Autonomous Robots (2024) 48:3
https://doi.org/10.1007/s10514-024-10156-6

Maximal coverage problems with routing constraints using cross-entropy Monte Carlo tree search

Pao-Te Lin[1] · **Kuo-Shih Tseng**[1]

Received: 24 June 2022 / Accepted: 9 January 2024 / Published online: 30 January 2024

Abstract
Spatial search, and environmental monitoring are key technologies in robotics. These problems can be reformulated as maximal coverage problems with routing constraints, which are NP-hard problems. The generalized cost-benefit algorithm (GCB) can solve these problems with theoretical guarantees. To achieve better performance, evolutionary algorithms (EA) boost its performance via more samples. However, it is hard to know the terminal conditions of EA to outperform GCB. To solve these problems with theoretical guarantees and terminal conditions, in this research, the cross-entropy based Monte Carlo Tree Search algorithm (CE-MCTS) is proposed. It consists of three parts: the EA for sampling the branches, the upper confidence bound policy for selections, and the estimation of distribution algorithm for simulations. The experiments demonstrate that the CE-MCTS outperforms benchmark approaches (e.g., GCB, EAMC) in spatial search problems.

Keywords Submodularity · Cross-entropy method · Monte Carlo tree search · Maximal covearage

1 Introduction

Various robotic problems can be reformulated as maximal coverage problems with routing constraints. For example, an unmanned aerial vehicle (UAV) builds maps within limited fuel constraints (Lu and Tseng, 2020). A robot searches for a fixed target (Tseng and Mettler, 2017, 2018) or a dynamic target (Lanillos et al., 2012) within the limited time. However, maximal coverage problems with routing constraints involve two NP-hard problems. First, finding the maximal coverage of K sets from N sets (Nemhauser et al., 1978) is a set covering problem (Grossman and Wool, 1997). Second, finding the minimal route from K nodes is a traveling salesman problem (TSP) (Lin and Kernighan, 1973). The submodularity is utilized to solve these problems with theoretical guarantees (Nemhauser et al., 1978).

The problem statement of this research is as follows: Given a set of subgoals $S = \{1, 2, 3, \ldots, N\}$, a coverage function (f) and a path function (c), the goal is to find the maximal coverage subject to the path cost within the budget (B). The coverage and path functions are oracle functions, which can be queried by the robot. The specific applications in this research include 2D coverage problems and 3D spatial search problems. In 2D coverage problems, the objective is to cover 2D environment with a 2D laser range finder. In 3D spatial search problems, the objective for the UAV is to find the target in 3D environment as soon as possible via a RGBD camera.

There are two major approaches for these problems. First, the generalized cost-benefit algorithm (GCB) adopts greedy methods to solve these problems with $\frac{1}{2}(1-\frac{1}{e})\widetilde{OPT}$ guarantees (Zhang and Vorobeychik, 2016), where $\widetilde{OPT}$ is the approximation of optimal performance. Second, for non-submodular problems, the evolutionary algorithm (so called EAMC) is to maximize monotone functions under monotone cost constraints (Bian et al., 2020). It can achieve $\frac{\alpha_f}{2}(1-\frac{1}{e^{\alpha_f}})\widetilde{OPT}$ in polynomial time, where α_f is the submodularity ratio of objective functions. $\alpha - submodular$ provides a way to solve these problems even the objective function is not submodular.

The disadvantages of GCB and EAMC are as follows: For GCB approach, although the greedy algorithms give theoretical guarantees, there is a gap between the $\frac{1}{2}(1-\frac{1}{e})\widetilde{OPT}$ and optimum (OPT). For EAMC approach, although it could find solutions via more samples, the initial samples could

✉ Kuo-Shih Tseng
kuoshih@cs.umn.edu

Pao-Te Lin
linbaodeee@gmail.com

[1] Department of Mathematics, National Central University, Taoyuan City, Taiwan

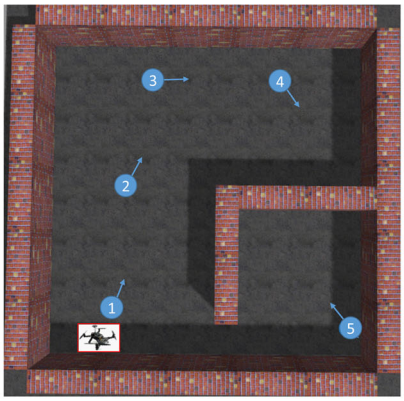
affect the performance. To cope with these issues, maximal coverage problems with routing constraints for N sets can be reformulated as a search tree. The number of bottom nodes is $O(2^N)$. Exploring solutions from the search tree could outperform the GCB and EAMC.

This research proposes CE-MCTS to find better solutions than GCB and EAMC. The first part of CE-MCTS adopts EAMC for sampling the trajectory and utilizes the budget resources via the distribution updating recursively by the cross-entropy method (CEM) (Rubinstein, 1999). The second part of CE-MCTS launches the procedure of MCTS to trade off between exploration and exploitation via upper confidence bound (UCB). The third part of CE-MCTS employs CEM for simulations.

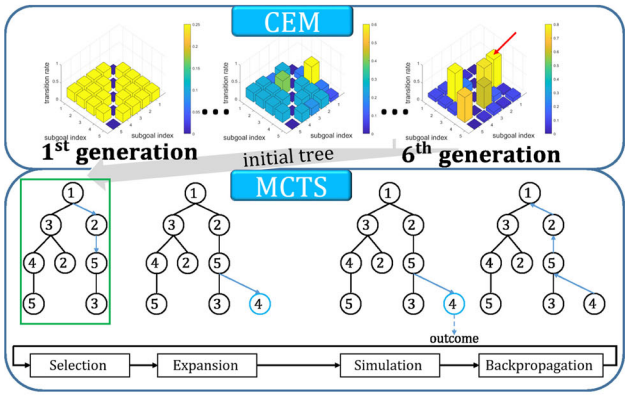
The CE-MCTS is illustrated in Fig. 1. There are 5 predefined subgoals (the ground set) and the robot is located at subgoal #1 (Fig. 1a). The goal of the robot is to find the path with the maximal coverage subject to a routing cost (e.g., less than 2 min). Hence, the number of possible paths is $O(2^4)$. As Fig. 1b shows, the CEM is to sample 3 paths for each generation, and the total generation is 6. Then, the initial tree of MCTS is built by the sampling trajectories (see the green box in Fig. 1b. In the MCTS part of Fig. 1b, the MCTS adopts EA to sample the $1 \rightarrow 2$ branch and selects subgoal #4 in the $Expansion$ stage. It keeps running the four steps until satisfying termination conditions.

