Robotica (2025), **43**, pp. 4402–4428
doi:10.1017/S0263574725102956

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

Informative path planning for unmanned aerial vehicles using cost-benefit spanning tree

Wei-Hsiang Chiu and Kuo-Shih Tseng

Department of Mathematics, National Central University, Taoyuan City, Taiwan
Corresponding author: Kuo-Shih Tseng; Email: kuoshih@math.ncu.edu.tw

Received: 3 November 2024; **Revised:** 7 November 2025; **Accepted:** 7 November 2025;
First published online: 3 December 2025

Keywords: aerial robotics; path planning; motion planning; navigation; control of robotic systems

Abstract

Informative path planning (IPP) is one of key applications for unmanned aerial vehicles. It can be applied to terrain monitoring problems, which are to find the static targets from bird's eye view. In this research, the proposed algorithm generates the cost-benefit spanning tree (CBST) to boost the IPP performance. The CBST is able to generate different tree structures based on different parameters. The proofs show that the theoretical guarantees depend on the tree structures (e.g., minimal spanning tree and shortest path tree). The simulations and experiments demonstrate that the proposed method outperforms the benchmark approaches.

1. Introduction

The goal of Informative path planning (IPP) is to find the optimal path that maximizes the information subject to budget constraints. Various applications in robotics include sensor placement for indoor temperature measurements and precipitation prediction [1], search and rescue for multi-robot [2], locomotion for quadruple robots [3], and terrain monitoring for unmanned aerial vehicles (UAVs) [4]. This research focuses on IPP for terrain monitoring using a UAV. The UAV-based terrain monitoring problem is to detect the targets via a UAV as soon as possible.

If the IPP has routing constraints, this problem includes two NP-hard problems.[1] First, that the agent finds the maximal information sets K from S sets [5] is a set-covering problem [6]. Second, that the agent finds the least route from K subgoals is a traveling salesman problem (TSP) [7].

To address these challenges, several solution strategies have been proposed, among which two representative categories are branch-and-bound methods and iterative sampling approaches. In ref. [8], a branch-and-bound technique is applied to the IPP problem, where the large search tree is efficiently pruned using bounded parameters. Alternatively, iterative sampling approaches have been investigated in refs. [9] and [4], where UAVs are deployed to monitor terrains at multiple resolutions. Although these methods have shown practical effectiveness, they still lack theoretical performance guarantees.

To solve the routing constraint of IPP problems, the generalized cost-benefit (GCB) method [10] adopts TSP approximation algorithms and proves the theoretical guarantees in the routing constraint [11]. Maximizing submodular functions subject to some constraints (e.g. cardinality [5], additive budget [12], and routing [10]) could be solved by variant greedy algorithms[2] with theoretical guarantees [5, 13]. To further improve GCB performance, minimum spanning trees (MSTs) are adopted as the tree structure.

[1]The NP-hard problem means that finding the optimal solutions is computationally intractable. For example, the problem size is N. The N affects the exponential term of the computational load.

[2]Greedy algorithms find the maximal or minimal value at the current step without looking more horizons. Such algorithms are efficient but could not find optimal solutions.

© The Author(s), 2025. Published by Cambridge University Press.

https://doi.org/10.1017/S0263574725102956 Published online by Cambridge University Press

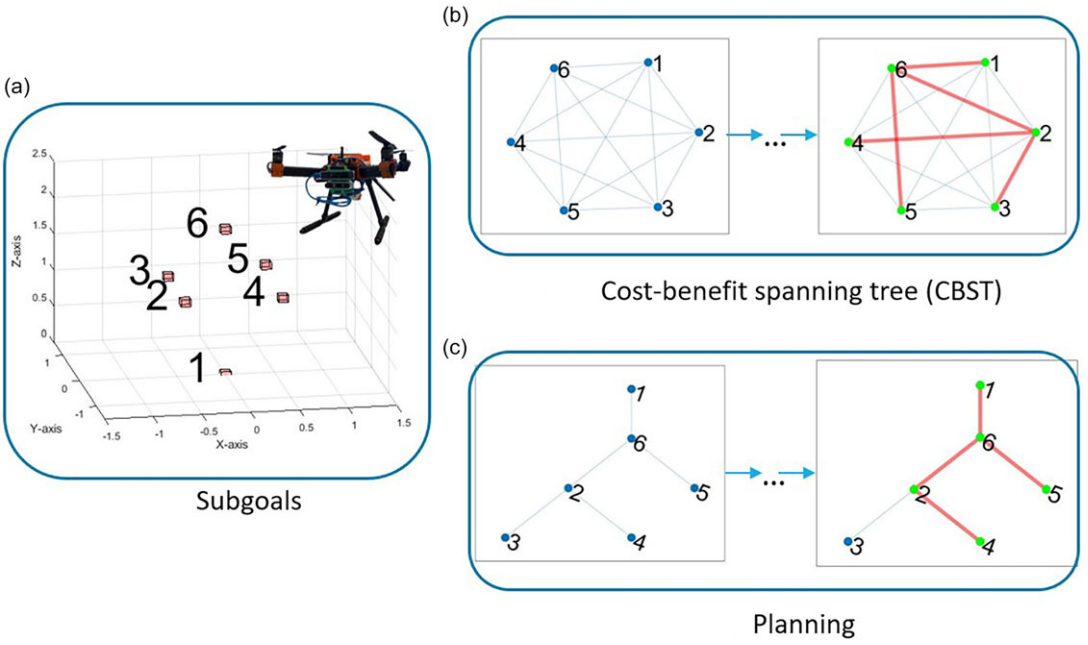


Figure 1. *Illustration of the proposed method. (a) Subgoals. The red cubes and decimal numbers represent subgoals and the index of subgoals, respectively. (b) The cost-benefit spanning tree. The green points, red lines, and decimal numbers represent nodes in spanning tree, edges in spanning tree, and the index of subgoals, respectively. (c) Planning. The green points and red lines represent the path nodes and path edges, respectively.*

The GCB-MST utilizes the submodularity of the spanning trees to boost the theoretical guarantee [14]. However, the MST could not be the best spanning tree.

To explore the spanning tree for UAV-based terrain monitoring problems, this research proposes cost-benefit spanning tree (CBST) algorithm, which generates subgoals as a cost-benefit objective. The parameter of CBST is able to generate the tree between the MST and the shortest path tree (SPT). To illustrate the CBST concept, as Figure 1 (a) shows, there is a UAV in the map. There are 6 nodes, including 1 source node (index 1). As Figure 1 (b) shows, the CBST generates the tree from six nodes. As Figure 1 (c) shows, the UAV plans the path to maximize the environmental coverage along the tree.

The contributions of this research are as follows: First, this is the first work to explore CBST for IPP problems. The proposed algorithm, CBST, is able to generate trees between the MST and the SPT. Second, this research further proves the performance bounds of spanning trees under different total curvatures. Third, the simulations and experiments demonstrate that the proposed approach outperforms the benchmark (e.g., Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [4] and GCB [10]).

The paper is organized as follows. Section 2 reviews the relevant work. Section 3 describes the background knowledge of this research. Section 4 introduces the problem formulation. Section 5 describes the proposed algorithms. Section 6 describes the experiments. Section 7 discusses the key issues of the proposed methods. Finally, Section 8 reports the conclusions and future work.

2. Related work

To clarify the key terms in this paper, the differences between path planning (PP), IPP, and coverage path planning (CPP) are as follows: The PP is to plan a path from location A to B [15]. The IPP is to plan a path, which maximizes the information subject to a constraint [8]. The CPP is to plan a path to completely cover a designated workspace [16]. This research focuses on IPP.

https://doi.org/10.1017/S0263574725102956 Published online by Cambridge University Press

The prior works of probabilistic search, IPP, submodular maximization problems, and the Prim-Dijkstra algorithm are discussed in this section.

2.1. Probabilistic search

Probabilistic search consists of perception and decision-making [18]. Perception is to estimate where the target is, while decision-making is to find the optimal path. In the perception part, Bayesian filter enables agents to estimate the probability distribution of the targets [19]. However, in the decision-making part, finding the optimal solution for this problem is NP-hard [20].

There are two steps in the perception of probabilistic search. First, a probabilistic search is used to construct a probabilistic map including the initial information. The probabilistic map is composed of cells. Each cell represents whether the target is located or not. Second, the agent runs the Bayesian filter to update the probabilistic cell of the target existing or not [21, 22].

Occupancy grid maps updated by Bayesian filter are the most commonly used for spatial sensing in perception [23]. In refs. [22] and [24], the researchers show searching for one target with different parameters using a Bayesian filter. In ref. [4], the researchers show the UAV is able to execute terrain monitoring in discrete environments using Bayesian search.

The advantage of probabilistic search is that the probabilistic model is adopted to deal with uncertainty of target motion. The disadvantage of probabilistic search is that finding the optimal actions along all possible paths is infeasible (see Table I) due to its exponentially exploded branches.

2.2. Informative path planning (IPP)

IPP is to find the optimal path for an agent to maximize the pre-defined information subject to budget constraints [25]. The IPP problems can be classified by (i) non-adaptive and (ii) adaptive planning strategies. If the agent has information about the environment in advance, and plans the path before taking off, it is called non-adaptive method [26], e.g., coverage methods [16, 27], Pareto optimization methods [28], and evolution algorithm methods [29]. On the other hand, if the agent is allowed to change its path online, it is called adaptive methods [30], e.g., continuous-space informative path planner method [31], adaptive submodularity with hypothesis pruning methods [32], hybrid planner [33], and non-myopic methods [2].

The IPP problem can be considered a data gathering mission involving sequential decision-making under uncertainty. This problem can be modeled as a Partially Observable Markov Decision Process (POMDP) [21], which is NP-hard. Although it is an NP-hard problem, the greedy approach can obtain near-optimal solutions [5].

In refs. [4] and [34], two-steps approach is adopted to adaptive plan strategies. First, the agents find the solutions greedily to maximize the reduction of Shannon's entropy in the map. It is similar to frontier-based approaches for map exploration problems [35]. Second, the agents optimize the subgoals by CMA-ES [9, 34]. CMA-ES is an evolutionary approach with generic global optimization [36]. Although the methods in refs. [4] and [34] are adaptive, there are no theoretical guarantees. In ref. [37], the researchers show that multiple autonomous underwater vehicles (AUVs) are able to collect environment data efficiently by maximizing the mutual information between the scalar field model and observations. In ref. [38], the researchers consider the semantic information of the focused environment to obtain the probability distribution of the target locations. Hence, it is called semantic-aware IPP.

Due to the popularity of large language model (LLM), semantic-aware IPP is proposed for object search. The semantic information is utilized for the target distribution to improve the IPP performance [38]. In ref. [39], semantic information is applied to segment the target with adaptive IPP. In ref. [40], without targets' layout and locations, the mobile robot is able to investigate spatial location relations between object and room type. The research shows that the potential applications combining LLM and IPP.

The advantage of IPP is that the general formulation can solve various problems (e.g., environmental inspection, search problems). Even if probabilistic search is a special case of IPP. The disadvantage of

https://doi.org/10.1017/S0263574725102956 Published online by Cambridge University Press

Table I. *Comparisons of search methods.*

Methods	Advantages	Disadvantages
Probabilistic search [17]	deal with uncertainties via target motion models	intractable for finding optimal solutions
Informative path planning [8]	works for various applications	need to solve NP-hard problems
Submodular maximization [1]	greedy approaches can give theoretical guarantees solutions	need to discretize the environments

IPP is that most of IPP are NP-hard problems, which are infeasible to find optimal solutions (see Table I). Hence, how to solve IPP with theoretical guarantees is important.

2.3. Submodular maximization problems

To solve IPP problems with theoretical guarantees, submodularity is the key. A set function is submodular if it follows the diminishing returns property. If the function is nondecreasing and submodular, greedy policies find the solutions with theoretical guarantees [5]. Various applications include map exploration [41, 42], collecting lake information [2], locating a non-adversarial target [24, 43], and placing sensors for indoor temperature prediction [1].

Although the submodularity of objective function is utilized, the routing constraint is still an NP-hard problem. In ref. [10], the researchers propose GCB algorithm for submodular maximization problems with routing constraints. It is proved that the GCB algorithm has $\frac{1}{2}(1-\frac{1}{e})\widetilde{OPT}$ theoretical guarantees where $\widetilde{OPT} \leq OPT$. In ref. [14], the researchers reformulate the routing problem as a MST to improve the guarantees to $\frac{1}{2}(1-\frac{1}{e})\overline{OPT}$. This approach is based on the submodularity of routing cost trees [11] and recovering set functions in the Fourier domain [24, 44], where $\widetilde{OPT} \leq \overline{OPT} \leq OPT$.

