setminus \{v\}) c(u, v)], \\ q(S) &= q(S \setminus \{v\})(1 - p(v)). \end{aligned} \quad (30)$$

The factor $q(S \setminus \{v\})$ appears because the edge (u, v) is traversed only if the robot has not terminated at any previously visited vertex. The optimal expected cost is

$$\min_{v \in V \setminus \{v_s\}} D(V, v). \quad (31)$$

APPENDIX B SYSTEM HYPERPARAMETERS

Table VI summarizes the main hyperparameters used by HATS. Unless otherwise stated, the same hyperparameters are used for all methods within a given experimental setting.

REFERENCES

- [1] R. M. Karp, "Reducibility among combinatorial problems," in *50 Years of Integer Programming 1958-2008: from the Early Years to the State-of-the-Art*. Springer, 2009, pp. 219–241.
- [2] D. L. Applegate, *The traveling salesman problem: a computational study*. Princeton university press, 2006, vol. 17.
- [3] V. Cacchiani, C. Contreras-Bolton, and P. Toth, "Models and algorithms for the traveling salesman problem with time-dependent service times," *European Journal of Operational Research*, vol. 283, no. 3, pp. 825–843, 2020.
- [4] M. Held and R. M. Karp, "A dynamic programming approach to sequencing problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 10, no. 1, pp. 196–210, 1962.
- [5] K. Kimura, S. Higa, M. Okita, and F. Ino, "Accelerating the held-karp algorithm for the symmetric traveling salesman problem," *IEICE TRANSACTIONS on Information and Systems*, vol. 102, no. 12, pp. 2329–2340, 2019.
- [6] J. Pearl and J. H. Kim, "Studies in semi-admissible heuristics," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-4, no. 4, pp. 392–399, 1982.
- [7] K. Helsgaun, "An effective implementation of the lin-kernighan traveling salesman heuristic," *European journal of operational research*, vol. 126, no. 1, pp. 106–130, 2000.
- [8] F. Berto, C. Hua, J. Park, M. Kim, H. Kim, J. Son, H. Kim, J. Kim, and J. Park, "Rl4co: a unified reinforcement learning for combinatorial optimization library," in *NeurIPS 2023 Workshop: New Frontiers in Graph Learning*, 2023.
- [9] G. Reinelt, "Tsplib—a traveling salesman problem library," *ORSA journal on computing*, vol. 3, no. 4, pp. 376–384, 1991.
- [10] G. Jocher and J. Qiu, "Ultralytics yolo11," 2024.

TABLE VI
MAIN HATS HYPERPARAMETERS.

Parameter	Meaning	Usage
α_1	True-positive rate in the Bayesian observation model	HATS-L (in Sec. V-B); set to 0.8
α_2	False-positive rate in the Bayesian observation model	HATS-L (in Sec. V-B); set to 0.4
p_H	Belief threshold for declaring target-present	HATS-L (in Sec. VI-D); set to 0.98
p_L	Belief threshold for declaring target-absent	HATS-L (in Sec. VI-D); set to 0.02
w_u	Weight on unknown-space density ϕ_u	HATS-U (in Sec. V-C); set to 0.3
w_g	Weight on geometric visibility ϕ_g	HATS-U (in Sec. V-C); set to 0.3
w_o	Weight on object-prior field ϕ_o	HATS-U (in Sec. V-C); set to 0.4
δ_c	Mean Shift convergence threshold	HATS-U (in Sec. V-C); set to 0.1 m
δ	Mean Shift bandwidth	HATS-U (in Sec. V-C); swept from 1.0 m to 5.0 m
δ_{dist}	Cluster-merging distance threshold	HATS-U (in Sec. V-C); swept from 4.0 m to 6.0 m
T_{YOLO}	Detection sliding-window duration	HATS-U (in Sec. VI-F); set to 2.0 s
ρ_{YOLO}	Required fraction of confident detections	HATS-U (in Sec. VI-F); set to 0.6
τ_{YOLO}	YOLO confidence threshold	HATS-U (in Sec. VI-F); set to 0.6