The contributions of this research are as follows: First, inspired by AlphaGo (Silver et al., 2016), the CE-MCTS is proposed to solve maximal coverage problems with routing constraints. To the best of our knowledge, this is the first work to adopt CE-MCTS for these problems. Second, the experiments show that the proposed CE-MCTS algorithms outperform the benchmark approaches (e.g., EAMC and GCB). Hence, it can be a trade-off between the performance and computational time. Third, the proposed triangular esitmation (TE) is able to compute the terminal criterion for the sampling-based method (e.g., CE-MCTS) and guarantees the CE-MCTS performance under certain conditions.

The advantage and disadvantage of the predefined subgoals in CE-MCTS are as follows: The advantage is that given a ground set and map, the greedy algorithms can find solutions for maximal coverage problems with theoretical guarantees via submodularity. The disadvantage is that theses subgoals could be unreachable. For example, in map exploration problems, the environment is unknown. The predefined subgoals could be around the obstacles or on the obstacles. Hence, the drone cannot reach these subgoals. To overcome the unreachable subgoal issues, a possible approach is to expand subgoals via real-time sensing and topology (Lu and Tseng, 2022). The algorithm in Lu and Tseng (2022) is to enable a drone to explore 3D environments with theoretical guarantees via submodularity.



(a) Environment.



(b) CEM and MCTS.

Fig. 1 The illustration of the CE-MCTS algorithm. **a** The number represents the location of each subgoal. There are 5 subgoals in the map. **b** In the CEM, there are 6 generations. For each generation, the distribution is shown as a 3D figure. The x-axis and y-axis represent subgoal index, and the z-axis means transition rate. In the 1st generation, the distribution is uniform except for the diagonal, since the same subgoal can't be visited twice. In the 6th generation, the robot will select the 3rd subgoal #3 with an 80% probability at the beginning point (the yellow bar pointed by the red arrow). Then, this distribution is utilized to generate the initial MCTS tree. In the MCTS, the branch $1 \rightarrow 2$ is selected to expand nodes through $Selection$, $Expansion$, $Simulation$, and $Backpropagation$ (Color figure online)

This paper is organized as follows: Sect. 2 reviews the relevant work. Section 3 describes the background knowledge of this research. Section 4 introduces the problem formulation and proofs. The proposed algorithms are addressed in Sect. 5. The experiments are demonstrated in Sect. 6. Section 7 reports the conclusions and outlines future work. Finally, the appendix introduces the mathematical formulation of cross-entropy method.

2 Related work

To review the prior work, map exploration is first introduced. To give theoretical guarantees of map exploration problems, greedy algorithms for submodular maximization problems

are reviewed. Evolutionary algorithms, cross-entropy methods and Monte-carlo tree search are further investigated to exceed the greedy approaches.

2.1 Map exploration

It is still an open problem of deploying a robot to build a map in an unknown environment with limited resources (e.g., time, battery, or fuel). There are two major approaches: the frontier area and the next best view.

The frontier is defined as the boundary between the free and unknown areas. The frontier approach is to choose the frontier points as subgoals and then select the next subgoal greedily according to the sensor range (Yamauchi, 1997). In Deng et al. (2020a, b), the researchers adopt gradient-based optimization to efficiently optimize information gain and other measures such as trajectory length simultaneously in simulated environments (e.g., the storage room, lab, and factory). In Zhou et al. (2021), the researchers propose a hierarchical framework which considers both the global tours and local viewpoints to generate a smooth minimum-time local path in an indoor cluttered environment and a forest.

In Cieslewski et al. (2017), the robot selects subgoals from the frontier points in its view to avoid spending much time changing the velocity to reach the goal in the indoor and outdoor real world scenarios. In Umari and Mukhopadhyay (2017), the researchers adopt Rapidly-exploring Random Trees (RRT) to search for the frontier and plan the paths simultaneously in a 2D lab environment. In Luperto et al. (2020), the robot combines prior knowledge of maps and the frontier method in the real world scenario.

The next best view approach is to treat the viewpoint with the highest information gain as the next subgoal (Connolly, 1985). In Amigoni and Gallo (2005) and González-Banos and Latombe (2002), the researchers define the exploration policy with different features (e.g., travelling cost and predictive information gain) in 2D gird maps. In Bircher et al. (2016, 2018), the robot utilizes sampling–based receding horizon path planning and the next-best-view to explore the unknown environment in the indoor and outdoor real world scenarios.

However, the aforementioned methods only utilize the sensor information and can not provide theoretic guarantees.

2.2 Submodular maximization problems

The researchers prove that maximizing monotone set functions with submodular properties has theoretical guarantees through greedy algorithms under variant constraints (e.g., cardinality (Nemhauser et al., 1978), knapsack (Khuller et al., 1999; Sviridenko, 2004), routing (Zhang and Vorobeychik, 2016) or matroid (Fisher et al., 1978)). The various applications include map exploration (Lu and Tseng, 2020; Corah and Michael, 2019), monitoring lake fulmar using sensors (Krause and Guestrin, 2007), sensor placement problems (Krause et al., 2006), planning informative paths for multiple robots (Singh et al., 2006), and searching for a non-adversarial target with multiple robots (Brock et al., 2009).

The researchers in Nemhauser et al. (1978) prove that the greedy algorithm can achieve a $(1-\frac{1}{e})$ approximation ratio under cardinality constraints. Moreover, the researchers in Feige (1998) prove that the greedy solutions are near-optimal unless $P = NP$. In Khuller et al. (1999), Sviridenko (2004), the researchers prove that the cost benefit algorithm, which selects the item with the largest marginal gain at each iteration until the cost exhausts, has a $(\frac{1}{2})(1-\frac{1}{e})$ approximation ratio under knapsack constraints. Furthermore, the researchers in Krause and Guestrin (2005) improve to a $(1-\frac{1}{e})$ approximation ratio by the partial enumeration. In Zhang and Vorobeychik (2016), the researchers reveal that under routing constraints, the generalized cost-benefit (GCB) greedy algorithm achieves a $(\frac{1}{2})(1-\frac{1}{e})\widetilde{OPT}$, where $\widetilde{OPT}$ is the approximated optimum.

