RPV*: Routing with Probabilistic Vertices and Its Use in Autonomous Exploration

Jingfan Tang[1], Ruizhi Deng[1,2], Yunpeng Lyu[1], Shizhe Zhao[1], and Zhongqiang Ren[1]

[1] Shanghai Jiao Tong University, Shanghai, China
[2] University of Michigan, Ann Arbor, MI 48109, USA

Abstract. This paper studies Routing with Probabilistic Vertices (RPV), a variant of the Hamiltonian Path Problem, where each vertex has a probability of existence, and the objective is to minimize the sum or the maximum of the expected arrival times at all vertices. RPV is motivated by autonomous exploration, where recent approaches use map prediction to guide the planning and speed up the exploration. Such prediction faces uncertainty about whether the predicted areas are actually reachable, which corresponds to the existence probability of vertices in RPV. While routing problems have been extensively studied, few consider the uncertainty as in RPV. To solve RPV, we propose an exact approach, RPV*, with solution optimality guarantees and its anytime variant, both of which leverage A*-like search in a new state space with novel heuristics and dominance pruning rules to speed up the computation. Integrated into a hierarchical exploration framework with a learned prediction model, RPV* reduces exploration time by up to 11.1% and path length by up to 12.4% over baselines in simulation, and is deployed on a real robot in indoor and outdoor environments.

Keywords: Path Planning, Combinatorial Optimization, Autonomous Exploration, Heuristic Search

1 Introduction

The well-known Hamiltonian Path Problem (HPP) seeks a shortest path that starts at a depot and visits every vertex in a graph exactly once. This paper studies Routing with Probabilistic Vertices (RPV), a variant of HPP, where each vertex is associated with a probability value indicating the likelihood that the vertex exists. Correspondingly, RPV seeks to minimize the sum (Min-Sum RPV) or the maximum (Min-Max RPV) of the expected arrival times of all vertices. When all probabilities equal one, Min-Max RPV and Min-Sum RPV become the regular HPP and the Minimum Latency Problem (MLP) [1], respectively. Both HPP [2] and MLP [1] are NP-hard, and so is RPV.

This work was supported by the Natural Science Foundation of China under Grant 62403313.

Our motivation for RPV comes from autonomous exploration of unknown environments [3, 4], where a robot navigates a bounded unknown environment to build a map as quickly as possible. This is important for robotic applications such as underground mining [5] and planetary exploration [6]. In exploration, areas to be explored are often formulated as graph vertices, and a routing problem is solved to determine the visit order so as to provide global guidance. To speed up the exploration, recent works [4, 7, 8] seek to leverage prediction of unexplored environments, such as predicting obstacles or floor plans, to guide the path planning. Such prediction is inherently uncertain, and the vertex probability in RPV naturally captures the likelihood that a predicted area is reachable.

RPV is challenging to solve for the following reasons. First, RPV inherits the fundamental challenge of vertex sequencing from conventional HPP [2] and MLP [1], i.e., determining the visit order of vertices in a graph. Second, introducing vertex probabilities further results in dependency on the visit history. Specifically, in conventional HPP, the optimal path cost for visiting a set of vertices is independent of the visit order, while in RPV, visiting the same set of vertices in different orders can lead to different expected path costs. This requires a planner to record not only the set of visited vertices, but also their visit order during planning, leading to a much larger state space and burdening the computation. Finally, RPV needs to be solved quickly enough to be useful in exploration tasks, since it must be re-solved whenever the robot observes new areas, which is computationally demanding.

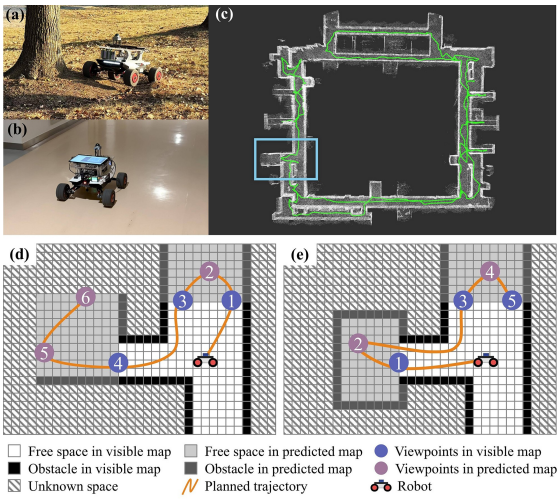


Fig. 1: Exploration using a wheeled robot in indoor (a) and outdoor (b) environments. (c) shows the result of the indoor experiment, with green lines indicating the trajectory. (d,e) zoom into the boxed region in (c) with different uncertain predictions. In (d), the left dead-end is incorrectly predicted as an open space, which can lead the robot to explore the upper tunnel first and come back later, resulting in longer exploration time. In contrast, (e) correctly predicts the dead-ends at Nodes 1 and 2, which allows the robot to explore the dead-end at first and then the upper tunnel, avoiding the back-and-forth motion and reducing the total exploration time. A planner needs to reason about these uncertain predictions (d) and (e) for efficient exploration.

To address these challenges, we develop Routing with Probabilistic Vertices Star (RPV*), an approach that finds optimal solutions for RPV, together with an anytime variant F-RPV* that returns bounded sub-optimal solutions within a given time budget. Similar to our prior work that uses search to address other routing problems [9–11], the proposed RPV* performs A*-like best-first search over partial paths from the depot, using customized dominance rules to prune unpromising partial solutions and novel fast-to-compute admissible heuristics to

guide the search. F-RPV* further incorporates anytime focal search [12] to find high-quality solutions under time budgets, which is essential for exploration.

To apply RPV* to exploration tasks, we adapt a recent hierarchical exploration framework [3] by (i) integrating a learning-based prediction method [4] to generate map predictions and vertex existence probabilities, and (ii) using RPV* as the global path planner.