The advantage of submodular maximization problems is that the objective function with submodularity can be solved by greedy algorithms with theoretical guarantees. The disadvantage of submodular maximization is that the environment needs to be discretized as sets (see Table I). If the constraint is an NP-hard problem (e.g., routing problems), the theoretical guarantees of submodular maximization problems are degraded. To improve this issue, the submodular tree approaches need to be further explored.

2.4. Multi-agent planning

Due to the high maneuverability of UAVs, multi-agent planning is also a key technology. To cover the whole area of interest, the agent tries to minimize the maximum time cost of UAVs via linear programing formulations [45]. Simulations show that this method outperforms other methods in tasks.

To achieve missions with learning methods, multi-agent reinforcement learning approach is proposed to cooperatively control UAVs via extrinsic and intrinsic rewards [46]. This approach is similar to multi-agent deep deterministic policy gradient. Compared to manual rewards, the reward function consists of three parts: encoder, forward, and reward prediction error modules. The simulations demonstrate the effectiveness of this approach.

2.5. Prim-Dijkstra algorithm

In ref. [14], the MST is adopted to reformulate routing problems. However, different tree structures can lead to different solution qualities. The Prim algorithm is typically used to construct MSTs, whereas the Dijkstra algorithm is designed to compute SPTs. In ref. [47], the authors combine the Prim and Dijkstra constructions (referred to as PD) to balance two competing objectives: path length (PL) and total tree weight (TW). Here, PL denotes the distance from the source vertex to a given vertex along the current tree, while TW refers to the total weight of the entire tree. Later, in ref. [48], the PD-II algorithm is

https://doi.org/10.1017/S0263574725102956 Published online by Cambridge University Press

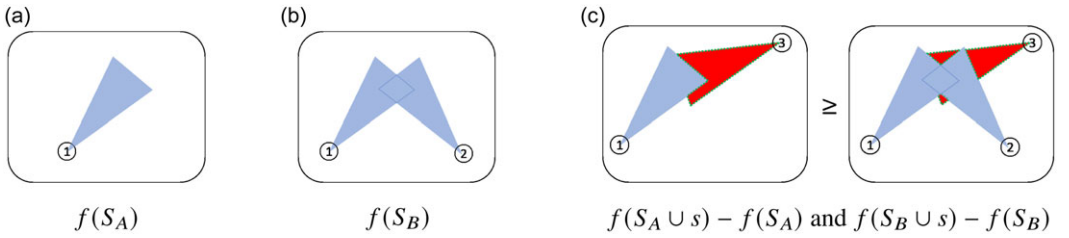


Figure 2. *Illustration of submodularity. The decimal number represents the selected sensor. The colorful and white areas represent the covered and uncovered areas, respectively. (a) $f(S_A)$ represents the covered area by S_A, where $S_A = \{1\}$. (b) $f(S_B)$ represents the covered area by S_B, where $S_B = \{1, 2\}$. (c) The green dash lines represent the submodular gain after adding s, where $s = \{3\}$. Left figure shows the $f(S_A \cup s) - F(S_A)$ and right figure shows that $f(S_B \cup s) - f(S_B)$..*

introduced, which incorporates the total detour cost and the degree of suboptimality in PL for each node to further enhance the original PD method.

The different tree structures could generate different solutions and guarantees of IPP. This research will explore their theoretical guarantees.

3. Background knowledge

The section contains submodularity, the lower bound of GCB under the tree-structured graph, PD algorithm, and extended probability of detection (EPD).

3.1. Submodularity

The definition and illustration of submodularity are as follows:

Definition 1 (Submodularity [5]). *Given a finite set $S = \{1, 2, \ldots, N\}$, a submodular function is a set function $f : 2^N \to \mathbb{R}$ that satisfies the diminishing return property. For every $S_A, S_B \subseteq S$ with $S_A \subseteq S_B$ and every $s \in S \setminus S_B$,*

$$f(S_A \cup s) - f(S_A) \geq f(S_B \cup s) - f(S_B) \tag{1}$$

holds.

To illustrate the concept of submodularity, an example is shown in Figure 2. There are three ground sets ($S = \{1, 2, 3\}$). $S_A = \{1\}$ and $S_B = \{1, 2\}$ represent the selected two sets, respectively. The set $S_B = \{1, 2\}$ means that the sensors are selected at location ① and ②. $f(S_A)$ and $f(S_B)$ mean the coverage of sensor at location ① and ① ② (see Figure 2(a, b)), respectively. The submodular gain of S_A and S_B after adding a set $s = \{3\}$ is represented by the green dashing lines (see Figure 2(c)). It is clear that the coverage function satisfies the diminishing return property Eq. (1). Alternatively, the objective function of maximizing coverage is submodular. Greedy approaches can generate near-optimal solutions even if maximal coverage is NP-hard problem.

3.2. Lower bound of GCB

To further apply submodularity to routing constraints, some definitions and the lower bound of GCB are introduced as follows:

Definition 2 (Total curvature [49]). *Given a finite set $S = \{1, 2, \ldots, N\}$, a monotone submodular function f, the total curvature of f is defined as:*

$$\kappa_f = 1 - \min_{s \in S : f(\{s\}) > 0} \frac{f(S) - f(S \setminus \{s\})}{f(\{s\})}. \tag{2}$$

If the $\kappa_f = 0$, then f is modular.

https://doi.org/10.1017/S0263574725102956 Published online by Cambridge University Press

Definition 3 (The largest size of feasible solution K_c [10]). *Given a ground set S and cost function c, K_c is defined as:*

$$K_c = \max_{X \subseteq S} \{|X| \; |c(X) \leq B\}, \tag{3}$$

where B is budget, c is the cost function.

Theorem 3.1 (Lower bound of GCB under the tree-structured graph [14]). *Given a submodular monotone set function f and a tree-structured graph cost function c, the GCB approach is to maximize f subject to the budget B. The performance of a set X achieves*

$$f(X) \geq \frac{1}{2}\left(1 - \frac{1}{e}\right) f(\overline{X}), \tag{4}$$

where

$$\overline{X} = \arg\max_{X} \left\{ f(X) | c(X) \leq B \left(1 - \kappa_c + \frac{\kappa_c}{K_c}\right) \right\}. \tag{5}$$

3.3. Prim and Dijkstra (PD) algorithm

MST is a spanning tree of a weighted undirected graph that minimizes the total weight (TW). Prim's algorithm is a classical greedy method for constructing an MST. In contrast, Dijkstra's algorithm is designed to compute the shortest path from a source node to all other nodes in a weighted graph, and is commonly used to generate a SPT. To integrate the properties of both PD, the following objective function is introduced:

Definition 4 (The objective function of PD algorithm [47]). *The v_i and v_j of PD algorithm are chosen to minimize*

$$(\alpha \cdot l_i) + d_{ij} \; s.t. \; v_j \in T, v_i \in V - T, \tag{6}$$

where $\alpha \in [0, 1]$ is a tuning parameter, l_i is the length from source (start point) to node v_i, d_{ij} is the distance between v_i and v_j, V is all vertices in graph, and T is current growing tree.

Prim-Dijkstra (*PD*) algorithm [47] can generate different types of spanning trees. If $\alpha = 0$, PD algorithm is Prim algorithm; if $\alpha = 1$, PD algorithm is Dijkstra algorithm. As Figure 3(a) shows, there are six subgoals on this map. As Figure 3(b) shows, the Prim algorithm generates the MST. As Figure 3(c) shows, the Dijkstra algorithm on this map generates the SPT. In the *PD* algorithm, different parameters generate different spanning trees.

3.4. Extended probability of detection (EPD)

Informative path planning (IPP) problems can be formulated as POMDP problems [50], which are NP-hard. Hence, probability of detection model is proposed. However, the assumptions of probability of detection are not realistic, i.e., the agent only moves to neighboring cells, and the sensor has no overlapping coverage. The extended probability of detection (EPD) was proposed [51] to apply to real world. The definition of EPD is as follows:

Definition 5 (Extended probability of detection (EPD) [51]). *The agent gets the information z. The assumptions of EPD are as follows:*

(i) *There is no target detection ($z = 1$) along the path.*

(ii) *The sensing overlapping is available.*

(iii) *The agent could move to any subgoals.*

https://doi.org/10.1017/S0263574725102956 Published online by Cambridge University Press

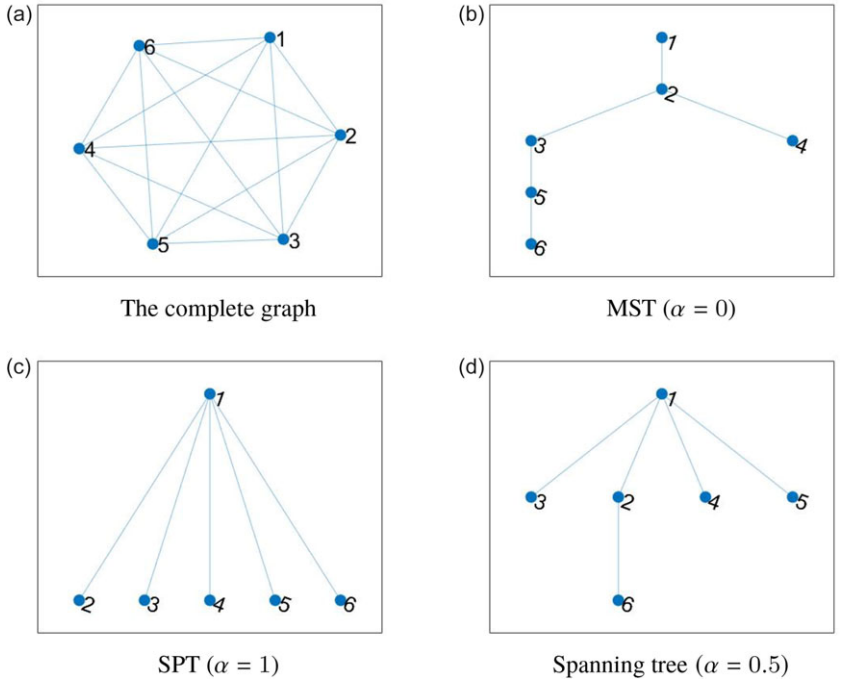


Figure 3. *Illustration of the PD algorithm. (a) The complete graph. The blue points, blue edges, and decimal numbers represent subgoals, paths, and the index of subgoals, respectively. (b) Minimum spanning tree (MST). (c) Shortest path tree (SPT). (d) The tree between MST and SPT. Tuning the α parameter generates different spanning trees.*

The equation of the cumulative EPD is defined as

$$f_p(S_g) = \sum_{i=1}^{T} P(S_{g,i}) \cdot g, \tag{7}$$

where f_p is the cumulative EPD along the path, g is glimpse function and $P(S_{g,i})$ is the probability of covered cells at the i-th subgoal. The value of glimpse function can be calculated from the confusion table.

As Figure 4 shows, the UAV's sensing area is not fitting in its local cell. Due to the availability of sensing overlap, the probability of sensing area is recomputed. The assumption is that the UAV moves along the path without any detections ($z = 0$). Finally, the UAV can navigate to any subgoals instead of neighborhood areas.

4. Problem formulation

The terrain monitoring problem is defined as follows [9]: As Figure 5 shows, there are 6 subgoals in a 3D map. The UAV can visit subgoals sequentially to inspect the static objects on the terrain. The detection area of higher subgoals (e.g., location 6) is larger, while that of lower subgoals (e.g., location 2) is smaller. However, the detection rate of higher subgoals (e.g., location 6) is lower due to the image quality. The goal is to maximize gain in some information-theoretic measure within a time or path budget.