2.3 Evolutionary algorithms (EA)

Although the GCB algorithm can achieve a $(\frac{1}{2})(1-\frac{1}{e})$-approximation ratio with $O(n^2)$ time complexity, there is still room for improvement. Hence, evolutionary algorithms (EA) are considered to solve submodular problems with better performance but more computation time. In Friedrich and Neumann (2015), the researchers propose the EA achieving a $(1-\frac{1}{e})$-approximation ratio for maximizing submodular functions under cardinality constraints.

To apply these algorithms to more general applications (e.g., without submodularity), the researchers in Qian et al. (2017) consider the general problem of maximizing monotone set functions under monotone cost constraints and propose the method based on Pareto Optimization (Qian et al., 2015) (called POMC). POMC achieves a $\frac{\alpha_f}{2}(1-\frac{1}{e^{\alpha_f}})$-approximation ratio under general cost constraints, but the computational time could grow exponentially.

The α_f is the submodularity ratio (Zhang and Vorobeychik, 2016), which characterizes how the function f is close to submodularity. Note that $\alpha_f = 1$ if and only if f is submodular. Furthermore, in Roostapour et al. (2019), the researchers prove the capability of POMC in dynamic environments. When the cost budget changes, POMC can maintain a $\frac{\alpha_f}{2}(1-\frac{1}{e^{\alpha_f}})$-approximation ratio. To improve the computational issue, the researchers in Bian et al. (2020) propose EAMC, which is an EA for maximizing monotone functions under general cost constraints. EAMC can achieve a $\frac{\alpha_f}{2}(1-\frac{1}{e^{\alpha_f}})$-approximation ratio in polynomial time, but the performance is worse than POMC in most cases.

2.4 Cross-entropy method (CEM)

The cross-entropy method (CEM) (Rubinstein, 1999; De Boer et al., 2005) aims to deal with two problems. The first one is for rare event simulations. CEM samples a few data to estimate the rare event probability. The second one is for optimization. In the complex space, CEM recursively updates the distributions from the elite samples to find better solutions.

In Costa et al. (2007), the researchers prove that CEM finds optimal solutions with a probability arbitrarily close to 1 under some conditions. In Guillaume et al. (2008), the researchers employ CEM to tune the various parameters in the Monte-Carlo tree search (MCTS) (Coquelin and Munos, 2007) framework of the GO program Mango and leverage its performance under different time or board size settings. In Lanillos et al. (2012), the researchers adopt CEM to search for a dynamic target with limited information in the minimum time.

2.5 Monte-carlo tree search (MCTS)

The goal of MCTS is to find solutions from the expansion search tree (Chaslot et al., 2008). The key to MCTS performance is how to decide or train the tree policy and default policy.

One well known tree policy is Upper Confidence Bounds (UCB), which is the first proposed in Auer et al. (2002). The researchers consider a K-armed bandit problem and propose four polices for the exploration-exploitation trade-off. Moreover, the researchers also prove the bounds of expected regret of four policies after some iterations.

Afterwards, in Kocsis and Szepesvári (2006), the research ers show a Monte-Carlo planning algorithm, Upper Confidence Bounds for Trees (UCT), that is a rollout-based planning and treats the action selection phase as a separate multi-armed bandit. The researchers prove that UCT has the theoretical bound. In Schadd et al. (2012), the researchers define the tree policy with the standard error term in the one-player game and adopt MCTS to achieve the high scores. In Moerland et al. (2020), the researchers introduce a new selection function based on the UCB formula which considers the uncertainty of subtrees below an action. In Xiao et al. (2019), the researchers combine MCTS and maximum entropy policy optimization to improve the efficiency of UCT.

On the other hand, the default policy is essential for the MCTS framework. In James et al. (2017), the researchers demonstrate the performance of different variances of default policies under various situations. In Silver et al. (2016), the researchers utilize deep neural networks to learn the default policy and make the robot defeat the human European Go champion by 5 games to 0. These works demonstrate the significance of the tree policy and default policy selection.

2.6 Coverage path planning (CPP)

The other famous problems, coverage path planning (CPP) (Galceran and Carreras, 2023), are given the map information. The goal of CPP is to cover the total area of interest with minimum overlapping. The researchers in Cao et al. (1988) define the criteria of the CPP in robotic field. There are various robotic applications with CPP. In Yasutomi et al. (1988), the researchers propose the algorithm for driving vacuum cleaning robots to avoid contacting obstacles and walls during cleaning. The researchers in Cheng et al. (2008) provide a method for determining a trajectory for UAVs to reconstruct 3D urban buildings and derive a lower bound on travelling time. In Englot and Hover (2012), the researchers present a sampling-based coverage path planning to generating paths for autonomous underwater vehicles (AUVs) to complete water ship hull inspection.

3 Background knowledge

This section describes the submodularity, CEM and MCTS.

3.1 Submodularity

Definition 1 [*Submodularity* (Nemhauser et al., 1978)] Given a finite set $S = \{1, 2, \ldots, N\}$, a set function $f : 2^N \longrightarrow \mathbb{R}$ has submodularity if it satisfies the **diminishing return** property: for every $S_A \subseteq S_B \subseteq S$, and for all $s \subseteq S$, $f(S_A \cup s) - f(S_A) \geq f(S_B \cup s) - f(S_B)$ holds.

To illustrate the concept of submodularity, an example is shown in Fig. 2. There are three sets: the ground set $S = \{1, 2, 3\}$ and two selected sets, $S_A = \{1\}$ and $S_B = \{1, 2\}$. The set $S_B = \{1, 2\}$ represents that there are two sensor sets at locations 1 and 2. Moreover, $S_A \subseteq S_B \subseteq S$. $f(S_A)$ and $f(S_B)$ mean the coverage of sensors at location $\{1\}$ and $\{1, 2\}$ as shown in Fig. 2a, b, respectively. Figure 2c illustrates the marginal gain of S_A adding s is greater than that of S_B adding s. Mathematically, the coverage function satisfies the diminishing return property. In other words, the objective function for the maximization coverage problem is submodular. Although it is a NP-hard problem, greedy approaches can generate near-optimal solutions (Nemhauser et al., 1978).

3.2 Theoretical guarantees of submodular approaches

This subsection shows theoretical guarantees of submodular and approximated submodular approaches. Since the routing cost is not submodular, submodularity ratio is adopted to analyze the theoretical bounds of GCB.

Definition 2 [*Submodularity Ratio* (Zhang and Vorobeychik, 2016)] Given a non-negative set function f, the submodularity ratio of f is defined as:

$$\alpha_f = \min_{S_A \subseteq S_B, s \notin S_B} \frac{f(S_A \cup s) - f(S_A)}{f(S_B \cup s) - f(S_B)} \tag{1}$$

The two properties of α_f holds:

1. $0 \leq \alpha_f \leq 1$.
2. f is submodular if and only if $\alpha_f = 1$.