To verify our approach, we compare our RPV* against several baselines, where our RPV* finds up to 9.3%–27.0% cheaper solutions than the best baseline method. For exploration, our approach helps reduce exploration time by up to 11.1% and path length by up to 12.4% over recent baseline methods. Finally, we deploy our approach on a real robot for both indoor and outdoor verification.

1.1 Related Works

Uncertainty has been incorporated into routing problems from several perspectives: the Canadian Traveler Problem [13] models edge traversability revealed only upon arrival; stochastic TSP variants address randomness in travel costs [14]; and the Probabilistic TSP [15] assigns each vertex a skip probability to minimize expected tour length. In addition, the MLP [1], also known as the Traveling Repairman Problem [16], minimizes the accumulated cost along the path, an objective similar to the one considered in this work but without accounting for vertex existence probabilities. Our prior work RPT* [10] considers the likelihood of finding target objects at each vertex, which also includes uncertainty in a routing problem but in a different way from the RPV here.

Autonomous exploration has been extensively studied under the frontier-based paradigm [17], information-driven methods [18], TSP-based planning [3], topological representations [19], and sampling-based strategies [20], yet these methods plan solely based on currently observed information, without reasoning about unobserved regions. Learning-based approaches enable longer-horizon planning [21], but typically require large amounts of training data and generalize poorly to unseen environments. Recent exploration work attempts to use map prediction to speed up exploration [22, 8, 7]: some improve prediction quality [22], while others [7, 8] use uncertainty for information-gain estimation.

2 Problem Statement

Let $G = (V, E)$ denote a complete finite undirected graph with N vertices. Each edge $e \in E$ is associated with a finite transition cost $c(e) \in \mathbb{R}^+$ satisfying the triangle inequality. Each vertex $v \in V$ is associated with a probability $p(v) \in [0, 1]$ denoting how likely v is to exist. Let v_1 denote the starting vertex (i.e. depot) of the robot, and let $p(v_1) = 1$, since the robot's current position must exist. Let $\pi = \pi(v_1, v_N) = (v_1, v_2, \ldots, v_N)$ denote a path from vertex v_1 to v_N in G. The arrival time at vertex v_i along π is:

$$t(v_1) = 0, \quad t(v_i) = \sum_{j=1}^{i-1} c(v_j, v_{j+1}), \quad i = 2, \ldots, N. \tag{1}$$

Definition 1 (Sum of Probable Arrival Times) *The sum of the probable arrival times $\xi_{\sum}(\pi)$ of π is defined as $\xi_{\sum}(\pi) := \sum_{i=1}^{N} p(v_i)t(v_i)$.*

Problem 1 (Min-Sum RPV) *The goal of the proposed min-sum RPV is to find a path π in G starting from v_1 such that, (i) π visits all vertices in G exactly once, and (ii) $\xi_{\sum}(\pi)$ is minimized.*

Remark 1 *The min-sum RPV is a generalization of MLP: when $p(v_i) = 1$ for all $v_i \in V$, $\xi_{\sum}(\pi)$ reduces to the sum of arrival times. The intuition behind Def. 1 is that the cost term of a high-probability vertex v_i decomposes as $p(v_i)t(v_i) = p(v_i)t(v_{i-1}) + p(v_i)c(v_{i-1}, v_i)$: the first term is a* history cost *that grows with every preceding vertex visited, making late visits to high-probability vertices disproportionately expensive. In exploration, minimizing this objective may encourage early visits to high-probability vertices, which correspond to predicted areas with high confidence.*

An alternative way to formulate the problem is to minimize the maximum of the probable arrival times.

Definition 2 (Maximum Probable Arrival Times) *The maximum probable arrival time $\xi_{\max}(\pi)$ of π is defined as $\xi_{\max}(\pi) := \max_i(p(v_i)t(v_i))$.*

Problem 2 (Min-Max RPV) *The goal is to find a path π in G starting from v_1 such that, (i) π visits all vertices in G exactly once, and (ii) $\xi_{\max}(\pi)$ is minimized.*

Remark 2 *The proposed min-max RPV is a generalization of the regular HPP for the following reasons. When all vertices have the same probability (i.e., when $p(v_i) = 1$ for all $v_i \in V$), $\xi_{\max}(\pi)$ equals the arrival time of the last vertex along π, which in turn equals the sum of all edge costs along π. As a result, the min-max RPV becomes a regular HPP that seeks a shortest path to visit all vertices exactly once. Intuitively, minimizing $\xi_{\max}$ encourages early visits to high-probability vertices as well.*

3 Methods

We first present RPV*, an approach for the min-sum RPV. Sec. 3.3 presents F-RPV*, which applies to both objectives. Then, we discuss how RPV* can be modified to handle the min-max RPV (Problem 2) in Sec. 3.4.

3.1 RPV* Search for Min-Sum RPV

Standard A* algorithms are not directly applicable to RPV since the objective in Def. 1 depends on the visiting order of vertices. We identify a transformation by augmenting the search state with the cumulative arrival time t, which helps bypass this dependency on the visiting order of vertices, and allows the planner to search a new state space efficiently.

State Definition and Expansion With $t(v_k)$ defined in Eq. (1), the sum of probable arrival times $\xi_{\sum}(\pi)$ admits a recursive formulation.

Lemma 1 *Let π be a path ending at v_k with a cumulative arrival time $t(v_k)$. Let $\pi' = \pi \cup \{v_{k+1}\}$ be an extension of π. The incremental update of $\xi_{\sum}$ is:*

$$\xi_{\sum}(\pi') = \xi_{\sum}(\pi) + p(v_{k+1})t(v_{k+1}) = \xi_{\sum}(\pi) + p(v_{k+1})\big(t(v_k) + c(v_k, v_{k+1})\big).$$

To capture this recursive dependency, we define a search state as a 4-tuple $s = (v, g, t, B)$, where $v \in V$ is the current vertex, $g = \xi_{\sum}(\pi)$ is the accumulated sum of probable arrival times, t is the cumulative arrival time at vertex v, and $B \subseteq V$ is the set of visited vertices.