- [11] J. Monnot, "Approximation algorithms for the maximum hamiltonian path problem with specified endpoint (s)," *European Journal of Operational Research*, vol. 161, no. 3, pp. 721–735, 2005.
- [12] S. Yadlapalli, J. Bae, S. Rathinam, and S. Darbha, "Approximation algorithms for a heterogeneous multiple depot hamiltonian path problem," in *Proceedings of the 2011 American Control Conference*, 2011, pp. 2789–2794.
- [13] B. Behdani and J. C. Smith, "An integer-programming-based approach to the close-enough traveling salesman problem," *INFORMS J. Comput.*, vol. 26, pp. 415–432, 2014.
- [14] N. Christofides, "Worst-case analysis of a new heuristic for the travelling salesman problem," in *Operations Research Forum*, vol. 3, no. 1. Springer, 2022, p. 20.
- [15] A. Hussain, Y. S. Muhammad, M. Nauman Sajid, I. Hussain, A. Mohamud Shoukry, and S. Gani, "Genetic algorithm for traveling salesman problem with modified cycle crossover operator," *Computational intelligence and neuroscience*, vol. 2017, no. 1, p. 7430125, 2017.
- [16] Y. Min, Y. Bai, and C. P. Gomes, "Unsupervised learning for solving the travelling salesman problem," 2024.
- [17] W. Kool, H. Van Hoof, and M. Welling, "Attention, learn to solve routing problems!" *arXiv preprint arXiv:1803.08475*, 2018.
- [18] A. Bar-Noy and B. Schieber, "The canadian traveller problem," in *SODA*, vol. 91, 1991, pp. 261–270.
- [19] V. Aksakalli, O. F. Sahin, and I. Ari, "An ao* based exact algorithm for the canadian traveler problem," *INFORMS Journal on Computing*, vol. 28, no. 1, pp. 96–111, 2016.
- [20] D. Bertsimas, "Probabilistic combinatorial optimization problems," Ph.D. dissertation, Massachusetts Institute of Technology, 1988.
- [21] A. Toriello, W. B. Haskell, and M. Poremba, "A dynamic traveling salesman problem with stochastic arc costs," *Operations Research*, vol. 62, no. 5, pp. 1107–1125, 2014.
- [22] P. Jaillet, "Probabilistic traveling salesman problems," Ph.D. dissertation, Massachusetts Institute of Technology, 1985.
- [23] A. Santini, A. Viana, X. Klimentova, and J. P. Pedroso, "The probabilistic travelling salesman problem with crowdsourcing," *Computers & Operations Research*, vol. 142, p. 105722, 2022.
- [24] D. P. Bertsekas and J. N. Tsitsiklis, "An analysis of stochastic shortest path problems," *Mathematics of Operations Research*, vol. 16, no. 3, pp. 580–595, 1991.
- [25] M. Guillot and G. Stauffer, "The stochastic shortest path problem: a polyhedral combinatorics perspective," *European Journal of Operational Research*, vol. 285, no. 1, pp. 148–158, 2020.
- [26] M. Lyu, Y. Zhao, C. Huang, and H. Huang, "Unmanned aerial vehicles for search and rescue: A survey," *Remote Sensing*, vol. 15, no. 13, p. 3266, 2023.
- [27] B. Bayat, N. Crasta, A. Crespi, A. M. Pascoal, and A. Ijspeert, "Environmental monitoring using autonomous vehicles: a survey of recent searching techniques," *Current Opinion in Biotechnology*, vol. 45, pp. 76–84, 2017.
- [28] X. Lin, F. Gao, and W. Bian, "A high-effective swarm intelligence-based multi-robot cooperation method for target searching in unknown hazardous environments," *Expert Systems with Applications*, vol. 262, p. 125609, 2025.
- [29] M. Dadgar, S. Jafari, and A. Hamzeh, "A pso-based multi-robot cooperation method for target searching in unknown environments," *Neurocomputing*, vol. 177, pp. 62–74, 2016.
- [30] T. Ishida and R. E. Korf, "Moving target search," in *IJCAI*, vol. 91, 1991, pp. 204–210.
- [31] M. Kumar and S. Mondal, "Recent developments on target tracking problems: A review," *Ocean Engineering*, vol. 236, p. 109558, 2021.
- [32] Z. Ren, A. K. Srinivasan, B. Vundurthy, I. Abraham, and H. Choset, "A pareto-optimal local optimization framework for multiobjective ergodic search," *IEEE Transactions on Robotics*, vol. 39, no. 5, pp. 3452–3463, 2023.
- [33] S. S. Blackman, "Multiple hypothesis tracking for multiple target tracking," *IEEE Aerospace and Electronic Systems Magazine*, vol. 19, no. 1, pp. 5–18, 2004.
- [34] G. Mathew and I. Mezić, "Metrics for ergodicity and design of ergodic dynamics for multi-agent systems," *Physica D: Nonlinear Phenomena*, vol. 240, no. 4–5, pp. 432–442, 2011.
- [35] R. Ramrakhya, E. Undersander, D. Batra, and A. Das, "Habitat-web: Learning embodied object-search strategies from human demonstrations at scale," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 5173–5183.
- [36] A. Aydemir, A. Pronobis, M. Göbelbecker, and P. Jensfelt, "Active visual object search in unknown environments using uncertain semantics," *IEEE Transactions on Robotics*, vol. 29, no. 4, pp. 986–1002, 2013.
- [37] Q. Jia, H. Xu, X. Feng, H. Gu, and L. Gao, "Research on cooperative area search of multiple underwater robots based on the prediction of initial target information," *Ocean Engineering*, vol. 172, pp. 660–670, 2019.
- [38] J. Crespo, J. C. Castillo, O. M. Mozos, and R. Barber, "Semantic information for robot navigation: A survey," *Applied Sciences*, vol. 10, no. 2, 2020.
- [39] A. Chikhalikar, A. A. Ravankar, J. V. S. Luces, and Y. Hirata, "Semantic-based multi-object search optimization in service robots using probabilistic and contextual priors," *IEEE Access*, 2024.
- [40] A. Pronobis and P. Jensfelt, "Large-scale semantic mapping and reasoning with heterogeneous modalities," in *2012 IEEE international conference on robotics and automation*. IEEE, 2012, pp. 3515–3522.
- [41] B. Yamauchi, "Frontier-based exploration using multiple robots," in *Proceedings of the second international conference on Autonomous agents*, 1998, pp. 47–53.
- [42] 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.
- [43] 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.
- [44] 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.
- [45] 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. ead0970, 2023.
- [46] 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.