For example, the coverage function f is submodular and $\alpha_f = 1$.

Before introducing the lower bounds of GCB and EAMC for submodular functions, some definitions are as follows:

Definition 3 [*Total Curvature* (Conforti and Cornuéjols, 1984)] Let f be a monotone submodular function, the total curvature of f is defined as:

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

The total curvature κ_f characterizes how close a monotone submodular function (f) is to modularity. Moreover, $\frac{1}{\alpha_f} \geq 1 - \kappa_f \geq 0$ can be derived directly from Definition 2.

Definition 4 [*The largest size of feasible solution* K_c (Zhang and Vorobeychik, 2016)] 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 a cost constraint.

K_c represents the solution with the largest set size satisfying the constraint.

Theorem 1 [Lower bound of GCB (Zhang and Vorobeychik, 2016; Qian et al., 2015)] *Given a submodular monotone set function f and a monotone set function c, the GCB approach is to maximize f subject to the cost constraint B. The GCB performance of a set X is*

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

where

$$\tilde{X} = \arg\max_X \left\{ f(X) | c(X) \leq B \frac{\alpha_{\hat{c}}(1 + \alpha_c^2(K_c - 1)(1 - \kappa_c))}{\psi(n) K_c} \right\} \tag{5}$$

and c and $\hat{c}$ represent the cost function and the approximation cost function, respectively. K_c is defined in Definition 4. $\psi(n)$ is the approximation rate of the TSP solver, where $c(X) \leq \hat{c}(X) \leq \psi(n)c(X)$ and $\psi(n) \geq 1$.

Since $0 \leq \alpha_{\hat{c}}, \alpha_c \leq 1$ and $\frac{1}{\alpha_f} \geq 1 - \kappa_f \geq 0$, therefore, $\frac{\alpha_{\hat{c}}(1+\alpha_c^2(K_c-1)(1-\kappa_c))}{\psi(n)K_c} \leq 1$. The special case is that the cost function c is cardinality. Since c is submodular, $\alpha_c = 1$. As c can be computed in $O(1)$, $\alpha_{\hat{c}} = \psi(n) = 1$, In the other words, $\alpha_c = \alpha_{\hat{c}}$. c is modular, so $\kappa_c = 0$. Hence, $\frac{\alpha_{\hat{c}}(1+\alpha_c^2(K_c-1)(1-\kappa_c))}{\psi(n)K_c} = 1$ and $f(\tilde{X}) = f(OPT)$ under cost constraint B.

Lemma 1 [Lower bound of marginal gain(Qian et al., 2017)] *A ground set S, approximation cost function $\hat{c}$ and objective function f are given. For any $S_A \subseteq S$, let $s^* = argmax_{s \notin S_A} \frac{f(S_A \cup s) - f(S_A)}{\hat{c}(S_A \cup s) - \hat{c}(S_A)}$, it holds that*

$$f(S_A \cup s^*) - f(S_A) \geq \alpha_f \frac{\hat{c}(S_A \cup s^*) - \hat{c}(S_A)}{B} \cdot (f(\tilde{X}) - f(S_A)), \tag{6}$$

where B is a cost constraint and $\tilde{X}$ is defined as Eq. 5.

Theorem 2 (**Lower bound of GCB** (Qian et al., 2015)) *Given a monotone set function f and a monotone set function c, the GCB is to maximize f subject to the cost constraint B and obtains a set X such that*

$$f(X) \geq \frac{\alpha_f}{2}\left(1 - \frac{1}{e^{\alpha_f}}\right) f(\tilde{X}), \tag{7}$$

where $\tilde{X}$ is defined as Eq. 5.

Notice that the difference between Theorems 1 and 2 is that the objective function f in Theorem 2 is not necessary to be submodular. It can be α_f submodular.

However, there is still a gap between the GCB and OPT performance. To achieve a better performance with more computation, the evolutionary algorithm (EAMC) is proposed with $\frac{\alpha_f}{2}(1 - \frac{1}{e^{\alpha_f}})$ lower bound (Bian et al., 2020). Empirically, EAMC performance is better than GCB.

Theorem 3 [Lower bound of EAMC (Bian et al., 2020)] *Given a monotone set function f and a monotone set function c, the EAMC is to maximize f and subject to the cost constraint B. Its lower bound is*

$$f(X) \geq \frac{\alpha_f}{2}\left(1 - \frac{1}{e^{\alpha_f}}\right) f(\tilde{X}), \tag{8}$$

where it obtains a set X with $\mathbb{E}[T] \leq 2en^2(n+1)$. $\mathbb{E}[T]$, n and e represent the expected number iteration, the size of the ground set and Euler's number, respectively.

3.3 CEM

In Rubinstein (1999), the researcher proposes the cross-entropy method (CEM), an estimation of distribution algo-

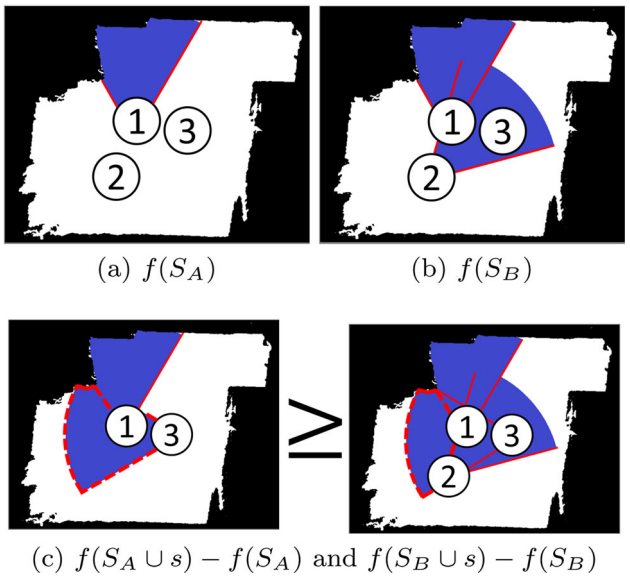


Fig. 2 Illustration of submodularity. The decimal number represents the sensor location. The blue and white colors represent the covered and uncovered areas, respectively. **a** $f(S_A)$ represents the covered area by $S_A = \{1\}$. **b** $f(S_B)$ represents the covered area by $S_B = \{1, 2\}$. **c** The marginal gain is the area within the red dash lines after adding a set $s = \{3\}$. The left figure depicts $f(S_A \cup s) - f(S_A)$, and the right figure depicts $f(S_B \cup s) - f(S_B)$ (Color figure online)

rithm (EDA), to estimate rare events and optimize combinatorial problems as follows:

$$\mathbf{X}^* = \arg\max_{\mathbf{X}} \quad f(\mathbf{X}), \tag{9}$$

where $\mathbf{X}$ is a n-dimension vector and f is the objective function. As shown in Fig. 3a, suppose that $n = 1$ and $f : \mathbb{R} \longrightarrow \mathbb{R}$. The goal is to find the maximum of f. The CEM is to recursively maintain the parametric distribution from the elites samples E to generate the new population. The E is defined as follows:

$$E = \{x \in \mathbb{R} \mid f(x) \geq \gamma\}, \tag{10}$$

where γ is the threshold.