The terrain monitoring problem can be reformulated as an IPP problem as follows: Given the ground set S, and $|S| = N$, the goal is to find subgoals that maximize the probability of detection subject to the routing constraints.

$$\max f(X) s.t. \quad c(X) \leq B, \tag{8}$$

https://doi.org/10.1017/S0263574725102956 Published online by Cambridge University Press

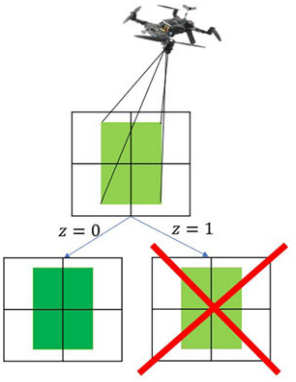


Figure 4. *Illustration of EPD assumption.*

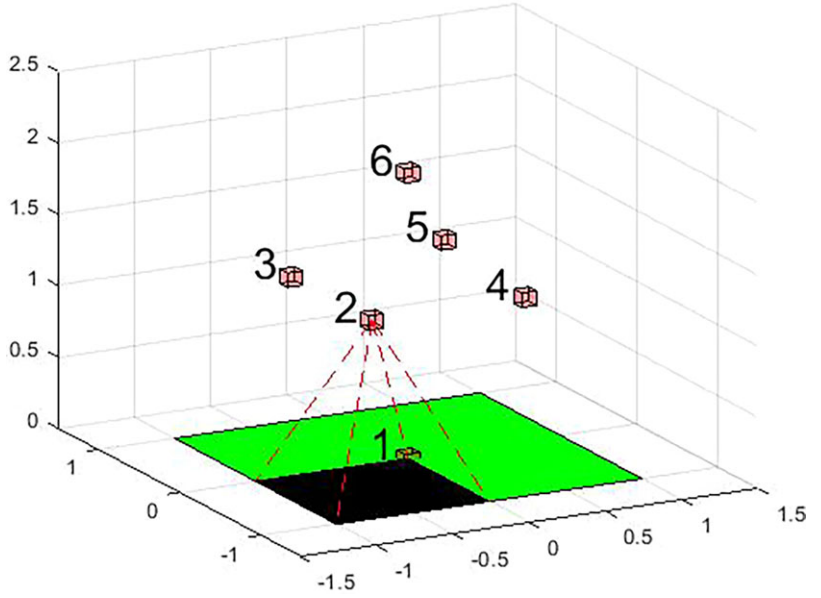


Figure 5. *Illustration of terrain monitoring. The red cubes and decimal numbers represent subgoals and indexes, respectively. The green and black regions represent lower probability and higher probability, respectively.*

where $X \subseteq S$, f is cumulative probability of detection function $f{:}2^N \rightarrow \mathbb{R}^+$, c is the routing cost function $c{:}2^N \rightarrow \mathbb{R}^+$, and B is the budget. The environment is known, hence f and c are oracles that output probability and routing cost values, respectively.

The assumptions of the problem formulation are as follows: First, the ground set of subgoal (S) and the budget limit (B) are given. Second, the cumulative probability of detection function (f) and the routing cost function (c) can be computed via the map and sensor information. Third, the theoretical guarantees only in the discrete subgoal level. Therefore, the motion control level is not considered (e.g., UAV dynamics).

4.1. Cost-benefit spanning tree (CBST)

This problem can be reformulated as a submodular maximization problem. It can achieve theoretical guarantees via GCB approaches. If the routing route is tree structure, the theoretical guarantees will be tighter than GCB [14]. However, how to find an efficient spanning tree structure is a key issue.

https://doi.org/10.1017/S0263574725102956 Published online by Cambridge University Press

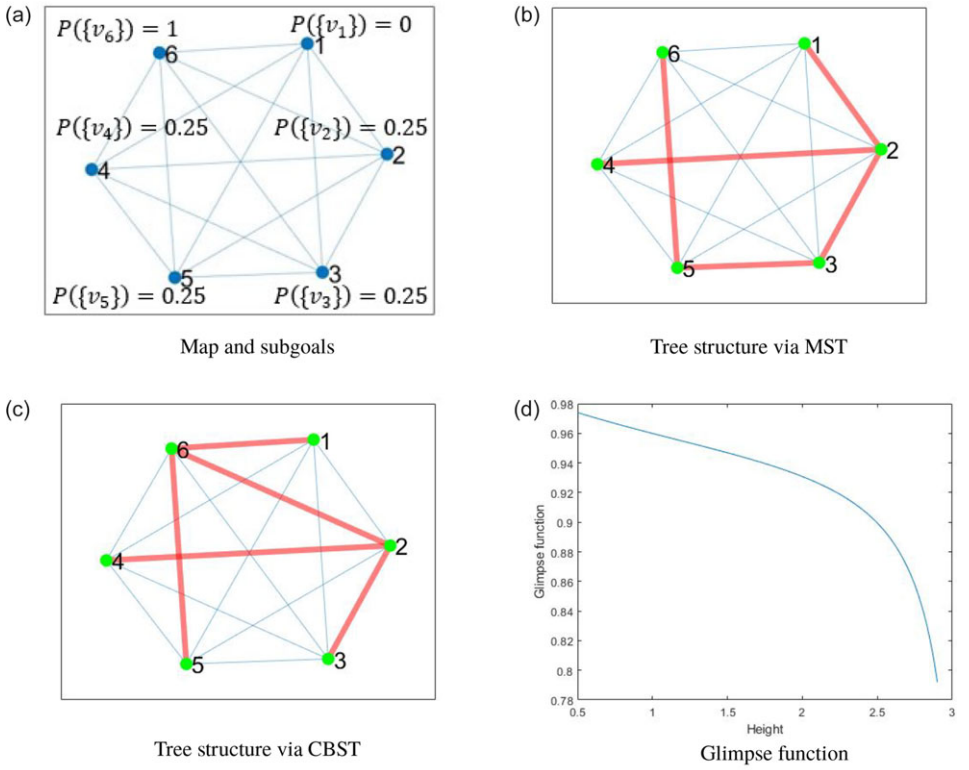


Figure 6. *Example of MST and CBST. The vertex 1 is the source. The map is as same as Figure 3 (a). (a) Map and subgoals. The blue points, blue edges, function, and decimal numbers represent subgoals, paths, probability of each nodes, and the index of subgoals, respectively. (b) Tree structure via MST. (c) Tree structure via CBST. (d) Glimpse function (g).*

Regardless of the distance between the information-rich vertex and the source node, the agent will always navigate to this vertex when using the GCB algorithm with a traditional shortest-path-tree (SPT)–based cost function, which can be inefficient for search tasks. Conversely, when the GCB algorithm is applied with an MST-based cost function, the agent may skip this information-rich vertex if it is located far from the source. To overcome this limitation, CBST further incorporates the benefit term f into the cost design.

The concept of CBST is similar to that Prim-Dijkstra algorithm. The major difference is that the objective function of CBST is cost-benefit. The objective function of CBST is

$$\max \left(\frac{f(v_j|T)}{\alpha \cdot l_i + d_{ij}} \right), \quad s.t. \quad v_i \in T, v_j \in V - T, \tag{9}$$

where f is accumulated extended probability of detection function in Eq. (7), $f(v_j|T)$ is equivalent to $f_p(S_g)$, p is implicit. $f(v_j|T)$ represents the agent go through v_j with T tree (path). The other notations are the same as Eq. (6) [47].

To illustrate the MST and CBST, an example is shown in Figures 5 and 6. There are 6 nodes, including a source node (index 1), in the map. As Figure 5 shows, the black area has less probability than green area after agent went to v_2 and scanned the area. The scanning areas of v_6 overlaps that of v_2, v_3, v_4, and v_5 since the altitude of v_6 is higher than that of v_2, v_3, v_4, and v_5. The probability of each vertex is $p(v_2) = 0.25, p(v_3) = 0.25, p(v_4) = 0.25, p(v_5) = 0.25$ and $p(v_6) = 1$. Notice that the total probability is over 1 because there is overlapping.

https://doi.org/10.1017/S0263574725102956 Published online by Cambridge University Press

Algorithm 1: Cost-benefit spanning tree.

Input:
$G = (V, E, w)$: undirected graph
s: the start point
f: objective function
Output: S: spanning tree
Parameter: α (between 0 and 1)

1: $Q = V$ #all of vertices in G
2: $Q = Q \setminus \{s\}$
3: $S = \phi$
4: **While** $Q \neq \phi$ **do**
5: let d_{qk} be the cost edge;
such that $q \in Q$ and $k \in V \setminus Q$
6: pl_q is distance from source to v_q
7: maximize $(\frac{f}{\alpha \times pl_i + d_{iu}})$ where $i \in Q, u \in V \setminus Q$
8: $Q = Q \setminus \{i\}$
9: $S = S \cup (u, i)$
10: **end while**

The tree structure of Figure 5 is shown in Figure 6. v_6 is far from v_1 in MST structure (see Figure 6(b)). On the contrary, v_6 is nearby v_1 in Figure 6(c) in CBST structure. Under such situations, GCB-CBST performs better than GCB-MST.

On the other hand, the UAV height affects the detection rate due to the image quality. As Figure 6(d) shows that the glimpse function (g) is to describe the relationship between the UAV height and detection rate. The f in Eq. (9) is calculated by $p(v_i)$, path and glimpse function (g), where $i \in \{1, \ldots, N\}$.

4.2. Algorithm

There are two algorithms: CBST algorithm and terrain monitoring algorithm. The Algorithm 1 is to generate the CBST (e.g., from Figure 1 (a) to (b)). The Algorithm 2 is to generate the path using GCB algorithm (e.g., from Figure 1 (b) to (c)).

In Algorithm 1, the inputs are an undirected graph (G), the source (s). The output is the spanning tree. Line 1 is to build Q (vertices in G). Lines $2-3$ are initializations. Line 5 is to find the distance connecting v_q and v_k, where v_q belongs to the current growing tree (S), and v_k does not belong to the current growing tree. Line 6 is to find the path length from the source vertex to v_q along the current growing tree (S). Line 7 is to maximize f/c, where c is the cost function about Prim's algorithm and Dijkstra algorithm. Lines $8-9$ are to update current tree and spanning tree.

The inputs of Algorithm 2 are S, s, f, c, and B. S, s represent spanning tree from Algorithm 1 and the start point, respectively. f, c represent the cumulative probability of detection and cost function from spanning tree S, respectively. B represents the cost budget, respectively. The output π represents the subgoal set with budget constraint. Lines $1-3$ are initializations. Lines $4-9$ are to find maximum cost-benefit point in the spanning tree. Lines $10-12$ are to select the subgoal satisfying the budget constraint. Line 13 is to avoid that the agent select the subgoal repeatedly.

The primary computational steps involve iterating over vertices and edges in the spanning tree S, with each major step taking $O(|V|)$ or $O(|E|)$. The while loop in Line 4 iterates at most $|V| - 1$ times, and the maximizing function in line 7 finds at most $|E|$ times. Therefore, the overall time complexity of Algorithm 1 is $O(|V||E|)$. Similarly, the time complexity of Algorithm 2 is $O(|V||E|)$.

5. Theoretical bound of CBST

Since the CBST could generate different spanning trees according to α parameter in Algorithm1, this section proves the theoretical guarantees of different spanning trees, SPT and MST.

https://doi.org/10.1017/S0263574725102956 Published online by Cambridge University Press

Algorithm 2: Terrain monitoring using CBST

Input:
$S = (V, E, w)$: spanning tree
s: the start point
f: objective function
c: cost function from spanning tree S
B: budget
Output: π: subgoal set

1: $G := \phi$
2: $\pi := \phi$
3: $V' = V$
4: **while** $V' \neq \phi$ **do**
5: **For** $X \in V$ **do**
6: $\Delta_f^X := f(G \cup X) - f(G)$
7: $\Delta_c^X := c(G \cup X) - c(G)$
8: **end for**
9: $X^* = argmax\{\frac{\Delta_f^X}{\Delta_c^X}\}$
10: **if** $c(G \cup X^*) \leq B$ **then**
11: $\pi = \pi \cup X^*$
12: **end if**
13: $V' = V' \setminus X^*$
14: **end while**

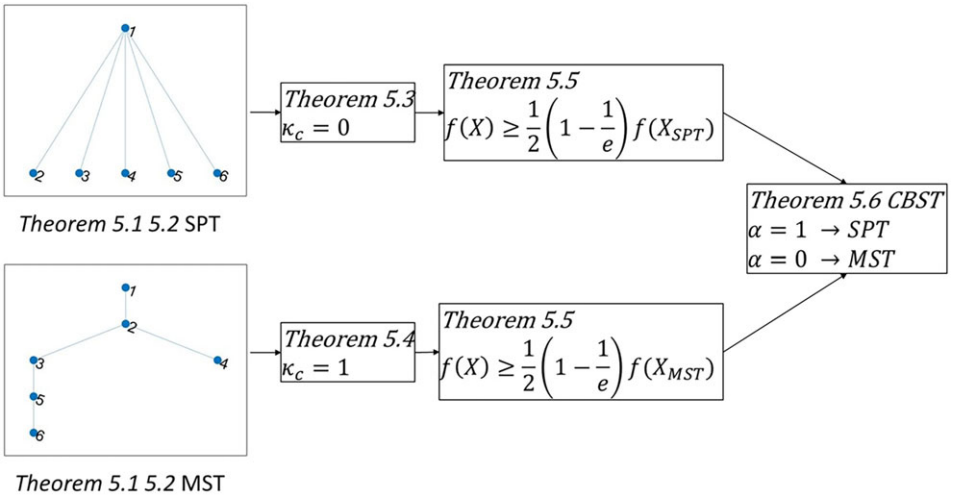


Figure 7. *Illustration of the proof flow.*

To introduce the theoretical bound of CBST, the proof flow is as follows: As Figure 7 shows, the goal is to prove the theoretical guarantees of SPT and MST. κ_c is the key parameter of the theoretical guarantees function and κ_c is relevant to the height of SPT and MST (Theorem 5.2). Hence, Theorems 5.3 and 5.4 prove κ_c of SPT and MST are 0 and 1, respectively. Then, κ_c is substituted into the Theorem 3.1 to derive the theoretical bound of SPT and MST (Theorem 5.5). Theorem 5.6 shows that the α in Algorithm 2 decides the tree structures (e.g., MST or SPT) and their theoretical bounds.