The successors $s' = (u, g', t', B \cup \{u\})$ of a state s are generated by visiting any unvisited neighbor vertex $u \in V \setminus B$ of v: $t' = t + c(v, u)$, $g' = g + p(u) \cdot t'$.

State Comparison and Pruning To avoid duplicated or unnecessary search, we introduce a dominance rule to prune sub-optimal states that cannot lead to an optimal solution.

Definition 3 (State Dominance) *s_1 dominates s_2 ($s_1 \preceq s_2$) if they reach the same vertex and: (i) $g(s_1) \leq g(s_2)$, (ii) $t(s_1) \leq t(s_2)$, (iii) $B(s_1) \supseteq B(s_2)$.*

If s_1 dominates s_2, we can show that any future extension of s_2 cannot lead to a better path (with smaller $\xi_{\sum}$) than s_1. To show the correctness, we first introduce Lemma 2, which quantifies the cost reduction achieved by shortcutting a vertex. Subsequently, we show that a dominated state cannot extend to an optimal path by a shortcut argument.

Lemma 2 *If π' is obtained by shortcutting a vertex from π, then the accumulated sum of probable arrival times does not exceed its original value.*

Proof. Let $\pi = (v_1, v_2, \ldots, v_k, \ldots, v_m)$ be any path, and let π' be a path obtained by removing a vertex v_k from π while preserving the relative order of the remaining vertices. Denote by $t(v_i)$ and $t'(v_i)$ the arrival times of vertex v_i along π and π', respectively. Since all edge costs are non-negative and satisfy the triangle inequality, removing a vertex from the path cannot increase the travel distance between any pair of remaining consecutive vertices. Therefore, for every vertex v_i that appears after v_k in the path, we have $t'(v_i) \leq t(v_i)$, while the arrival times of vertices before v_k remain unchanged. For the expected sum of probable times, we have

$$\xi_{\sum}(\pi') \;=\; \sum_{v_i \in \pi'} p(v_i)t'(v_i) \;\leq\; \sum_{v_i \in \pi'} p(v_i)t(v_i) \;\leq\; \sum_{v_i \in \pi} p(v_i)t(v_i) \;=\; \xi_{\sum}(\pi),$$

where the first inequality follows from $t'(v_i) \leq t(v_i)$ and $p(v_i) \geq 0$, and the second inequality follows from removing a non-negative term corresponding to v_k. Thus, removing a vertex from a path cannot increase the expected sum of probable times.

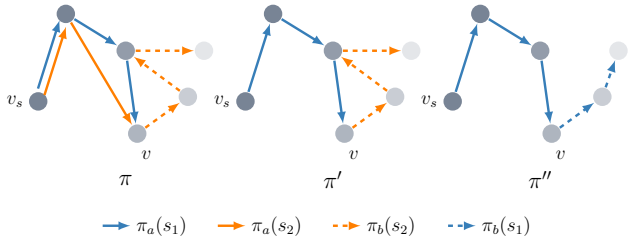


Fig. 2: Shortcut construction used in the proof of Lemma 3. Given two states $s_1 \preceq s_2$ reaching the same vertex with $B(s_1) \supseteq B(s_2)$, we show that any feasible completion of s_2 induces a feasible completion of s_1 with no greater cost. π **(left):** a full path $\pi_a(s_2)+\pi_b(s_2)$ through s_2. π' **(middle):** the same edges $\pi_b(s_2)$, viewed from s_1, which has already visited the extra vertices $B(s_1) \setminus B(s_2)$. π'' **(right):** the shortcut completion from s_1, obtained by removing the already-visited vertices from $\pi_b(s_2)$ and connecting the remaining vertices directly.

Lemma 3 *Any state pruned by the dominance rule in Def. 3 cannot lead to an optimal solution.*

Proof. As illustrated in Fig. 2, we denote the partial path leading to state s_1 and s_2 as $\pi_a(s_1)$ and $\pi_a(s_2)$, respectively, and let $\pi_b(s_2)$ be an arbitrary completion path for s_2. We construct a completion path $\pi_b(s_1)$ for s_1 by extracting from $\pi_b(s_2)$ the vertices not yet visited by s_1 (i.e., $\{v \in \pi_b(s_2) \mid v \notin B(s_1)\}$) while preserving their relative order. Consequently, $\pi_b(s_1)$ is a subsequence of $\pi_b(s_2)$. We first verify the feasibility of $\pi'' = \pi_a(s_1) + \pi_b(s_1)$. Since $\pi = \pi_a(s_2) + \pi_b(s_2)$ is feasible, $B(s_2) \cup B(\pi_b(s_2)) = V$. By construction:

$$B(\pi_a(s_1)) \cup B(\pi_b(s_1)) = B(s_1) \cup B(\pi_b(s_2)) \supseteq B(s_2) \cup B(\pi_b(s_2)) = V,$$

so π'' is feasible. We then compare costs by rewriting the objective for a complete path π_0: $\xi_{\sum}(\pi_0) = \xi_{\sum}(\pi_a(s)) + \sum_{v_i \in V \setminus B(s)} p(v_i)\, t(s) + \xi_{\sum}(\pi_b(s))$. Since $g(s_1) \leq g(s_2)$ and $t(s_1) \leq t(s_2)$, by Lemma 2 we have $g(\pi_b(s_1)) \leq g(\pi_b(s_2))$. Since $B(s_1) \supseteq B(s_2)$, every term in $\xi_{\sum}(\pi'')$ is no greater than the corresponding term in $\xi_{\sum}(\pi)$, and thus $\xi_{\sum}(\pi'') \leq \xi_{\sum}(\pi)$.