As Fig. 3a shows, there are three steps of CEM for optimization: sample, filter, and update. Assume $h(\cdot; \mu, \sigma)$ is a Gaussian distribution with a mean (μ) and standard derivation (σ). The green line in Fig. 3b represents the samples $x_1, x_2, \ldots, x_N$ with $h(\cdot; \mu_t, \sigma_t)$. The grey dash line and solid line represent μ_t and σ_t, respectively. The red line in Fig. 3c represents the threshold γ_t depending on the quality of $\{x_i\}_{i=1}^N$. For example, γ_t is set to the best 1% of $\{x_i\}_{i=1}^N$, where 1% is a parameter called "elite rate" (ρ). In Fig. 3d, $(\mu_{t+1}, \sigma_{t+1})$ is updated and the next generation $\{x_i\}_{i=1}^N$ is sampled with $(\mu_{t+1}, \sigma_{t+1})$ until the stop criterion is met (e.g., loop iterations).

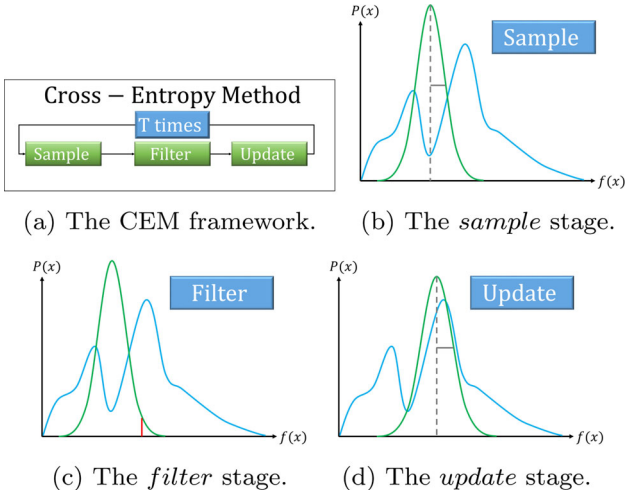


Fig. 3 Illustration of CEM. **a** represents three steps *Sample*, *Filter*, and *Update* in CEM. The x-axis is the objective function f and the y-axis is the probability, respectively. The blue curve is an unknown probability density function (PDF) of the random variable x. The green line represents N samples via mean and standard derivation (μ,σ). **b** the *sample* step in CEM. **c** the *filter* step in CEM, and the red line means the threshold γ_t. **d** the *update* step in CEM (Color figure online)

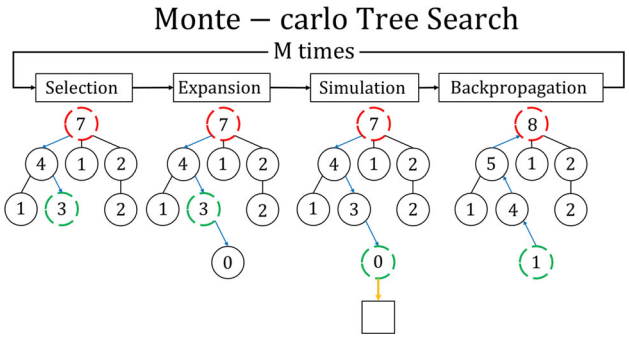


Fig. 4 The MCTS framework. The flowchart includes *Selection*, *Expansion*, *Simulation*, and *Backpropagation*. The decimal number represents the total simulation times at the corresponding node. The node with a red dash circle represents the root node. The node with a green dash circle represents the current node. The yellow arrow represents the expanding process of the default policy. The box represents the outcome of the branch (Color figure online)

3.4 MCTS

The MCTS is a probabilistic approach to search for the utility outcomes via four steps as follows (see Fig. 4):

Selection At the state s, the next action is chosen recursively according to the selection function (the tree policy), which balances exploration and exploitation until the leaf node is reached. As the *Selection* shown in Fig. 4, the agent starts at the root node and expands the tree via the tree policy until it reaches an unobserved child node (the green dash node).

Expansion Determining the exploration or exploitation is the key of MCTS. The simplest strategy (Coulom, 2006) is

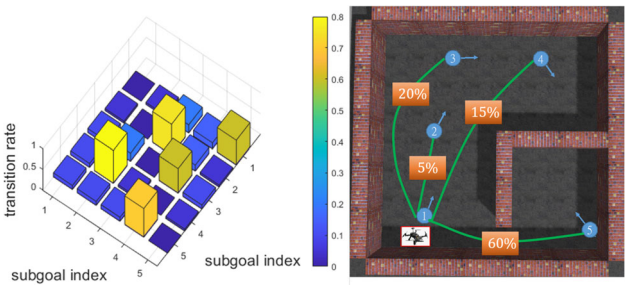


Fig. 5 The left figure represents the transition matrix $D = [d_{ij}]$. The x-axis and y-axis represent subgoals indexes. The z-axis represents the transition rate. The right figure is to illustrate the concept of the D. The decimal number with a blue circle represents the subgoal index. The decimal number in the orange box represents the transition probability d_{ij} between #i subgoal and #j subgoal (Color figure online)

that when the state s is not expandable fully (actions not been explored), s selects the unobserved action as the next action and switches to *Simulation* stage. As the *Expansion* shown in Fig. 4, the agent selects the unexplored child node (the node with "0") to expand.

Simulation For the rest of selections, the tree expands according to some default policies. The default policy, depending on prior knowledge, is a uniform random policy. As the *Simulation* shown in Fig. 4, the agent expands the search tree until the budget exhausts and obtains the value.