Theorem 5.1 (κ_c in tree structure [52]). *Given the tree structure of the cost function c, then* κ_c *either equals to* 0 *or* 1.

https://doi.org/10.1017/S0263574725102956 Published online by Cambridge University Press

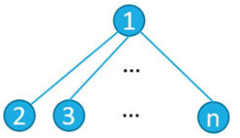


Figure 8. *Illustration of tree height of Theorem 5.3.*

Since the κ_c in tree structure is equal to 1 or 0, the subsequent theorems focus on exploring the relationship between κ_c and the height of the tree (H).

Theorem 5.2 (The relationship between κ_c and the height of trees). *Given the tree structure of the cost function and the height of the tree (H),*

if $H = 1$, $\kappa_c = 0$;

if $H > 1$, $\kappa_c = 1$.

Proof. W.L.O.G., we assume v_1 is the source node. To analyze the κ_c, two cases of the height of tree are as follows:

Case 1: ($H = 1$)

$\because H = 1$.

v_1 connects to the other $n - 1$ nodes (see Figure 8).

$\therefore \forall v_i$ are only connecting v_1, where $i \in \{2, \ldots, n\}$, n is the amount of nodes.

$\Rightarrow c(v_i) = ||v_i - v_1||, i \in \{2, \ldots, n\}$

$\Rightarrow \kappa_c = 0$ Eq. (2)

Case 2: ($H > 1$)

$\therefore \exists$ the height is equal to 2 or greater than 2.

Since the case of $H > 2$ includes the case of $H = 2$, only the case that the height is 2 will be proved.

W.L.O.G., assume v_2 is the node of depth $= 1$, v_3 is the node of depth $= 2$, and the node between v_1 and v_3 is v_2.

Let $S = \{v_1, v_2, v_3\}$, and $s = \{v_2\}$

$\because c(S) = 2 * (||v_1 - v_2|| + ||v_3 - v_2||)$ and

$$c(S \setminus \{s\}) = 2 * ||v_1 - v_3|| = 2 * (||v_1 - v_2|| + ||v_3 - v_2||)$$

The first equation is by definition, and the second one is by the tree structure.

There is only one path between v_1 and v_3.

$\therefore c(S \setminus \{s\}) = c(S)$

By definition of κ_c Eq. (2),

$\min_{s \in S: f(\{s\}) > 0} \frac{f(S) - f(S \setminus \{s\})}{f(\{s\})} = 0$

$\therefore \kappa_c = 1$ □

The following proofs focus on the height of SPT and MST. If height of SPT and MST is $H > 1$ or $H = 1$, κ_c of SPT and MST are proved.

Theorem 5.3 (κ_c in SPT). *Given an empty map and the SPT structure of the cost function c, then* $\kappa_c = 0$.

Proof. Given a complete graph $G = (V, E)$, $V = \{v_1, \ldots, v_n\}$

$\because \forall i \neq j \neq k \in \{1, \ldots, n\}$, $||v_k - v_i|| + ||v_i - v_j|| \geq ||v_k - v_j||$

the equality only holds in the v_i, v_j, and v_k in a straight line.

$\therefore$The height of shortest spanning tree is 1, i.e. the number of maximal edges between the source node and the others is 1.

$\because$ the height of shortest spanning tree is 1.

$\therefore$ by Theorem 5.2, $\kappa_c = 0$ □

As Figure 8 shows, the vertex 1 is the source node, and the others will be in level 2 due to triangle inequality.

https://doi.org/10.1017/S0263574725102956 Published online by Cambridge University Press

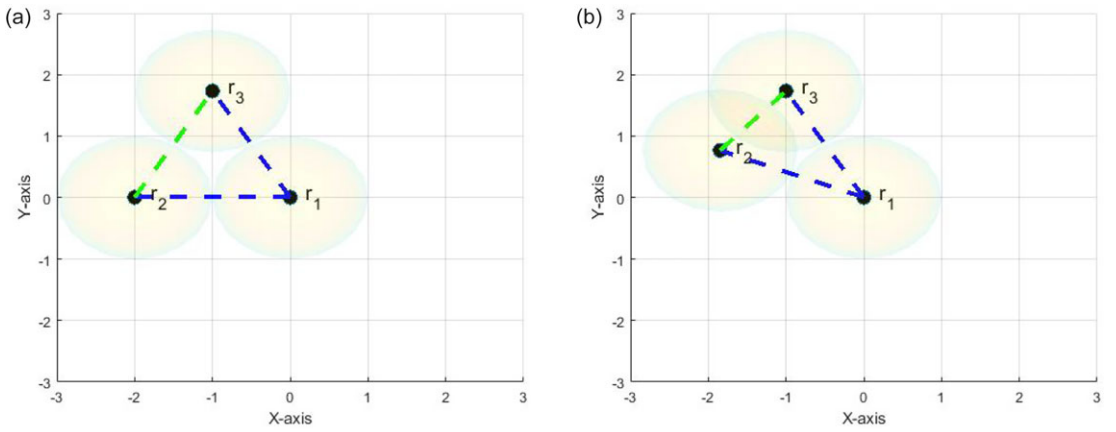


Figure 9. *Illustration of overlapping balls in Theorem 5.4. The black points, the decimal numbers, the blue lines, and the green line(s) represent the subgoals, the indexes of subgoals, the distance between source node (r_1) and the other nodes (r_2, r_3), and the distance between the other nodes (r_2 and r_3), respectively. (a) The non-overlapping case. (b) The overlapping case.*

Since the height of MST could depend on the number of subgoals, the kissing number problem is applied for the proof. The kissing number problem investigates the maximum number of non-overlapping spheres that can touch a central sphere in a given space. It aims to determine the highest possible number of points on the surface of a sphere that can be equidistant from a fixed central point.

Theorem 5.4 (κ_c in MST). *Given an empty map and the MST structure of the cost function c, the* $\kappa_c = 1$ *when*

$$(i)|n| > 7 \text{ in } 2D \text{ environments}; \tag{10}$$

$$(ii)|n| > 10 \text{ in } 3D \text{ environments}. \tag{11}$$

Proof. Given complete graph $G = (V, E)$, $V = \{v_1, \ldots, v_n\}$

W.L.O.G., we assume v_1 is the source node.

2D and 3D cases are discussed (see Figures 9 and 10).

Case (i): 2D environments

Proof by contradiction is as follows.

$|S| \leq 7$ in 2D environment.

The subgoals are arranged in the pattern of vertices of a hexagon, and the source node is in the center of a hexagon (see Figure 11).

$$\therefore \forall j, k \in \{1, \ldots, n\}, ||v_1 - v_j|| \leq ||v_j - v_k||. \tag{12}$$

By Eq. (12) and the property of MST, the height of MST is equal to 1.

$\Rightarrow \kappa_c = 0$. (by Theorem 5.2)

Hence, $|S| > 7$.

Case (ii): 3D environments

This problem can be seen as

max (N) s.t. $||v_1 - v_j|| \geq ||v_j - v_k||, j, k \in \{1, \ldots, N\}$. When $n > N$, it represents $H > 1 \Rightarrow \kappa_c = 1$

It is similar to the kissing number problem. $k(d)$ denotes the highest number of non-overlapping spheres in $\mathbb{R}^d$ that can touch another sphere of the same size. $k(3) = 12$ has been proved [53], which means the maximal kissing number is 12 in 3D space. The relationship between the kissing balls and

https://doi.org/10.1017/S0263574725102956 Published online by Cambridge University Press

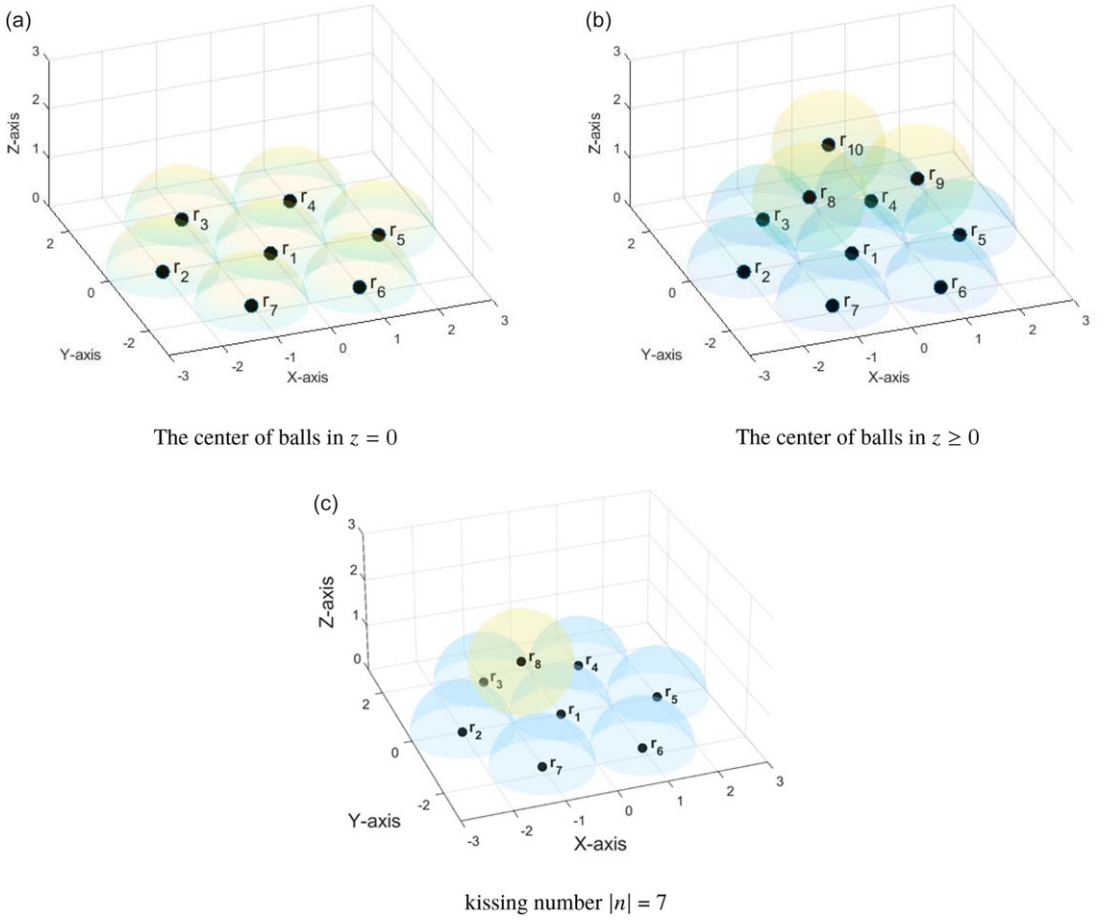


Figure 10. *Illustration of kissing balls of Theorem 5.4.*

the tree is as follows: As shown in Figures 9 and 10, the central sphere (r_1) represents the root of the tree. The highest possible number of points on the surface of a sphere ($r_1, \ldots, r_N$) corresponds to the number of points (N) (see Figure 9(a)). The overlapping spheres correspond to the distances between these points (green lines) that are smaller than the distances between these points and root (blue lines) (see Figure 9(b)).