Similar to A*, the f-value of a state s is $f(s) = g(s) + h(s)$, where $g(s) = \xi_{\sum}(\pi)$ is the cost-to-come, and $h(s)$ is an admissible heuristic estimating the minimum cost-to-go for visiting the remaining vertices $V \setminus B(s)$ (Sec. 3.2).

Planning Algorithm The RPV* is detailed in Algorithm 1. RPV* maintains a priority queue, `OPEN`, which stores candidate states prioritized by their f-values $f(s)$ in ascending order. To facilitate the dominance check, at each vertex $v \in V$, we maintain a set F_v of non-dominated states that have reached that vertex.

The algorithm begins by initializing $s_0 = (v_1, 0, 0, \{v_1\})$. For each state s popped from `OPEN`, we first verify if it is dominated by any existing state in $F_{v(s)}$ according to the rules in Def. 3. If not, s is added to $F_{v(s)}$, and $F_{v(s)}$ is updated by removing any existing states that are now dominated by s.

Algorithm 1: RPV* Search

$s_0 \leftarrow (v_1, 0, 0, \{v_1\})$;
OPEN.push($s_0, f(s_0)$);
$\mathcal{F} \leftarrow \{F_v = \emptyset \mid \forall v \in V\}$;
parent $\leftarrow$ Map();
while *OPEN is not empty* **do**
  $s \leftarrow$ OPEN.pop();
  if isDominated($s, F_{v(s)}$) **then**
    continue;
  if $|B(s)| = |V|$ **then**
    return ReconstructPath(s, *parent*);
  UpdateSet($s, F_{v(s)}$);
  foreach $u \in V \setminus B(s)$ **do**
    $s_{\text{new}} \leftarrow$ Expand(s, u);
    if ***not*** isDominated(s_{new}, F_u) **then**
      parent$[s_{\text{new}}] \leftarrow s$;
      OPEN.push($s_{\text{new}}, f(s_{\text{new}})$);
return Failure;

3.2 Heuristic

To enhance computational efficiency and guide the search toward the optimal solution, we introduce an admissible heuristic function $h(s)$ that provides a lower bound on the cost-to-go. As defined in Lemma 1, the future cost depends on both the probabilities of unvisited vertices and the time required to reach them.

Let $U(s) = V \setminus B(s)$ denote the set of unvisited vertices. Any feasible completion from state s must start at the current vertex $v(s)$ and eventually visit every vertex in $U(s)$. We propose a point-wise lower bound:

$$h_{\text{pt}}(s) = \sum_{u \in U(s)} p(u) \cdot \big(t(s) + c(v(s), u)\big), \tag{2}$$

Lemma 4 *$h_{pt}(s)$ is admissible.*

Proof. Let π_b^* be an optimal completion from state s, let $t^*(u)$ denote the arrival time at each $u \in U(s)$ along π_b^*, and let h^* denote the optimal cost-to-go. Since the robot departs from $v(s)$ at time $t(s)$ and must travel to u before visiting it, the triangle inequality gives $t^*(u) \geq t(s) + c(v(s), u)$ for every $u \in U(s)$. Since $p(u) \geq 0$, we can derive:

$$h^* = \sum_{u \in U(s)} p(u)\, t^*(u) \geq \sum_{u \in U(s)} p(u)\big(t(s) + c(v(s), u)\big) = h_{\text{pt}}(s).$$

Thus $h_{\text{pt}}(s) \leq h^*$ and h_{pt} is admissible.

This helps ensure the proposed RPV* finds an optimal solution as elaborated later in Theorem 1.

3.3 F-RPV*

While RPV* finds optimal solutions, it may not run in *real-time* for large graphs. We thus leverage Anytime Focal Search [12] and build F-RPV* on top of RPV* to provide high-quality solutions within a limited time budget.

Focal List and Greedy Guidance In addition to the aforementioned `OPEN`, F-RPV* maintains a `FOCAL` list. Let $f_{\min}$ denote the minimum f-value of any state in `OPEN`. Then, `FOCAL` contains a subset of states s from `OPEN` whose f-values satisfy $f(s) \leq \omega \cdot f_{\min}$, where $\omega \geq 1$ is a sub-optimality bound. The states in `FOCAL` are then prioritized based on another criterion, the size of the visited set $|B(s)|$, from the largest to the smallest. By doing so, F-RPV* expands states that have visited more vertices at first, which leads to a complete path earlier.

Anytime Optimization and Pruning In addition, F-RPV* iteratively improves the quality of the solution over time. First, by popping from `FOCAL`, the search quickly completes a full path π_{best}. The cost $\xi_{\sum}(\pi_{\text{best}})$ is recorded as the current upper bound. Second, any state s in `OPEN` with $f(s) \geq \xi_{\sum}(\pi_{\text{best}})$ is pruned. As $f_{\min}$ increases and more states are processed, the algorithm continues to find improved paths with decreasing costs. Finally, if the search runs until OPEN is exhausted, F-RPV* matches RPV*'s optimal solution; otherwise it returns π_{best} with ω-bounded sub-optimality.

3.4 RPV* for Min-Max RPV

The aforementioned RPV* solves the min-sum RPV. RPV* can be modified as follows to solve the min-max RPV.

State Expansion For $\pi' = \pi \cup v_{k+1}$, $\xi_{\max}$ admits a recursive form:

$$\xi_{\max}(\pi') = \max_{i \in [k+1]} (p(v_i)t(v_i)) = \max(\xi_{\max}(\pi), p(v_{k+1}) \cdot (t(v_k) + c(v_k, v_{k+1}))) \quad (3)$$

A new state $s' = (u, g', t', B \cup \{u\})$ is generated as follows by expanding state s:

$$t' = t + c(v, u), \quad g' = \max(g, p(u) \cdot t').$$

Heuristic Unlike the previous heuristic $h(s)$ that estimates the cost-to-go, here the heuristic $h(s)$ provides a lower bound for the f-value directly:

$$h(s) = \max\Big(g(s), \max_{u \in U(s)} p(u) \cdot \big(t(s) + c(v(s), u)\big)\Big), \quad (4)$$

Lemma 5 *The min-max heuristic is admissible for Min-Max RPV.*

Proof. Let $\xi^*_{\max}$ denote the optimal objective value of any complete path extending state s. We show both terms in Eq. (4) are valid lower bounds.