Backpropagation After the stop criterion (e.g. time or cost constraints) is satisfied, the information of traversed tree nodes (e.g. simulated time, win rate, or reward) is updated. As the *Backpropagation* shown in Fig. 4, the simulation time of the traversed tree nodes is updated.

Since the search space of MCTS grows exponentially in coverage maximization problems, random methods for building the initial tree are inefficient. If the root tree is near the optimal trajectory, MCTS could find the better solutions efficiently. Although the default policy is the key to affect MCTS performance, it is unknown. Hence, the cross-entropy method (CEM) could be adopted for building the initial tree and training the default policy simultaneously.

4 Proposed CE-MCTS

This section describes the proposed CE-MCTS, relevant theorems, and its terminal condition.

4.1 CE-MCTS

The coverage maximization problem with routing constraints is defined as follows: Given N fixed subgoals in the map, the goal is to find subgoals that maximize the total number of free grids within the budget constraints (e.g., action time, or

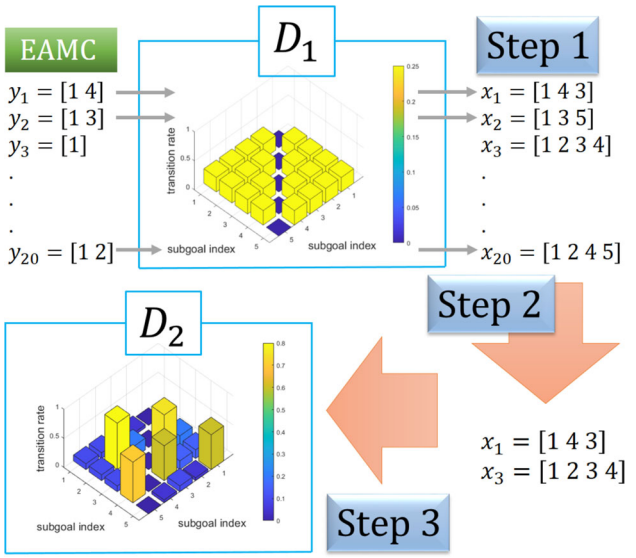


Fig. 6 The toy example of CEM. The size of ground set S is 5. The parameters N and ρ are 20 and 0.1, respectively

battery usage). Mathematically,

$$\max f(S) \quad s.t. \quad c(S) \leq B \tag{11}$$

, where the ground set is $S = \{1, 2, 3, \ldots, N\}$, and f is a coverage function $f : 2^N \longrightarrow R^+$. Since the environment is known, f is an oracle, which outputs R^+. In other words, given any input sets, f will output the coverage values directly. The $c(\cdot)$ and B represent the cost function and cost budget, respectively.

As Fig. 5 illustrates the CE-MCTS includes two parts: CEM and MCTS. The first part is CEM. The transition matrix $D = [d_{ij}]$ represents the transition rate from subgoal #i to subgoal #j and $1 \leq i, j \leq n$. There are 5 subgoals. $V = \{1, 2, 3, 4, 5\}$ represents the index of each subgoal. Since the subgoal #i must not be visited twice, $d_{ii} = 0$ for all i. Assume $[d_{11}, d_{12}, d_{13}, d_{14}, d_{15}] = [0\,0.05\,0.20\,0.15\,0.60]$. d_{12} represents the agent choosing subgoal #2 with a 5% probability to append to the trajectory. If the selected subgoal is not valid (no more budget to reach it), it is replaced by a random valid subgoal.

In the first part, CEM, there are three steps as follows (see Fig. 3):

1. *Sample* Sample N data from D_t.
2. *Filter* Evaluate coverage of each sampling data via the coverage function (f) and choose the elites ($f(x) \geq \gamma_t$) from them.
3. *Update* Build D_{t+1}.

The details of three steps (sample, filter, and update) are illustrated in Fig. 6 and introduced as follows: First, EAMC

is adopted to sample a path y_i. It remains budgets for exploration and expands y_i via D_1 until the budget is exhausted to obtain x_i. Since there is no prior distribution, D_1 is assumed as the uniform distribution with the zero diagonal. Second, f is adopted to evaluate the coverage values of $\{x_i\}_{i=1}^{20}$ (see Step 1 in Fig. 6). Second, it is to choose the best N_{elites} trajectories of $\{x_i\}_{i=1}^{20}$, where $N_{elites} = \lceil 20 \times 0.1 \rceil = 2$ in Fig. 6. The best elites are x_1 and x_3 (see Step 2 of Fig. 6). Third, D_1 is updated by solving Eq. 20 for maximization coverage problems and obtaining D_2 and

$$d_{ij} = \frac{\sum_{k=1}^{N_{elites}} I_{\{x_k \in S_{ij}\}}}{N_{elites}}, \tag{12}$$

where I is an indicator function and S_{ij} represents the sets of paths from i to j.

In the second part, MCTS, the initial tree is built from the CEM. Similarly, EAMC (Bian et al., 2020) is adopted to sample a path y_i and executes MCTS for decision making. The tree policy is for $Selection$ while the default policy is for $Simulation$.

In $Selection$ stage, the tree policy adopts the UCB (Auer et al., 2002) formula to balance between exploration and exploitation:

$$\bar{X} + C\sqrt{\frac{\ln t(N)}{t(N_i)}} \tag{13}$$

where $\bar{X}$ is the average coverage of the node. $t(N)$ stands for the number of simulations for the parent nodes, and $t(N_i)$ represents the number of simulations for the considered child nodes. C is the parameter for the trade-off between exploration and exploitation. The higher value of C means that it encourages the agent in exploration behavior.

In $Simulation$ stage, the transition matrix D_T, which is trained in the CEM part, is adopted for $Simulation$. The four phases in MCTS are repeated until the terminal condition is satisfied (see Fig. 4).

4.2 Theoretical guarantees of CE-MCTS

Due to MCTS suffers from expensive computation time, the CE-MCTS adopts EAMC for sampling the branches of the search tree both in the CEM and MCTS parts. Hence, Theorem 3 also holds for CE-MCTS.

Theorem 4 [Lower bound of CE-MCTS (Bian et al., 2020)] *Given a monotone set function f and a monotone set function c, the CE-MCTS is to maximize f and subject to the cost constraint B. Its lower bound is*

$$f(X) \geq \frac{\alpha_f}{2}\left(1 - \frac{1}{e^{\alpha_f}}\right) f(\tilde{X}), \tag{14}$$

where it obtains a set X with $\mathbb{E}[T] \leq 2en^2(n+1)$. $\mathbb{E}[T]$, n and e represent the expected number iteration, the size of the ground set and Euler's number, respectively.