The source node is placed on the surface ($x = y = z = 0$), and the other nodes are placed above the surface ($x \in \mathbb{R}, y \in \mathbb{R}$ and $z \geq 0$). First, there are at most six points in xy plane (see Figure 10(a)). Second, since $k(3) = 12$, the kissing number in the upper xy plane and in lower xy plane are 3 and 3, respectively (see Figure 10(b)). It represents $v_2, \ldots, v_7$ are points in xy plane, and v_8, v_9, v_{10} are upper xy plane. $\because$ There are no overlapping spheres.

$\Rightarrow ||v_1 - v_j|| \leq ||v_j - v_k||, \forall j \neq k \in \{2, \ldots, 10\}$

Finally, there are six balls in xy plane and three balls above xy plane.

$\Rightarrow$ there are 9 balls satisfying conditions.

$\Rightarrow$ the height of MST is greater than 1 if the number of subgoals greater than 10 (including root) in a three-dimension environment.

$\Rightarrow \kappa_c = 1.$ □

To give examples of the proofs, 2D and 3D cases are as follows: As Figure 11(a) shows, the source node is at the intersection point between two solid lines. If the edges of graph are 7 and the subgoals are the vertices of regular heptagon, the solid blue line is greater than the dashed line. It implies that if

https://doi.org/10.1017/S0263574725102956 Published online by Cambridge University Press

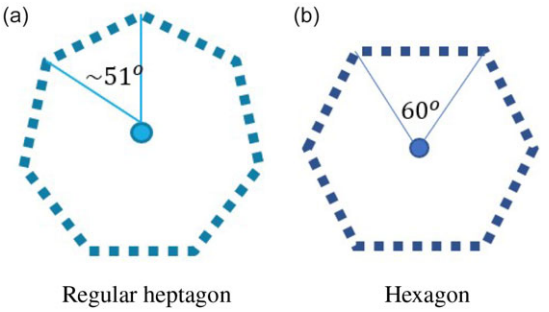


Figure 11. *Illustration of minimum numbers of Theorem 5.4.*

$|S| > 7, H > 1$ for MST; if the number of graph edges is 6 and the subgoals are in the vertices of hexagon, the solid blue line is less than or equal to the dashed line in Figure 11(b). It implies that if $|S| = 7, H = 1$ for MST.

Note that if $|S| \leq 10$ in 3D environment (see Figure 10(c)), either $\kappa_c = 0$ or $\kappa_c = 1$ is possible. As Figure 6 shows, there are 6 nodes in graph. The height of MST in Figure 6 is 4, i.e., $\kappa_c = 1$.

As Figure 10(a) shows, there are at most six non-overlapping balls in xy plane. By $k(3) = 12$ [53], there are three centers upper xy-plane (see Figure 10(b)). There are at most 10 points (including central point) satisfying $||v_1 - v_j|| \leq ||v_j - v_k||, \forall j \neq k \in \{2, \ldots, 10\}$.

Theorem 5.5 (The theoretical guarantees of GCB-SPT and GCB-MST). *By Theorems 5.3 and 5.4, given a submodular monotone set function f, a MST cost function c_{MST} with the condition of subgoals, and a SPT cost function c_{SPT}, the GCB approach is to maximize f subject to the budget B. The performance of a set X with c_{MST} and of a set X with c_{SPT} achieve*

$$f(X) \geq \frac{1}{2}\left(1 - \frac{1}{e}\right) f(X_{MST}), \tag{13}$$

and

$$f(X) \geq \frac{1}{2}\left(1 - \frac{1}{e}\right) f(X_{SPT}), \tag{14}$$

where

$$X_{MST} = arg\max_{X}\left\{f(X)|c_{MST}(X) \leq B\left(\frac{1}{K_{c_{MST}}}\right)\right\}, \tag{15}$$

and

$$X_{SPT} = arg\max_{X}\{f(X)|c_{SPT}(X) \leq B\}, \tag{16}$$

respectively.

Proof. The MST and SPT cases are as follows:

(MST) The theoretical guarantees of GCB-MST is as follows:

By Theorem 5.4, if the subgoal condition is satisfied, $\kappa_c = 1$. Substituting $\kappa_c = 1$ into Eq. (5), the budget constraint is $c(X) \leq B(\frac{1}{K_{c_{MST}}})$

$\therefore X_{MST} = arg\max_X\{f(X)|c_{MST}(X) \leq B(\frac{1}{K_{c_{MST}}})\}$

(SPT) The theoretical guarantees of GCB-SPT are as follows:

By Theorem 5.3, the κ_c of SPT is 0. Substituting $\kappa_c = 0$ into Eq. (5), the budget constraint is $c(X) \leq B$)

$\therefore X_{SPT} = arg\max_X\{f(X)|c_{SPT}(X) \leq B\}$ □

Theorem 5.6 (The theoretical guarantees of GCB-CBST). *Given a submodular monotone set function f, a CBST cost function $\bar{c}$ with $\alpha = 0$ and Eqs. (10) and (11), and a CBST cost function $\check{c}$ with $\alpha = 1$, the GCB approach is to maximize f subject to the budget B in the case of IPP on the terrain. The*

https://doi.org/10.1017/S0263574725102956 Published online by Cambridge University Press

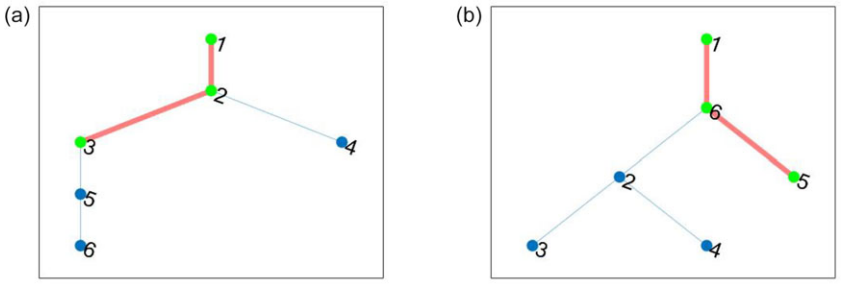


Figure 12. *Example of difference between GCB-MST and GCB-CBST.*

performances of a set X achieve

$$f(X) \geq \frac{1}{2}\left(1 - \frac{1}{e}\right) f(\overline{X}), \tag{17}$$

and

$$f(X) \geq \frac{1}{2}\left(1 - \frac{1}{e}\right) f(\breve{X}), \tag{18}$$

where

$$\overline{X} = arg \max_{X} \left\{ f(X) | \overline{c}(X) \leq B\left(\frac{1}{K_{\overline{c}}}\right) \right\}. \tag{19}$$

and

$$\breve{X} = arg \max_{X} \{ f(X) | \breve{c}(X) \leq B \}. \tag{20}$$

respectively.

Proof. The numerator of the CBST objective function is fixed (see Eq. (9)). When $\alpha = 0$, denominator of the CBST objective function is the same as that of the MST; when $\alpha = 1$, denominator of the CBST objective function is the same as that of the SPT. Hence, the theoretical guarantee is the same as Eqs. (15) and (16). □

Although the theoretical guarantees of CBST are the same as MST, the performance of CBST is different from that of MST due to objective function of the tree spanning. The problem is with budget constraints. The CBST spans the tree adopts cost-benefit objective, so the agent will go through the subgoal with more cost-benefit ratio than that of MST. For example, the budget is 3 and the budget constraint is routing in Figure 1(a) , the path in GCB-MST is $\{1, 2, 3\}$ and the path in GCB-CBST is $\{1, 6, 5\}$ in Figure 12(a) and (b). The accumulated EPD in GCB-MST is 0.48 and that in GCB-CBST is 0.95.

6. Experiments

To evaluate the performance of different approaches, there are three experiments: the maximization extended probability of detection problem (EX1), the search problem (EX2), and Tuning α parameters (EX3). The EX1 is to evaluate the extended probability of target detection (EPD) of different offline approaches with routing constraints in simulations as Eq. (8). The EX2 is to evaluate the search time of different approaches with routing constraints in simulations and real world. The purpose of EX3 is to evaluate the influence of changing α for search results.

https://doi.org/10.1017/S0263574725102956 Published online by Cambridge University Press

Table II. *Drone hardware setup in simulation.*

Parameter	description	value
c_r	range	3.0 (m)
FOV_h	horizontal FOV	60°
FOV_v	vertical FOV	45°
c_{fr}	frame rate	10 (FPS)
h_{max}	flight minimum altitude	0.5 (m)
h_{min}	flight maximum altitude	3.0 (m)
h_{level}	flight altitude levels	0.3 (m)
ω_{max}	limitation of angular velocity	42 (deg/s)
v_{max}	limitation of linear velocity	1 (m/s)

6.1. Experiment setup

6.1.1. Simulation setup

In the EX1 and EX2, there are two different-sized maps. The size of map 1 and map 2 is $4 \times 4 \times 3$ (m^3) and $8 \times 8 \times 3$ (m^3), respectively. To further evaluate the scalability, the size of map 3 is $100 \times 100 \times 3$ (m^3). As shown in Figure 13, the numbers of ground set $|S|$are 58, 58, and 714 in maps 1, 2, and 3, respectively. The number of subgoals in low altitude is more than that in high altitude, due to the camera's field of view (FOV). The index 1 is start point. The experimental setup (e.g., camera and drone parameters) is as shown in Table II. The CPU is Intel(R) i7-8700K CPU with 3.70 GHz and 64 GB RAM.

The cost constraints are 10 (m) and 40 (m), respectively. The cost function c is measured by Euclidean distance. Note that, since computing routing cost is NP-hard problem, c is approximated by $\hat{c}$ via Nearest Neighbor (NN) algorithm [54], which provides a $\frac{3}{2}-$approximation factor.

The chosen parameters in Table II are as follows: The sensing range and FOV are close to Intel Realsense cameras. The frame rate is 10Hz since the RGBD cameras work in $10 \sim 30$Hz. The maximal and minimum flight altitudes are based on the image quality. The higher or lower flight altitude cannot detect targets accurately. Since the fast motion leads to the images blurring, the maximal linear and angular velocities are set.

In single target scenario, the target is located in 121 places uniformly. The drone searches 5 times in each place. Hence, there are 605 times. The search processing is as follows: The agent takes off and moves to the next subgoal. If the drone cannot find the target ($P(Z = 1) \leq 95\%$), the agent will go through the next subgoal until finding the target ($P(Z = 1) \geq 95\%$) or utilizing all budgets.

In multiple-targets scenario, the targets are located in 20 places randomly. The drone searches 5 times in each place. Hence, there are 100 times. The search processing is as follows: The agent takes off and moves to the next subgoal. If the drone cannot find all the targets ($P(Z = 1) \leq 95\%$), the agent will go through the subgoal until finding all the targets ($P(Z = 1) \geq 95\%$) or utilizing all budgets.

There are two setups in the search experiments, online and offline. The online approach for the drone is to compute the information to decide the next subgoal(s); the offline approach for the drone is to compute all subgoals before taking off. There are two steps in the offline setup. First, the path is generated by algorithm subject to budget constraints offline. Second, the agent flies through the path. There are three steps in the online setup. First, the path (or the point) is generated by algorithm online. Second, the agent flies through the path. Third, the agent repeats first step via new measurements by second step until the terminal condition.

To evaluate the search performance, the expected time till positive decisions ($E[TTD^+]$) is adopted [22]. $E[TTD^+]$ represents the expected time of successful searches, while $E[TTD]$ represents the expected time of successful and failed searches.