First term. By Eq. (3), $\xi_{\max}$ is non-decreasing as vertices are added: for any extension π' of $\pi_a(s)$, $\xi_{\max}(\pi') \geq \xi_{\max}(\pi_a(s)) = g(s)$. Hence $\xi^*_{\max} \geq g(s)$.

Second term. Let π^*_b be an optimal completion from state s, and let $t^*(u)$ denote the arrival time at $u \in U(s)$ along π^*_b. Since the robot departs from $v(s)$ at time $t(s)$ and must travel to u before visiting it, the triangle inequality gives $t^*(u) \geq t(s) + c(v(s), u)$ for every $u \in U(s)$. Since $\xi^*_{\max} \geq p(u) \cdot t^*(u)$ for every $u \in U(s)$:

$$\xi^*_{\max} \geq \max_{u \in U(s)} p(u) \cdot t^*(u) \geq \max_{u \in U(s)} p(u) \cdot \big(t(s) + c(v(s), u)\big).$$

Taking the maximum of both lower bounds gives $\xi^*_{\max} \geq h(s)$, so $h(s)$ is admissible.

3.5 Analysis

Theorem 1 *RPV* is complete in the sense that (i) if the problem instance is solvable, RPV* returns a feasible optimal solution in finite time; (ii) if the problem instance is unsolvable, RPV* returns a failure in finite time.*

Proof. RPV* conducts systematic best-first search as A* [23], 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. By Lemma 3, such pruning only removes sub-optimal solutions since for any future solution from s, there must exist a better solution from s'. As a result, for any solvable problem instance, RPV* never eliminates the only feasible joint path unless a better path has already been found, and it terminates the search with an optimal solution. For an unsolvable problem instance, since the graph is finite, there is a finite number of possible joint paths. RPV* terminates after enumerating all joint paths and reports failure.

Theorem 2 *F-RPV* returns an ω-bounded sub-optimal solution for RPV.*

Proof. F-RPV* performs a systematic search over partial paths in G, and by Def. 3, dominance pruning never eliminates a path along an optimal solution. With an admissible heuristic, the f-value of any state s in `OPEN` is a lower bound of the true optimal solution cost, i.e., $f(s) \leq g_*$ with g_* being the true optimal cost of an optimal solution. The use of `FOCAL` search allows F-RPV* to select a state s with f-value within range $[f_{\min}, \omega f_{\min}]$ with $f_{\min}$ being the minimum f-value of any state in `OPEN`. At termination, the returned solution π_{best} corresponds to a state s with $|B(s)| = |V|$, where $h(s) = 0$ and therefore $g(s) = f(s) \leq \omega f_{\min} \leq \omega g_*$.

4 Application in Autonomous Exploration

Let $\mathcal{W} \subset \mathbb{R}^3$ denote a bounded 3D workspace to be explored, and let $\mathcal{W}_{\text{obs}} \subset \mathcal{W}$ and $\mathcal{W}_{\text{free}} = \mathcal{W} \setminus \mathcal{W}_{\text{obs}}$ denote the obstacles and free space respectively. Let $\mathcal{S} \subseteq \mathcal{W}$ represent the set of surfaces to be mapped, which is the boundary between obstacles and free space. Let $q \in \text{SE}(3)$ denote the pose of the on-board sensor, which perceives a volume $\mathcal{W}(q) \subseteq \mathcal{W}_{\text{free}}$. The subset of surfaces perceived at pose q is denoted by $\mathcal{S}(q) \subseteq \mathcal{S}$. A robot starts the exploration at $t = 0$. Let $\tau = \{q_1, \dots, q_l\}$ denote its trajectory in SE(3), where consecutive points are separated by a constant time interval t_Δ. A trajectory τ is *feasible* if it satisfies the robot's kinodynamic constraints. The exploration cost is defined as the total travel time $cost(\tau) = (l-1) \cdot t_\Delta$.

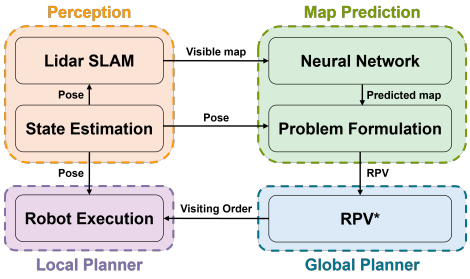


Fig. 3: System overview of the proposed exploration pipeline.

Problem 3 (Exploration) *The goal of the exploration problem is to find a feasible trajectory τ such that (i) all surfaces are perceived $\bigcup_{q\in\tau} \mathcal{S}(q) = \mathcal{S}$ and (ii) the exploration time $cost(\tau)$ is minimized.*

Fig. 3 shows the overall system design. We integrate RPV* into a frontier-based exploration pipeline similar to [3]. Our contributions at the system level are: (i) augmenting the viewpoint set with *predicted viewpoints* from a learned map prediction model (Sec. 4.1), (ii) replacing standard TSP-based ordering with F-RPV* (Sec. 4.2), and (iii) extending the local planner to navigate through predicted free space (Sec. 4.3).