Although the GCB, EAMC, and CE-MCTS achieve a lower bound $\frac{\alpha_f}{2}(1 - \frac{1}{e^{\alpha_f}})f(\tilde{X})$, $f(\tilde{X})$ is unknown. If $f(\tilde{X})$ is known, the percentage of $f(\tilde{X})$ can be adopted as the terminal criterion for the sampling-based algorithm (e.g., CE-MCTS, EAMC). However, computing $f(\tilde{X})$ is still a NP-hard problem (Nemhauser et al., 1978). Hence, the Triangular estimation (TE) algorithm is proposed to estimate $f(\tilde{X})$ as the tighter upper bound. Moreover, the key parameters α_c and $\alpha_{\hat{c}}$ affecting the performance of $f(\tilde{X})$ are also estimated by the TE algorithm. As shown in Eq. 5, if $\alpha_{\hat{c}}$ and $\alpha_{\hat{c}}$ are close to 1, $f(\tilde{X})$ is closer to OPT.

Lemma 2 (The terminal criterion of sampling-based method) *Assume a monotone set function f, a cost function c, the threshold η ($0 < \eta \leq 1$) and the tight upper bound ($\bar{b}$) of $f(\tilde{X})$ are given. If c is the cardinality function, then once the outcome of sampling-based method is higher than $\eta \cdot \bar{b}$, the sampling-based algorithm terminates and returns the outcome.*

For instance, if $\eta = \frac{\alpha_f}{2}(1 - \frac{1}{e^{\alpha_f}})$, Lemma 2 connects to Theorem 4. Theorem 4 shows the upper bound of total expected iteration number, and Lemma 2 offers the terminal criterion.

4.3 Terminal conditions of CE-MCTS

The terminal criteria of MCTS is usually the computation time or iteration number. However, these criteria do not give performance guarantees. For example, there are 100 samples of CE-MCTS. The best sample of CE-MCTS is X', and its performance is $f(X')$. How could an algorithm decide if the $f(X')$ is good enough to stop?

To find terminal criteria with performance guarantees, the tighter upper bound of $f(\tilde{X})$ defined in Eq. 5 has to be estimated. Meanwhile, the performance of $f(\tilde{X})$ depends on the submodularity ratio (α_c) of cost functions. However, computing the submodularity ratio needs to compute all possible set combinations, which is infeasible. Therefore, the proposed triangular estimation (TE)[1] is a partial enumeration algorithm to compute only three sets combinations for finding two values (α_c' and $\bar{b}$). α_c' is the tighter upper bound of α_c and $\alpha_{\hat{c}}$ for estimating the cost constraint of $f(\tilde{X})$ while $\bar{b}$ is the tighter upper bound of $f(\tilde{X})$ ($f(\tilde{X}) \leq \bar{b} \leq 1$). In other words, if α_c' is closer to 1, $f(\tilde{X})$ is closer to OPT and $\bar{b}$ can be adopted as the terminal criteria.

The purpose of TE method is to simultaneously estimate the upper bounds (α_c' and $\bar{b}$) of α_c, $\alpha_{\hat{c}}$, and $f(\tilde{X})$ for the

[1] In the geometry, the three sets consists a triangle. Hence, it is called triangular estimation.

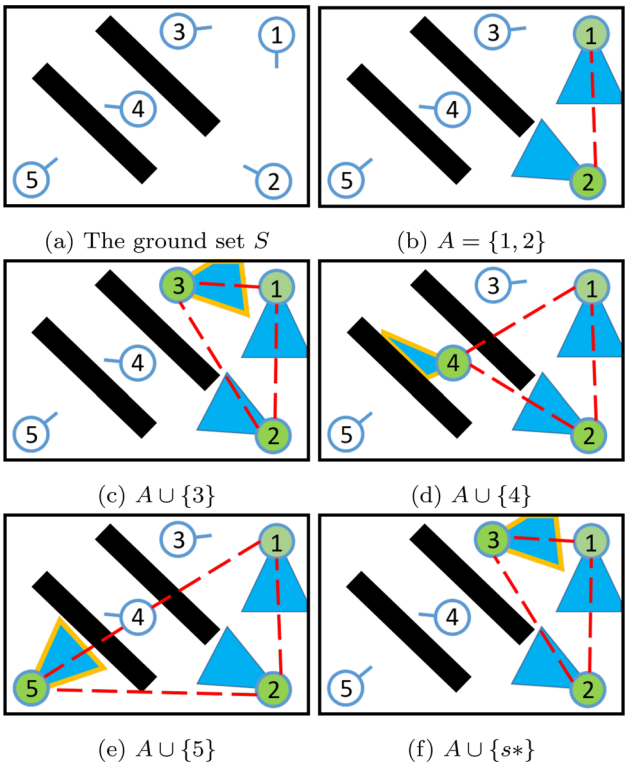


Fig. 7 The decimal numbers represent the sensor index. The black and blue areas represent obstacle and covered areas, respectively. The red dash lines represent the Euclidean distance of the candidate set (green nodes). **a** The ground set $S = \{1, 2, \ldots, 5\}$. **b** $A = \{1, 2\}$. **c**–**e** represent $A \cup \{3\}$, $A \cup \{4\}$ and $A \cup \{5\}$, respectively. The areas within the solid golden lines represent $f(A \cup s) - f(A)$ where $s \notin A$. **f** The next sensor number (3) is selected according to $s^* = argmax_{s \notin A} \frac{f(A \cup s) - f(A)}{\hat{c}(A \cup s) - \hat{c}(A)}$ (Color figure online)

terminal criterion. As shown in Fig. 7, suppose that the coverage function (f), routing cost function (c), ground set (S) with size N and cost constraint (B) are given. The following example demonstrates how to estimate α_c, $\alpha_{\hat{c}}$, and $f(\tilde{X})$. Figure 7a represents the ground set (S) with $N = 5$. Suppose $S_A = \{1, 2\}$ (see Fig. 7b), $\bar{b} = 1$, and $\alpha'_c = 1$ at initialization.