Along with the path, the agent updates the probability of target detection (Z) in grid map via Bayesian filter. There are 20×20 and 40×40 grids in maps 1 and 2, respectively. When the probability of the

https://doi.org/10.1017/S0263574725102956 Published online by Cambridge University Press

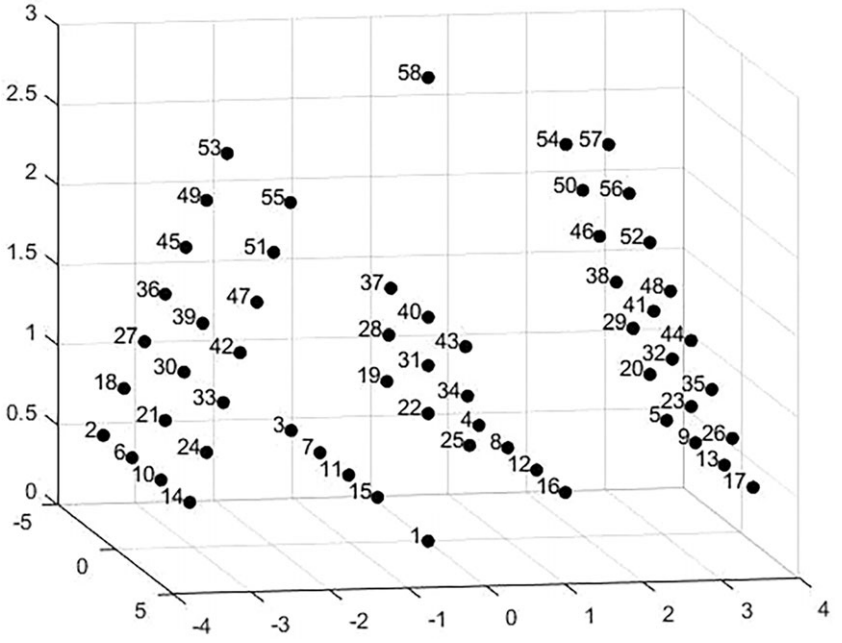
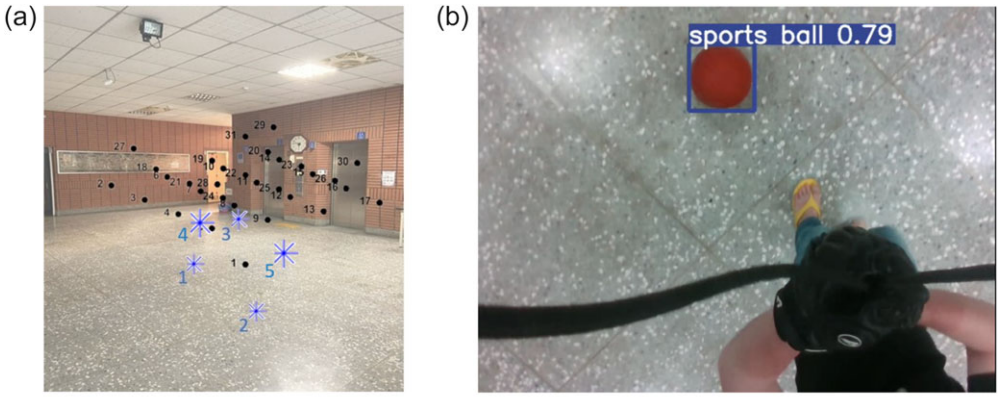


Figure 13. *Ground set in simulation.*



The lobby environment. The detection outcome of YOLOv5.

Figure 14. *(a) The black points, black decimal numbers, the index 1 in black represent the subgoals, the subgoal indexes, and the source node, respectively. The blue stars and decimal numbers represent the target locations and the subgoal indexes, respectively. (b) The d435i provides the RGB images. The YOLOv5 detects the target in the image. The blue box represents the target location in the image with the detection probability.*

cell in grid map is greater than 95% ($P(Z = 1) \geq 95\%$), the agent finds the target. Once the agent finds the target, it will land, and the search time is from taking off to landing.

6.1.2. Real world setup

The map in the real world is a $4 \times 4 \times 2$ (m^3) lobby at the Mathematics Department of National Central University (see Figure 14 (a)). The target is located in 5 places randomly. The drone searches 5 times in each place. Hence, there are 125 times. The budget $B = 10$ (m) is adopted in this map. Figure 14 (a) shows the ground set $|S| = 31$ and the fixed start point is the subgoal 1. The target is located in three positions (blue stars) at each search.

https://doi.org/10.1017/S0263574725102956 Published online by Cambridge University Press

Table III. *Drone hardware setup and specification in real world.*

Parameter	description	value
c_r	range	3.0 (m)
FOV_h	horizontal FOV	69°
FOV_v	vertical FOV	42°
c_{fr}	frame rate	30 (FPS)
h_{max}	flight minimum altitude	1.1 (m)
h_{min}	flight maximum altitude	2.0 (m)
h_{level}	flight altitude levels	0.3 (m)
ω_{max}	limitation of angular velocity	120 (deg/s)
v_{max}	limitation of linear velocity	0.2 (m/s)
	Maximum Flight Time (hovering, no payload)	12 (min)
	Maximum Takeoff Weight	1.5 (kg)
	Flight Speed	10–20 (km/h), max 35 (km/h)

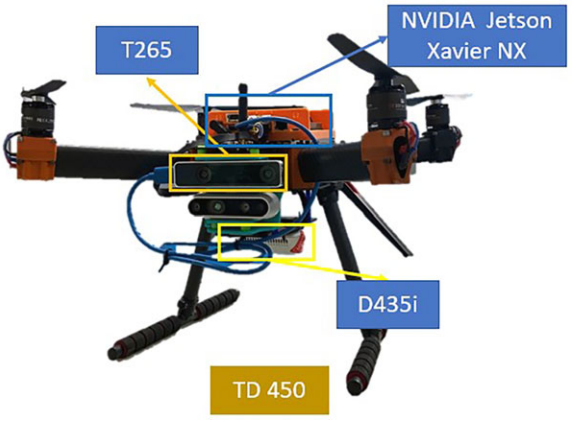


Figure 15. *The TD100-EDU450, sensors, and development board.*

As shown in Figure 15, the TD100-EDU450 is developed by Taiwan Drone 100 corporation. Its specification is shown in Table III.

The D435i RGBD camera is used to explore the environment, while T265 camera is used to localize itself. The YOLOv5 [55] is adopted to detect objects running on Nvidia Jetson Xavier NX (see Figure 14(b)). The drone's operating system is Ubuntu 18.04 and runs the Robot Operating System. The parameters of TD100-EDU450 hardware setup are shown in Table III.

6.2. EX1: EPD maximization

In the experiment, the GCB-CBST is compared with four methods: A-optimality with CMA-ES [4], GCB [10], GCB-MST [14], and GCB-SPT. CMA-ES is online method, while the others are offline methods.

The experiment results in the Gazebo simulation are summarized in Table IV. Since CMA-ES is online algorithm, the accumulated EPD is not available. GCB-CBST outperforms other approaches in the accumulated EPD in maps 1 and 2. The tree structure of SPT is inefficient due to the repeated paths. Since the path in GCB-SPT has high cost-benefit, the accumulated EPD of GCB-SPT is higher than GCB-MST. These experiments show that GCB-CBST outperforms the other three methods in the maximal EPD problem.

https://doi.org/10.1017/S0263574725102956 Published online by Cambridge University Press

Table IV. *EPD maximization experiment results. The accumulated extended probability of detection (AEPD) as follows.*

Method	AEPD (map1)	AEPD (map2)
A-opt with CMA-ES	NA	NA
GCB	0.90	0.80
GCB-CBST	**0.99**	**0.84**
GCB-MST	0.489	0.47
GCB-SPT	0.843	0.52

6.3. EX2: Search experiment

The goal of the drone is to find the target as soon as possible subject to budget constraint, so the metrics in this experiment are expected time till positive decisions ($\mathbb{E}[\text{TTD}^+]$) and success rate.

In the multiple targets scenario, the metrics in this experiment are expected time till positive decisions with multiple targets ($\mathbb{E}[\text{TTD}_m^+]$) and the average number of detected objects. The $\text{E}[\text{TTD}_m^+]$ is defined as follows:

$$E[TTD_m^+] = \frac{1}{Nn} \sum_{j=1}^{N} \sum_{i=1}^{n} t_i^j,$$

where N, n, and t_i^j represent the number of trials, number of objects in trial $j \in N$, and detection time of object $i \in n$ in trial $j \in N$. However, the search time of non-detected objects will be set to the maximal time constraint if it reaches the time constraint. The parameters are same as EX1.

6.3.1. Simulation

In single target scenario, the expected time till positive decision ($\mathbb{E}[\text{TTD}^+]$) is compared to evaluate the search performance. GCB-CBST outperforms other four methods in $\mathbb{E}[\text{TTD}^+]$. GCB-CBST and GCB-SPT are competitive in map 1 since the drone moves to high cost-benefit subgoals. However, the success rate in GCB-SPT is less than that in GCB-CBST since the path in GCB-SPT is inefficient.

The success rates of the GCB and GCB-CBST are competitive in map 2 (see Table V). The success rate of the GCB is higher than that of the GCB-CBST in map 1 (see Table V). In the GCB-CBST path, the drone moves to the repeated subgoals. Hence, it has fewer false detections than GCB does. GCB-CBST attains a little less success rate than GCB in map1 because the drone moves to the repeated subgoals in GCB-CBST.

In multiple targets scenario, the average number of detected targets of the GCB and GCB-CBST is the same (see Table VII). GCB and GCB-CBST outperform the other three methods in the average number of detected targets. The $\text{E}[\text{TTD}_m^+]$ of the GCB and GCB-CBST is competitive (see Table VII) since the targets' locations are random. GCB and GCB-CBST outperform the other three methods in $\text{E}[\text{TTD}_m^+]$.

In the single target and multiple targets scenario of large-scale simulations (map 3), the success rates and $\text{E}[\text{TTD}^+]$ of GCB-CBST outperform other approaches (see Table VI for single target and Table VIII for multiple targets). The computational time, CPU usage, and memory usage of GCB-CBST, GCB-MST, and GCB-SPT are very close.

In summary, GCB-CBST is the best method for search tasks, since it can move through the high cost-benefit subgoals immediately and repeatedly.

6.3.2. Real world

The tree structures of GCB-CBST, GCB-MST, and GCB-SPT are shown in Figure 16. GCB-CBST outperforms the other four methods in $\mathbb{E}[\text{TTD}^+]$ (see Table IX). The height in real world is lower than that in the simulation, so the path in GCB-SPT does not have high coverage. The $\mathbb{E}[\text{TTD}^+]$ in GCB-SPT is the worst.

https://doi.org/10.1017/S0263574725102956 Published online by Cambridge University Press

Table V. *Search experiment results reveal indicators success rate and $\mathbb{E}[TTD^+]$ in map 1 and map 2.*

Method	Map1		Map2	
	$\mathbb{E}[TTD^+]$	success rate	$\mathbb{E}[TTD^+]$	success rate
CMA-ES	12.6	0.77	85.7	0.28
GCB	12.6	**0.96**	30.4	**0.73**
GCB-CBST	**5.9**	0.8	**26.9**	0.705
GCB-MST	18.4	0.44	48.3	0.30
GCB-SPT	6.0	0.75	30.5	0.43

Table VI. *Search experiment results reveal indicators success rate and $\mathbb{E}[TTD^+]$ in map 3.*

Method	$\mathbb{E}[TTD^+]$	success rate	computational time (sec.)	CPU usage	Memory usage
CMA-ES	585.4	20%	453.3	35%	7%
GCB	533.2	27%	413.5	29%	6%
GCB-CBST	**509.8**	**33%**	382.4	25%	5%
GCB-MST	548.6	29%	374.2	24%	5%
GCB-SPT	590.2	14%	375.5	26%	5%

Table VII. *Search experiment results reveal indicators the average number of detected objects and $\mathbb{E}[TTD_m^+]$ in simulation.*

Map1	$\mathbb{E}[TTD_m^+]$	average number
CMA-ES	123.61	1.4
GCB	**10.87**	**2**
GCB-CBST	13.24	**2**
GCB-MST	451.96	1.1
GCB-SPT	237.63	1.5

Table VIII. *Search experiment results reveal indicators the average number of detected objects and $\mathbb{E}[TTD_m^+]$ in map 3.*

Method	$\mathbb{E}[TTD^+]$	average number	computational time (sec.)	CPU usage	Memory usage
CMA-ES	563.2	0.32	462.5	38%	7%
GCB	503.4	0.39	405.6	31%	7%
GCB-CBST	**453.1**	**0.41**	376.7	26%	5%
GCB-MST	589.4	0.28	376.2	25%	5%
GCB-SPT	593.2	0.24	373.7	25%	6%

In single target scenario, the GCB-CBST outperforms the other four methods in success rate (see Table IX, since the GCB-CBST must go to subgoals via tree structure, and the drone moves to the repeated subgoals which have high cost-benefit values. The GCB-MST and GCB-SPT are the worst in success rate.