4.1 Map Prediction

We employ the map prediction model ψ_θ from [4], which takes the current occupancy grid and outputs a predicted occupancy grid extending into unknown regions along with a per-cell confidence grid with the same spatial resolution. Predicted free cells outside the observed map define a *predicted free region*, in which predicted viewpoints are sampled (rather than only at frontiers, since predicted free space has no boundary with unknown space). Each predicted viewpoint v is assigned existence probability $p(v)$ equal to the average confidence of ψ_θ over the local neighborhood $\mathcal{N}(v)$ covered by v's sensor footprint:

$$p(v) = \frac{1}{|\mathcal{N}(v)|} \sum_{x \in \mathcal{N}(v)} \psi_\theta^{\text{conf}}(x), \tag{5}$$

where $\psi_\theta^{\text{conf}}(x) \in [0, 1]$ denotes the confidence at cell x. Observed frontier viewpoints, in contrast, are assigned $p(v) = 1$ since their underlying free space has been directly perceived.

4.2 Global Planning

At each planning cycle, we construct a complete graph $G = (V, E)$ using the robot's current pose, observed viewpoints, and predicted viewpoints as vertices. For each pair of vertices, the edge cost $c(e)$ is computed as the shortest collision-free path length on the current occupancy map; vertices unreachable from the robot are discarded. F-RPV* is invoked on G with a time budget (e.g., 1.5 s) for real-time operation, using its anytime variant.

4.3 Local Planning

The robot executes the tour by planning a collision-free trajectory to each successive viewpoint after obtaining the sequence from global planning. Since predicted viewpoints may lie in unknown regions, standard A^* restricted to the observed map cannot reach them. We therefore extend A^* to allow traversal of *predicted free* cells. If no feasible path is found within a fixed search budget, the viewpoint is skipped and reconsidered at the next planning cycle after the map updates.

5 Experiments

We evaluate the proposed RPV* and F-RPV* at three levels. We first study algorithmic properties, including the impact of the proposed heuristics (Sec. 5.1), the scalability of RPV* with respect to problem size (Sec. 5.2), and the anytime behavior of F-RPV* (Sec. 5.3). We then compare our solver against non-optimal baselines on RPV instances (Sec. 5.4). Finally, we integrate RPV* into a hierarchical exploration system and evaluate performance in simulation and real-world experiments (Sec. 5.5).

All experiments run on a computer with an Intel Ultra 9 275HX CPU, 32 GB RAM, and a NVIDIA 5080 GPU (16 GB VRAM). For algorithm-level experiments, RPV instances are generated by sampling $|V|$ vertices from three benchmark grid maps [24] and computing pairwise shortest-path distances as edge cost; each vertex (except the depot) is assigned an existence probability drawn uniformly from $[0, 1]$. Results are averaged over 20 instances per setting unless otherwise noted.

5.1 Impact of Heuristics

We fix $|V| = 15$ and compare RPV* with and without the proposed heuristics under both min-sum and min-max objectives. Since our heuristics are admissible, solution optimality is preserved in all variants, so the comparison

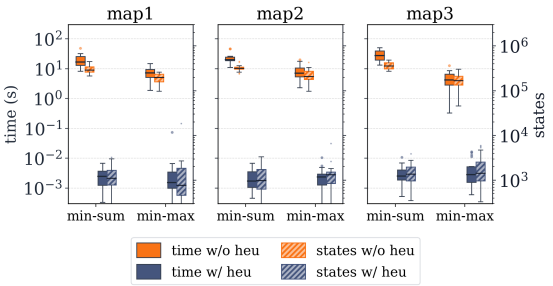


Fig. 4: Runtime and number of expanded states of RPV* with and without the proposed heuristic ($|V| = 15$).

focuses solely on runtime efficiency. Fig. 4 shows that the proposed heuristics reduce the mean runtime by approximately 4 orders of magnitude and the mean number of expanded states by approximately 2 orders of magnitude, indicating its effectiveness.

5.2 Scalability

We evaluate the scalability by running RPV* on 20 random instances per $|V| \in [15, 24]$. Fig. 5 reports the mean and min/max runtime at each size, where the runtime grows with $|V|$, consistently aligning with the NP-hardness. RPV* stays under 1 s up to $|V| \approx 21$, and remains tractable within the 60 s limit up to $|V| \approx 24$, which is enough for online exploration.

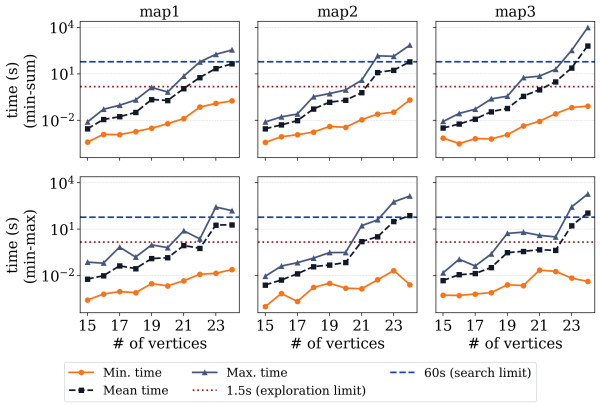


Fig. 5: Scalability of RPV*: runtime (min-sum top, min-max bottom) vs. number of vertices on three maps.

5.3 Anytime Behavior

We evaluate the anytime behavior of F-RPV* by tracking how its suboptimality bound ω evolves over runtime. We fix $|V|$=20 and 25, and run F-RPV* with a 3-second budget on 20 RPV instances. We adopt a budget longer than the one in Sec. 4.2 so as to observe the full convergence behavior.

As shown in Fig. 6, ω decreases steadily towards 1 across all settings, confirming that F-RPV* progressively tightens its quality guarantee as the search proceeds. For $|V| = 20$, the min-sum bound reaches 1 within roughly 0.1 s, while the min-max bound converges more slowly and approaches 1 near the end of the budget. For $|V| = 25$, convergence is slower under both objectives, and the bound typically remains around 2–3 within the budget.

5.4 Comparison with Non-Optimal Baselines

We compare RPV* against baselines in terms of solution quality, with a 30 s time limit per instance. The depot is fixed with $p(v_1) = 1$, and each remaining vertex is assigned a probability sampled uniformly at random from $[0, 1]$.