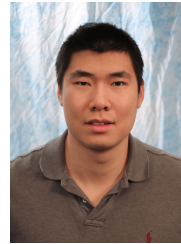
- [47] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.
- [48] C. Hernández, W. Yeoh, J. A. Baier, H. Zhang, L. Suazo, S. Koenig, and O. Salzman, "Simple and efficient bi-objective search algorithms via fast dominance checks," *Artificial Intelligence*, vol. 314, p. 103807, 2023.
- [49] Z. Ren, C. Hernández, M. Likhachev, A. Felner, S. Koenig, O. Salzman, S. Rathinam, and H. Choset, "Emoa*: A framework for search-based multi-objective path planning," *Artificial Intelligence*, vol. 339, p. 104260, 2025.
- [50] Z. Ren, R. Zhan, S. Rathinam, M. Likhachev, and H. Choset, "Enhanced multi-objective A* using balanced binary search trees," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 15, no. 1, 2022, pp. 162–170.
- [51] 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.
- [52] Z. Ren, S. Rathinam, M. Likhachev, and H. Choset, "Multi-objective safe-interval path planning with dynamic obstacles," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 8154–8161, 2022.
- [53] 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.
- [54] M. Fu, A. Kuntz, O. Salzman, and R. Alterovitz, "Asymptotically optimal inspection planning via efficient near-optimal search on sampled roadmaps," *The International Journal of Robotics Research*, vol. 42, no. 4-5, pp. 150–175, 2023.
- [55] H. Peer, E. Weiss, R. Alterovitz, and O. Salzman, "Generalizing multi-objective search via objective-aggregation functions," 2025.
- [56] Z. Ren, S. Rathinam, M. Likhachev, and H. Choset, "Multi-objective path-based d* lite," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3318–3325, 2022.
- [57] 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.
- [58] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*, ser. Intelligent robotics and autonomous agents. Cambridge, Mass: MIT Press, 2005, oCLC: ocm58451645.
- [59] H. Ye, Y. Chen, and M. Liu, "Tightly coupled 3d lidar inertial odometry and mapping," in *2019 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2019.
- [60] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous robots*, vol. 34, no. 3, pp. 189–206, 2013.
- [61] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE robotics & automation magazine*, vol. 4, no. 1, pp. 23–33, 2002.
- [62] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, 2002.
- [63] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2024.
- [64] M. F. Ginting, S.-K. Kim, D. D. Fan, M. Palieri, M. J. Kochenderfer, and A.-a. Agha-Mohammadi, "Seek: Semantic reasoning for object goal navigation in real world inspection tasks," *arXiv preprint arXiv:2405.09822*, 2024.



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.



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.



include robotic navigation, spanning localization, mapping, planning, and autonomous exploration.

Ji Zhang received the Ph.D. degree in robotics from Carnegie Mellon University, Pittsburgh, PA, USA, in 2017. He is a Systems Scientist at the Robotics Institute at Carnegie Mellon University, where he leads in the development of a series of autonomous navigation algorithms. His work was ranked 1 on the odometry leaderboard of KITTI Vision Benchmark between 2014 and 2021. He founded Kaarta, Inc., a CMU spin-off commercializing 3D mapping & modeling technologies, and stayed with the company for 4 years as chief scientist. His research interests



robotics to low-dimensional simpler ones for design, analysis, and planning.

Howie Choset (Fellow, IEEE) received his B.S. Eng. degree in computer science and his B.S. Econ. degree in entrepreneurial management from the University of Pennsylvania (Wharton), Philadelphia, PA, USA, in 1990. He received M.S. and Ph.D. degrees in mechanical engineering from California Institute of Technology (Caltech), Pasadena, CA, USA, in 1991 and 1996, respectively. He is currently a Professor of Robotics at Carnegie Mellon University, Pittsburgh, PA, USA. His research group reduces complicated high dimensional problems found in



and motion planning, and multi-robot systems. He is an Associate Editor for IEEE Transactions on Robotics, Autonomous Robots, ICRA and IROS.

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.