In multiple targets scenario, the GCB-CBST outperforms the other four methods in both $\mathbb{E}[TTD_m^+]$ and average number (see Table X). In the GCB-MST method, the drone traverses unnecessary subgoals, resulting in the worst performance.

These experiments show that GCB-CBST outperforms the other four benchmarks in search problems.

https://doi.org/10.1017/S0263574725102956 Published online by Cambridge University Press

Table IX. *Search experiment results reveal indicators success rate and $\mathbb{E}[TTD^{+}]$ in real world.*

	$\mathbb{E}[\mathbf{TTD}^{+}]$	**success rate**
CMA-ES	86.91	0.88
GCB	69.23	0.88
GCB-CBST	**51.92**	**1.00**
GCB-MST	68.77	0.52
GCB-SPT	79.59	0.68

Table X. *Search experiment results reveal indicators the average number of detected objects and $\mathbb{E}[TTD_m^{+}]$ in real world.*

	$\mathbb{E}[\mathbf{TTD}_m^{+}]$	**average number**
CMA-ES	91.98	1.9
GCB	61.82	**2**
GCB-CBST	**37.0**	**2**
GCB-MST	153.28	1.36
GCB-SPT	138.32	1.54

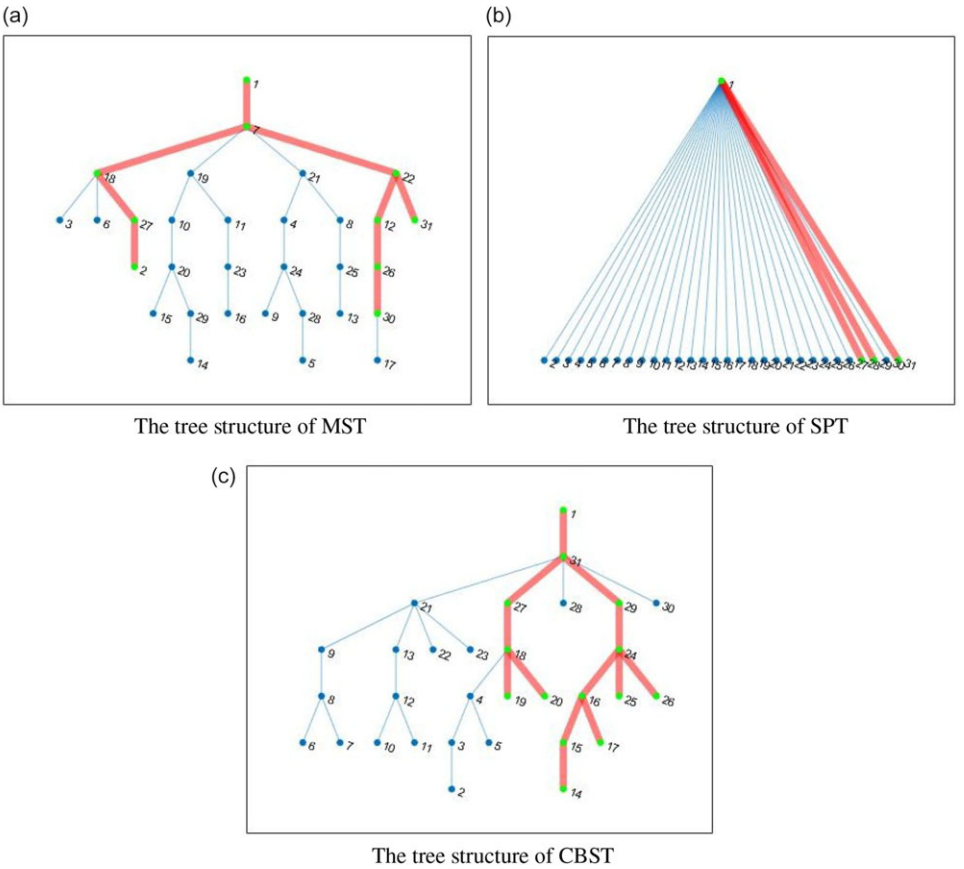


Figure 16. *Three tree structures in the lobby environment. The blue circles and green circles represent the ground set and the picked subgoals, respectively. The red lines represent the paths.*

https://doi.org/10.1017/S0263574725102956 Published online by Cambridge University Press

Table XI. *Search experiment results reveal indicators success rate and $\mathbb{E}[TTD^+]$ in map 1 with different α.*

α	E[TTD$^+$]	Success rate
$0 \sim 0.15$	7.85	**0.74**
$0.2 \sim 0.25$	**5.76**	0.60
$0.3 \sim 0.4$	6.48	0.63
$0.45 \sim 1$	6.58	0.64

Table XII. *Search experiment results reveal indicators success rate and $\mathbb{E}[TTD^+]$ in map 2 with different α.*

α	E[TTD$^+$]	Success rate
0	**26.9**	0.70
0.05	38.8	**0.83**
$0.1 \sim 0.15$	38.5	**0.83**
0.2	34.1	0.62
$0.25 \sim 0.5$	38	0.50
$0.55 \sim 1$	30.5	0.43

6.4. EX3: α tuning

The α ranges from 0 to 1 with an interval of 0.05. In the map1, the alpha values generate 4 kinds of tree structures. the alpha ranges are $0 \sim 0.15$, $0.2 \sim 0.25$, $0.3 \sim 0.4$, and $0.4 \sim 1$. In the map2, the alpha values generate 6 kinds of tree structures. The alpha ranges are 0, 0.05, $0.1 \sim 0.15$, 0.2, $0.25 \sim 0.5$, and $0.55 \sim 1$.

As shown in Table XI, since map 1 is not big enough, the results in different tree structures in GCB-CBST show insignificant differences. As shown in Table XII, there is significant difference between $\alpha = 0$ and $\alpha = 1$ since the map 2 ($8 \times 8 \times 3(m^3)$) is bigger than map 1 ($4 \times 4 \times 3(m^3)$). The cost function with parameter $\alpha = 0$ is the best in $\mathbb{E}[\text{TTD}^+]$, because the tree structure is minimum weight spanning tree. In the MST algorithm, it is possible to generate edges with the minimum total weight, which results in shorter distances between subgoals. Consequently, a better $\mathbb{E}[\text{TTD}^+]$ can be achieved.

In summary, the performance when α approaches 0 is better than the performance when α approaches 1 due to the path efficiency. How to tune an optimal α is still an issue. The $\alpha = 0$ could be a fast solution in search experiments.

7. Discussions

7.1. Guidance and search problems

This research focuses on the algorithms of search problems instead of guidance problems. Guidance and search problems differ in three key aspects [51, 56]: knowledge of locations, problem formulation, and computational complexity. In guidance problems, the robot knows both the starting position (A) and the destination (B) and needs to find a path between them. In contrast, search problems involve finding an unknown target (C) without prior knowledge of its location. Additionally, guidance problems are usually continuous, meaning the robot moves through a continuous space, while search problems are often discrete, where there are distinct subgoals in the environment. Finally, when search problems are transformed into set-cover problems, they become NP-complete, making them computationally challenging to solve optimally.

https://doi.org/10.1017/S0263574725102956 Published online by Cambridge University Press

The key distinction between guidance and search problems lies in the robot's knowledge of locations. In guidance problems, the robot knows exactly where it starts (A) and where it needs to go (B), so the challenge is to determine an efficient path between the two points. In contrast, search problems involve an unknown target (C), meaning the robot must explore the environment to locate it. Since the robot does not have prior knowledge of C's location, it must rely on strategies such as systematic scanning, probabilistic search, or heuristic exploration to find the target. This lack of information makes search problems inherently more uncertain and complex than guidance problems.

Another major difference is the problem formulation. Guidance problems are typically continuous, where the robot moves through an environment with smooth trajectories and can adjust its path dynamically. These problems are often solved using optimization techniques, control theory, or path-planning algorithms like A* or Dijkstra's algorithm. In contrast, search problems are usually discrete, meaning the environment is covered by a set of discrete subgoals, and the robot moves step-by-step between them. The search process involves selecting the next subgoal based on prior observations, often modeled as a decision-making process in a discrete space.

Finally, guidance and search problems differ in computational complexity. While guidance problems can often be solved efficiently with deterministic algorithms, search problems become significantly harder when transformed into set-cover problems. In this formulation, the goal is to cover all possible target locations with a minimal number of sensing or search actions. This is an NP-complete problem, meaning there is no known polynomial-time algorithm to find the optimal solution. As a result, heuristic and approximation techniques, such as greedy algorithms, genetic algorithms, or probabilistic sampling methods, are commonly used to achieve near-optimal solutions within a reasonable computation time.

7.2. Potential limitations of CBST

Although CBST works well in experimental scenarios, there are some limitations of CBST. First, one of the assumptions for CBST is that the target is static. If the target is dynamic, a Bayes filter should be added to estimate the target distribution. Second, there are no obstacles between two nodes. Hence, the path cost of spanning tree is computed based on the straight line between two nodes. If there are obstacles between two nodes, path cost needs to be further estimated (e.g., receding horizon). Third, the tree structure of CBST is fixed during the search mission. If the robot tunes α online, it can expand different tree structures. However, in such cases, the theoretical guarantees are different.

8. Conclusion and future work

This research proposes the CBST tree structure to solve submodular maximization problems subject to routing constraints for terrain monitor problems. The research is summarized as follows: First, this is the first work to explore CBST for this problem. Second, to the best of our knowledge, this is the first work to prove the relationship between the theoretical guarantees and the total curvature (κ_c) of tree structures. Third, the experiments show that the proposed CBST using GCB algorithm outperforms the benchmark approaches in probability of detection, search time, and success rate for both single target and multiple targets.

The future work of this research is as follows: First, the proposed CBST outperforms the benchmark in search experiments in the empty map. If there are obstacles in the map, the routing path cost needs to be further modified. Second, the GCB with tree structured graph cost function is offline approach. How to apply adaptive submodularity to tree structures is another research direction. Third, tuning alpha is going to be very map-specific. How to estimate the best alpha is a research topic. Fourth, how to apply proposed algorithms for search and rescue is an useful direction [57]. When the targets are humans, there are more uncertainties to be considered (e.g., different shapes and colors). Finally, the proposed algorithm is applied to a static target. How to extend the proposed algorithm to multiple moving targets is another potential research.

Acknowledgement. The authors would like to thank Yan-Shuo Li for his help with the simulations during the revision.

https://doi.org/10.1017/S0263574725102956 Published online by Cambridge University Press

Competing interests. No conflicts of interest exist

Ethical standard. Not applicable under the heading.

Authors' contributions. Conceptualization, Wei-Hsiang Chiu, and Kuo-Shih Tseng; methodology, Wei-Hsiang Chiu and Kuo-Shih Tseng; software, Wei-Hsiang Chiu and Kuo-Shih Tseng; validation, Wei-Hsiang Chiu and Kuo-Shih Tseng; formal analysis, Wei-Hsiang Chiu and Kuo-Shih Tseng; investigation, Wei-Hsiang Chiu and Kuo-Shih Tseng; resources, Wei-Hsiang Chiu and Kuo-Shih Tseng; data collection, Wei-Hsiang Chiu; writing–original draft preparation, Wei-Hsiang Chiu; writing–review and editing, Kuo-Shih Tseng; visualization, Wei-Hsiang Chiu; supervision, Kuo-Shih Tseng; project administration, Kuo-Shih Tseng; funding acquisition, Kuo-Shih Tseng.

Financial support. This research was completed thanks to the financial support from Taiwan NSTC 113-2221-E-008-101-MY2.

Use of AI. ChatGPT 4o is applied to revise some paragraphs of this manuscript. ChatGPT (GPT-4o, accessed October 2025) was employed to refine the manuscript's English expression and improve readability. The tool was not used for data generation, analysis, or idea development, and all substantive content and conclusions were produced by the authors.