We consider three baselines. Let u be the current vertex. **Greedy** picks the unvisited vertex v that minimizes $c(u, v)/p(v)$, favoring nearby vertices with high existence probabilities. **LKH-3** [25] is a state-of-the-art solver for many TSP variants, which is applied to the original edge costs, ignoring $p(v)$ entirely.

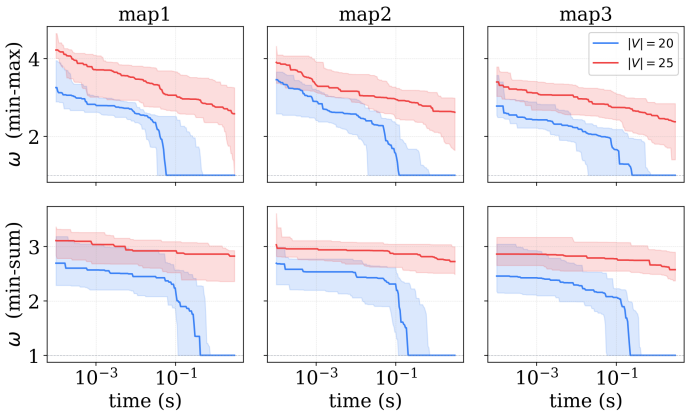


Fig. 6: Anytime behavior of F-RPV*. Top row: min-max objective; bottom row: min-sum objective. Each column corresponds to one map. Curves show the mean ω over 20 instances, with shaded regions indicating the min/max range.

p-LKH-3 adapts LKH-3 to RPV by constructing an asymmetric TSP instance with reweighted edge costs $\tilde{c}(u, v) = p(v) \cdot c(u, v)$, encoding the vertex's probability into each transition.

Fig. 7 shows the results. RPV* and F-RPV* achieve the lowest cost in most settings. Even against the *strongest* baseline in each objective, RPV* reduces the median solution cost by 9.3% under the min-sum objective (over LKH-3) and by 27.0% under the min-max objective (over Greedy) on all maps. Against the weakest baseline, the median gap widens to 54.1% and 96.1% respectively, and the largest per-instance gap reaches 108% (min-sum) and 235% (min-max). F-RPV* stays close to the optimum throughout, with a median gap of at most 6% relative to the solutions found by RPV* within the time limit, confirming that the anytime variant recovers near-optimal solutions despite the loose certified bound observed in Sec. 5.3. Notably, the weakest baseline in both objectives is p-LKH-3, a natural attempt to incorporate vertex probabilities into a state-of-the-art routing solver, which performs the worst with the highest median cost.

5.5 Exploration

Simulation Comparison We compare the exploration completion time and total traveled distance of our method as in Sec. 4 against a TSP-based exploration strategy. This baseline is similar to a recent method [4], which uses the same system as ours, except that the RPV* component is replaced by a vanilla TSP solver that treats all observed and predicted vertices as targets with $p(v) = 1$. We evaluate both systems in three simulated environments: indoor, forest, and village, and repeat each experiment 5 times from the same starting position to reduce variability. The results are summarized in Table 1.

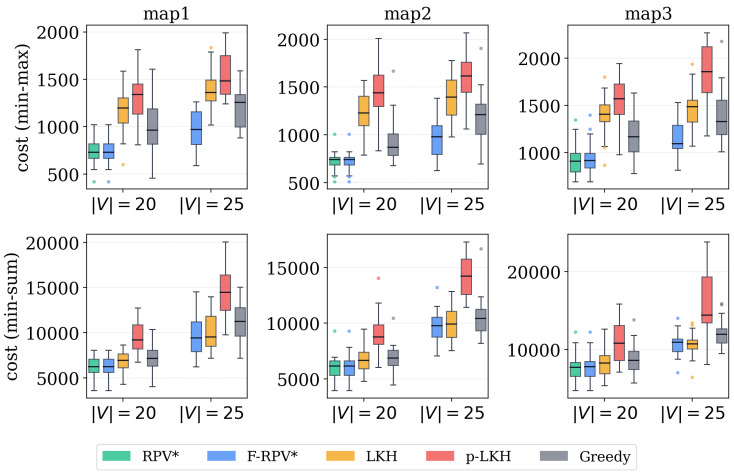


Fig. 7: Comparison with non-optimal baselines. Top row: min-max objective; bottom row: min-sum objective. Each column corresponds to one map. Boxplots aggregate solution costs over 20 instances per setting.

Our system consistently outperforms the baseline in both completion time and traveled distance across all three environments, with maximum improvements of 11.11% in time and 12.37% in distance. Since all system components except the global planner's solver are identical across the two conditions, the observed performance gains can be directly attributed to the probability-awareness of RPV*.

Table 1: Simulation results comparing our system against the TSP baseline across three environments. Bold marks the best result per row. Results are reported as mean (std) over 5 runs. Margin shows the percentage reduction of our system compared to the TSP baseline.

Environments and Metrics	Baseline	RPV*	Margin
Indoor (43m × 35m)			
Time (s)	595.4 (10.4)	**545.9 (22.3)**	8.31%
Length (m)	204.5 (3.5)	**199.0 (20.9)**	2.69%
Forest (100m × 100m)			
Time (s)	695.1 (43.2)	**617.9 (14.5)**	11.11%
Length (m)	986.9 (56.7)	**864.8 (17.3)**	12.37%
Village (80m × 120m)			
Time (s)	529.4 (18.6)	**472.5 (4.5)**	10.75%
Length (m)	992.7 (36.0)	**873.8 (6.5)**	11.98%

Real-world Validation We deploy our system on an AgileX Scout Mini vehicle (Fig. 1) for real-world validation of our method. The robot is equipped with a LIVOX Mid-360 LiDAR configured with a 10-meter detection range and a 360° horizontal field of view, and its maximum speed is limited to 1.25 m/s.

We adopt FAST-LIO2 [26] for both localization and mapping. Using this setup, we explore three different environments: a simple indoor scene, a complex indoor scene, and an

Table 2: Real-world exploration with RPV*.

Metrics	Simple Indoor (72m × 66m)	Complex Indoor (67m × 44m)	Outdoor (37m × 45m)
Time (s)	1178.7	1041.8	335.1
Length (m)	496.4	435.0	165.0

outdoor scene. Our system successfully completes the exploration task in all three environments, with the results summarized in Table 2 and the exploration trajectories and maps shown in Fig. 8. This demonstrates the practical applicability of RPV* in real-world scenarios, especially in complex indoor and outdoor environments.

6 Conclusion

This paper introduces a new problem RPV, where each vertex carries an existence probability, together with exact (RPV*) and anytime (F-RPV*) solvers that respectively provide optimality and bounded-suboptimality guarantees. Applied to exploration tasks, our method reduces exploration time and path length by up to 11.11% and 12.37% over a recent baseline method. Future work will explore a decentralized multi-agent extension of RPV.



Fig. 8: Real-world exploration results: (a) Simple Indoor, (b) Complex Indoor, and (c) Outdoor. Orange lines and cyan dots are robot trajectories and starting positions, respectively.

References

1. A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan, and M. Sudan, "The minimum latency problem," in *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pp. 163–171, 1994.
2. M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*. Mathematical Sciences Series, W. H. Freeman, 1979.
3. C. Cao, H. Zhu, Z. Ren, H. Choset, and J. Zhang, "Representation granularity enables time-efficient autonomous exploration in large, complex worlds," *Science Robotics*, vol. 8, no. 80, p. eadf0970, 2023.
4. K. Song, G. Chen, M. Tomizuka, W. Zhan, Z. Xiong, and M. Ding, "P^{2}explore: Efficient exploration in unknown cluttered environment with floor plan prediction," in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 13090–13096, IEEE, 2025.
5. R. R. Murphy, J. Kravitz, S. L. Stover, and R. Shoureshi, "Mobile robots in mine rescue and recovery," *IEEE Robotics & Automation Magazine*, vol. 16, no. 2, pp. 91–103, 2009.
6. J. P. Grotzinger, J. Crisp, A. R. Vasavada, R. C. Anderson, C. J. Baker, R. Barry, D. F. Blake, P. Conrad, K. S. Edgett, B. Ferdowski, *et al.*, "Mars science laboratory mission and science investigation," *Space science reviews*, vol. 170, pp. 5–56, 2012.
7. Y. Wang, H. He, J. Liang, Y. Cao, R. Chakraborty, and G. Sartoretti, "Cogniplan: Uncertainty-guided path planning with conditional generative layout prediction," *arXiv preprint arXiv:2508.03027*, 2025.
8. C. Ho, S. Kim, B. Moon, A. Parandekar, N. Harutyunyan, C. Wang, K. Sycara, G. Best, and S. Scherer, "Mapex: Indoor structure exploration with probabilistic information gain from global map predictions," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 13074–13080, IEEE, 2025.

9. 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, pp. 11–19, 2024.
10. Y. Lyu, C. Cao, J. Zhang, H. Choset, and Z. Ren, "Rpt*: Global planning with probabilistic terminals for target search in complex environments," *arXiv preprint arXiv:2601.12701*, 2026.
11. 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, pp. 171–179, 2025.
12. L. Cohen, M. Greco, H. Ma, C. Hernández, A. Felner, T. S. Kumar, and S. Koenig, "Anytime focal search with applications.," in *IJCAI*, pp. 1434–1441, 2018.
13. A. Bar-Noy and B. Schieber, "The canadian traveller problem.," in *SODA*, vol. 91, pp. 261–270, 1991.
14. 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.
15. P. Jaillet, *Probabilistic traveling salesman problems*. PhD thesis, Massachusetts Institute of Technology, 1985.
16. A. Garcia, P. Jodrá, and J. Tejel, "A note on the traveling repairman problem," *Networks: An International Journal*, vol. 40, no. 1, pp. 27–31, 2002.
17. B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97.'Towards New Computational Principles for Robotics and Automation'*, pp. 146–151, IEEE, 1997.
18. F. Bourgault, A. Makarenko, S. Williams, B. Grocholsky, and H. Durrant-Whyte, "Information based adaptive robotic exploration," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, pp. 540–545 vol.1, 2002.
19. M. Kulkarni, M. Dharmadhikari, M. Tranzatto, S. Zimmermann, V. Reijgwart, P. De Petris, H. Nguyen, N. Khedekar, C. Papachristos, L. Ott, R. Siegwart, M. Hutter, and K. Alexis, "Autonomous teamed exploration of subterranean environments using legged and aerial robots," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 3306–3313, 2022.
20. 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)*, pp. 1462–1468, IEEE, 2016.
21. Y. Cao, T. Hou, Y. Wang, X. Yi, and G. Sartoretti, "Ariadne: A reinforcement learning approach using attention-based deep networks for exploration," *arXiv preprint arXiv:2301.11575*, 2023.
22. R. Shrestha, F.-P. Tian, W. Feng, P. Tan, and R. Vaughan, "Learned map prediction for enhanced mobile robot exploration," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 1197–1204, IEEE, 2019.
23. 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.
24. N. R. Sturtevant, "Benchmarks for grid-based pathfinding," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 2, pp. 144–148, 2012.
25. K. Helsgaun, "An extension of the lin-kernighan-helsgaun tsp solver for constrained traveling salesman and vehicle routing problems," *Roskilde: Roskilde University*, vol. 12, pp. 966–980, 2017.
26. W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "Fast-lio2: Fast direct lidar-inertial odometry," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2053–2073, 2022.