Supplementary material. The supplementary material for this article can be found at https://doi.org/10.1017/S0263574725102956

References

[1] A. Krause, C. Guestrin, A. Gupta and J. Kleinberg, "Near-Optimal Sensor Placements: Maximizing Information While Minimizing Communication Cost," **In:** *International Conference on Information Processing in Sensor Networks,* (IEEE, 2006) pp. 2–10.
[2] A. Singh, A. Krause and W. J. Kaiser, "Nonmyopic Adaptive Informative Path Planning for Multiple Robots," **In:** *International Joint Conference on Artificial Intelligence*, (ACM, 2009) pp. 1843–1850.
[3] H.-C. Chang and K.-S. Tseng, "Localizing Complex Terrains Through Adaptive Submodularity," **In:** *International Symposium on Safety, Security, and Rescue Robotics,* (IEEE, 2022) pp. 138–144.
[4] M. Popović, T. Vidal-Calleja, G. Hitz, J. J. Chung, I. Sa, R. Siegwart and J. Nieto, "An informative path planning framework for UAV-based terrain monitoring," *Auton. Robot.* **44**(6), 889–911 (2020).
[5] G. L. Nemhauser, L. A. Wolsey and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions—I," *Math. Program.* **14**(1), 265–294 (1978).
[6] T. Grossman and A. Wool, "Computational experience with approximation algorithms for the set covering problem," *Eur. J. Oper. Res.* **101**(1), 81–92 (1997).
[7] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Oper. Res.* **21**(2), 498–516 (1973).
[8] J. Binney and G. S. Sukhatme, "Branch and Bound for Informative Path Planning," **In:** *International Conference on Robotics and Automation*, (IEEE, 2012) pp. 2147–2154.
[9] M. Popović, T. Vidal-Calleja, G. Hitz, I. Sa, R. Siegwart and J. Nieto, "Multiresolution Mapping and Informative Path Planning for UAV-based Terrain Monitoring," **In:** *International Conference on Intelligent Robots and Systems (IROS)*, (IEEE/RSJ, 2017) pp. 1382–1388.
[10] H. Zhang and Y. Vorobeychik, "Submodular Optimization with Routing Constraints," **In:** *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. **30**(1) (AAAI, 2016) pp. 819–825.
[11] M. M. Flood, "The traveling-salesman problem," *Oper. Res.* **4**(1), 61–75 (1956).
[12] S. Khuller, A. Moss and J. Naor, "The budgeted maximum coverage problem," *Inform. Process. Lett.* **70**(1), 39–45 (1999).
[13] U. Feige, "A threshold of ln n for approximating set cover," *J. ACM* **45**(4), 634–652 (1998).
[14] P.-T. Lin and K.-S. Tseng, "Improvement of submodular maximization problems with routing constraints via submodularity and fourier sparsity," *IEEE Robot. Autom. Lett.* **8**(4), 1927–1934 (2023).
[15] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki and S. Thrun, *Principles of Robot Motion* (MIT, Cambridge, Massachusetts, 2005).
[16] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robot. Auton. Syst.* **61**(12), 1258–1276 (2013).
[17] T. H. Chung, G. A. Hollinger and V. Isler, "Search and pursuit-evasion in mobile robotics," *Auton. Robot.* **31**, 299–316 (2011).
[18] L. D. Stone, *Theory of Optimal Search* (Elsevier, 1976).
[19] F. Bourgault, T. Furukawa and H. F. Durrant-Whyte, "Coordinated decentralized search for a lost target in a bayesian world," **In:** *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1 (2003) pp. 48–53.
[20] K. E. Trummel and J. R. Weisinger, "The complexity of the optimal searcher path problem," *Oper. Res.* **34**(2), 324–327 (1986).

https://doi.org/10.1017/S0263574725102956 Published online by Cambridge University Press

[21] T. H. Chung and J. W. Burdick, "A Decision-Making Framework for Control Strategies in Probabilistic Search," **In:** *IEEE International Conference on Robotics and Automation* (2007) pp. 4386–4393.

[22] T. H. Chung and J. W. Burdick, "Analysis of search decision making using probabilistic search strategies," *IEEE Trans. Robot.* **28**(1), 132–144 (IEEE, 2011).

[23] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer* **22**(6), 46–57 (1989).

[24] K.-S. Tseng and B. Mettler, "Near-optimal probabilistic search via submodularity and sparse regression," *Auton. Robot.* **41**(1), 205–229 (2017).

[25] H. Lau, S. Huang and G. Dissanayake, "Discounted MEAN bound for the optimal searcher path problem with non-uniform travel times," *Eur. J. Oper. Res.* **190**(2), 383–397 (2008).

[26] E. Besada-Portas, L. de la Torre, J. A. M. de la Cruz and B. de Andrés-Toro, "Evolutionary trajectory planner for multiple uavs in realistic scenarios," *IEEE Trans. Robot.* **26**(4), 619–634 (2010).

[27] M. Torres, D. A. Pelta, J.é L. Verdegay and J. C. Torres, "Coverage path planning with unmanned aerial vehicles for 3D terrain reconstruction," *Expert Syst. Appl.* **55**, 441–451 (2016).

[28] C. Qian, J.-C. Shi, Y. Yu and K. Tang, "On Subset Selection with General Cost Constraints," **In:** *Proceedings of the Twenty-Sixth International Joint International Joint Conference on Artificial Intelligence*, vol. 17 (2017) pp. 2613–2619.

[29] C. Bian, C. Feng, C. Qian and Y. Yu, "An Efficient Evolutionary Algorithm for Subset Selection with General Cost Constraints," **In:** *AAAI Conference on Artificial Intelligence*, vol. **34**(04) (2020) pp. 3267–3274.

[30] I. K. Nikolos, K. P. Valavanis, N. C. Tsourveloudis and A. N. Kostaras, "Evolutionary algorithm based offline/online path planner for uav navigation," *IEEE Trans. Syst. Man. Cybern. Part B (Cybern.)* **33**(6), 898–912 (2003).

[31] G. Hitz, E. Galceran, M. È. ve Garneau, F. Pomerleau and R. Siegwart, "Adaptive continuous-space informative path planning for online environmental monitoring," *J. Field. Robot.* **34**(8), 1427–1449 (2017).

[32] S. Javdani, M. Klingensmith, J. A. Bagnell, N. S. Pollard and S. S. Srinivasa, "Efficient Touch Based Localization Through Submodularity," **In:** *IEEE International Conference on Robotics and Automation* (2013) pp. 1828–1835.

[33] Y. Yu, H. Zheng and W. Xu, "Learning and sampling-based informative path planning for auvs in ocean current fields," *IEEE Trans. Syst. Man. Cybern: Syst.* **55**(1), 51–62 (2024).

[34] M. Popovic, G. Hitz, J. Nieto, I. Sa, R. Siegwart and E. Galceran, "Online Informative Path Planning for Active Classification Using UAVs," **In:** *IEEE International Conference on Robotics and Automation* (IEEE, 2017) pp. 5753–5758.

[35] B. Yamauchi, "Frontier-Based Exploration Using Multiple Robots," **In:** *International Conference on Autonomous Agents* (1998) pp. 47–53.

[36] N. Hansen, "The CMA Evolution Strategy: A Comparing Review," **In:** *Towards a New Evolutionary Computation*, (Springer, 2006) pp. 75–102 .

[37] R. Cui, Y. Li and W. Yan, "Mutual information-based multi-AUV path planning for scalar field sampling using multidimensional RRT," *IEEE Trans. Syst. Man. Cybern.: Syst.* **46**(7), 993–1004 (IEEE, 2015).

[38] C. Wang, J. Cheng, W. Chi, T. Yan and M. Q.-H. Meng, "Semantic-aware informative path planning for efficient object search using mobile robot," *IEEE Trans. Syst. Man. Cybern.: Syst.* **51**(8), 5230–5243 (IEEE, 2021).

[39] A. Vashisth, J. Rückin, F. Magistri, C. Stachniss and M. Popovi'c, "Deep reinforcement learning with dynamic graphs for adaptive informative path planning," *IEEE Robot. Autom. Lett.* **9**(9), 7747–7754 (2024).

[40] Y. Zhang, G. Tian, X. Shao, M. Zhang and S. Liu, "Semantic grounding for long-term autonomy of mobile robots toward dynamic object search in home environments," *IEEE Trans. Ind. Electron.* **70**(2), 1655–1665 (2023).

[41] M. Corah and N. Michael, "Distributed matroid-constrained submodular maximization for multi-robot exploration: Theory and practice," *Auton. Robot.* **43**, 485–501 (2019).

[42] B.-X. Lu and K.-S. Tseng, "3d Map Exploration Via Learning Submodular Functions in the Fourier Domain," **In:** *International Conference on Unmanned Aircraft Systems (ICUAS)* (2020) pp. 1199–1205.

[43] G. Hollinger and S. Singh, "Proofs and experiments in scalable, near-optimal search by multiple robots," *Robot.: Sci. Syst.* **1** (2008).

[44] P. Stobbe and A. Krause, "Learning Fourier Sparse Set Functions," **In:** *Artificial Intelligence and Statistics*, (PMLR, 2012) pp. 1125–1133.

[45] J. Chen, Y. Zhang, L. Wu, T. You and X. Ning, "An adaptive clustering-based algorithm for automatic path planning of heterogeneous UAVs," *IEEE Trans. Intell. Transp. Syst.* **23**(9), 16842–16853 (2022).

[46] C. Chen, Y. Wang, Y. Zhang, Y. Lu, Q. Shu and Y. Hu, "Extrinsic-and-intrinsic reward-based multi-agent reinforcement learning for multi-UAV cooperative target encirclement," *IEEE Trans. Intell. Transp.* **26**(10), 1–13 (2025).

[47] C. J. Alpert, T. C. Hu, J.-H. Huang and A. B. Kahng, "A Direct Combination of the Prim and Dijkstra Constructions for Improved Performance-driven Global Routing," **In:** *IEEE International Symposium on Circuits and Systems* (1993) pp.1869–1872.

[48] C. J. Alpert, W.-K. Chow, K. Han, A. B. Kahng, Z. Li, D. Liu and S. Venkatesh, "Prim-Dijkstra Revisited: Achieving Superior Timing-Driven Routing Trees," **In:** *International Symposium on Physical Design* (2018) pp. 10–17.

[49] M. Conforti and G. Cornuéjols, "Submodular set functions, matroids and the greedy algorithm: Tight worst-case bounds and some generalizations of the Rado-Edmonds theorem," *Discrete Appl. Math.* **7**(3), 251–274 (Elsevier, 1984).

[50] J. B. Kadane and H. A. Simon, "Optimal strategies for a class of constrained sequential problems," *Ann. Stat.* **5**(2), 237–255 (1977).

[51] K.-S. Tseng, Learning in human and robot search: Subgoal, submodularity, and sparsity Ph.D. Thesis (University of Minnesota, 2016).

[52] R.-S. Shieh and K.-S. Tseng, "Map explorations via dynamic tree-structured graph," *IEEE Robot. Autom. Lett.* **9**(6), 5062–5069 (2024).

https://doi.org/10.1017/S0263574725102956 Published online by Cambridge University Press

[53] K. Schütte and B. L. van der Waerden, "Das problem der dreizehn kugeln," *Math. Ann.* **125**(1), 325–334 (Springer, 1952).
[54] D. J. Rosenkrantz, R. E. Stearns and P. M. Lewis II, "An analysis of several heuristics for the traveling salesman problem," *SIAM J. Comput.* **6**(3), 563–581 (1977).
[55] G. Jocher, A. Chaurasia, A. Stoken, J. Borovec, Y. K. NanoCode012, K. Michael, J. F. TaoXie, L. imyhxy, Z. Yifu, C. Wong, V. Abhiram, D. Montes, Z. Wang, C. Fati, J. Nadar, U. K. De Laughing and V. Sonck, tkianai, yxNONG, Piotr Skalski, Adam Hogan, Dhruv Nair, Max Strobel, and Mrinal Jain. ultralytics/yolov5:v7.0 - YOLOv5 SOTA Realtime Instance Segmentation, November 2022, (2022).
[56] R. Chai, A. Tsourdos, A. Savvaris, S. Chai, Y. Xia and C. L. P. Chen, "Review of Advanced Guidance and Control Methods," **In:** *Advanced Trajectory Optimization, Guidance and Control Strategies for Aerospace Vehicles* (Springer, 2023).
[57] B.-X. Lu, Y.-C. Tsai and K.-S. Tseng, "People search via deep compressed sensing techniques," *Robotica* **40**(7), 2320–2348 (Cambridge University Press, 2022).

Cite this article: W. Chiu and K. Tseng (2025). "Informative path planning for unmanned aerial vehicles using cost-benefit spanning tree", Robotica **43**, 4402–4428. https://doi.org/10.1017/S0263574725102956

https://doi.org/10.1017/S0263574725102956 Published online by Cambridge University Press