In Fig. 7c, assume $B = A \cup \{3\}$, the first step is to compute two values, α_c of B and $\frac{f(B) - f(A)}{c(B) - c(A)}$. α_c of B ($\bar{\alpha}$) can be evaluated according to Definition 2 by enumeration due to $|B| = 3$. The four enumeration terms are as follows:

1. $S_A = \{1\} = S_B$, $s = \{2\}$, and the first term is $\frac{c(\{1\} \cup \{2\}) - c(\{1\})}{c(\{1\} \cup \{2\}) - c(\{1\})} = 1$.
2. $S_A = \{1\} = S_B$, $s = \{3\}$, and the second term is $\frac{c(\{1\} \cup \{3\}) - c(\{1\})}{c(\{1\} \cup \{3\}) - c(\{1\})} = 1$.
3. $S_A = \{1\} \subseteq \{1, 2\} = S_B$, $s = \{3\}$, and the third term is $\frac{c(\{1\} \cup \{3\}) - c(\{1\})}{c(\{1 \cup 2\} \cup \{3\}) - c(\{1 \cup 2\})}$.
4. $S_A = \{1\} \subseteq \{1, 3\} = S_B$, $s = \{2\}$, and the fourth term is $\frac{c(\{1\} \cup \{2\}) - c(\{1\})}{c(\{1 \cup 3\} \cup \{2\}) - c(\{1 \cup 3\})}$.

Since c is the routing cost function, $c(\{1\}) = 0$. Mathematically, by Definition 2, $\bar{\alpha} = min\{1, \frac{c(\{1,3\})}{c(S_B) - c(\{1,2\})}, \frac{c(\{1,2\})}{c(S_B) - c(\{1,3\})}\} \geq \alpha_c, \alpha_{\hat{c}}$. Hence, α'_c is updated according to $\alpha'_c = min\{\alpha'_c, \bar{\alpha}\}$ and $\frac{f(B) - f(A)}{c(B) - c(A)}$ is stored to the buffer.

The TE method then continues to compute other set combinations (see Fig. 7c–e). Finally, as shown in Fig. 7f, {3} is selected as s^* according to

$$s^* = argmax_{s \notin S_A} \frac{f(S_A \cup s) - f(S_A)}{\hat{c}(S_A \cup s) - \hat{c}(S_A)}.$$

$\bar{b} = min\{\bar{b}, \frac{B}{\alpha_f} \frac{f(S_A \cup s^*) - f(S_A)}{c(S_A \cup s^*) - c(S_A)} + f(S_A)\} \geq f(\tilde{X})$ is updated by Lemma 1. Notice that $\alpha_f = 1$ since f is the coverage function.

Consequently, α'_c is to qualify the performance of $f(\tilde{X})$. α'_c is lower, and $f(\tilde{X})$ is lower. Moreover, $\bar{b}$ is viewed as the tight upper bound of $f(\tilde{X})$, and the percentage of $\bar{b}$ can be adopted for the terminal criterion.[2] For instance, suppose the desired lower bound performance is 31% of $f(\tilde{X})$. If the CE-MCTS finds a solution that outperforms 31% of $\bar{b}$, then CE-MCTS terminates the process.

5 Proposed algorithms

There are three algorithms: CEM, CE-MCTS, and TE. In Algorithm 1, the inputs are the objective function (f), approximation cost function ($\hat{c}$), and a cost budget (B). The outputs are the sampling paths ($Tree$), D, P, u, and v. D represents the updated distribution, which is the uniform distribution at the initialization. P, u, and v represent the candidate branch set P, the branch set u ordering by the surrogate function, and the branch set v ordering by f (Bian et al., 2020), respectively.

Line 1 is to check the criterion (e.g., loop iteration). Line 2 is to initialize the path set at each generation. Line 4 is to select a branch from P uniformly. Line 5 is to obtain x' by bit-wise mutation. Line 6 is to replace x' with x if $c(\hat{x}) > B$ holds. Line 9 is to update P, u, and v by Bian et al. (2020). Lines 8–11 are to expand x' via D until the budget B is exhausted. Line 12 is to append x' to G. Line 13 is to append x' to $Tree$. Line 15 is to filter the best $\lceil N \times \rho \rceil$ trajectories in G. Line 16 is to obtain D' by solving Eq. 20. Line 17 is to update D with D' by the step size α which balances exploration and exploitation. A faster convergence to a specific distribution is encouraged by a higher α, but the risk of falling into the local optimum increases.

In Algorithm 2, the inputs are the objective function (f), approximation cost function ($\hat{c}$), and cost budget (B) $Tree$,

[2] The 31% is the lower bound of GCB. $0.5 \times (1 - 1/e) \approx 31\%$.

Algorithm 1: CEM algorithm

Input: objective function f, a monotone approximation cost function $\hat{c}$, and a cost budget B
Output: $Tree$, D(default policy), P, u and v
Parameters : N(population size), ρ(elites rate), α(adaption rate)
Initialization : D

while $termination()$ **do**
  $G = \{\}$
  for $n = 1$ to N **do**
    Select x from P randomly
    Generate x' by applying bit-wise mutation to x
    if $\hat{c}(x') > B$ **then** $x' \leftarrow x$
    end if
    while $!IsExausted(x')$ **do**
      Update P, u and v
      Expand x' via D
    end while
    $G = G \cup x'$
    $Tree = Tree \cup x'$
  end for
  Select $\lceil N \times \rho \rceil$ elites in G
  Build D' by solving Equ. 20
  $D \leftarrow (1-\alpha)D + \alpha D'$
end while

Algorithm 2: CE-MCTS algorithm

Input: objective function f, a monotone approximation cost function $\hat{c}$, a budget B, $Tree$, D(default policy), P, u, v, and L(loop iteration)
Output: θ

for $i = 1$ to L **do**
  Select x from P randomly
  Generate x' by applying bit-wise mutation to x
  if $\hat{c}(x') > B$ **then** $x' \leftarrow x$
  end if
  while $!IsExhausted(x')$ **do**
    Update P, u and v
    if $!IsFullObserved(x')$ **then**
      $x' \leftarrow Expansion(x', Tree)$
      $x' \leftarrow Simulation(x', D)$
    else
      $x' \leftarrow Selection(x', Tree)$
    end if
  end while
  $Tree \leftarrow Backpropagation(x', Tree)$
end for
$\pi \leftarrow argmax_{x \in P} f(x)$

D, P, u and v are built by Algorithm 1. L represents the loop time. The output is π which is a searched trajectory with the highest objective value. Lines 2–5 are similar to lines 4–7 in Algorithm 1. Lines 6–16 are the progress of MCTS. Line 8 is to check whether x' leaf nodes are expanded. If it is not, line 9 is to execute $Expansion$ and line 10 is to launch $Simulation$ via the trained default policy D in Algorithm 1. If x' leaf nodes are not fully expanded, line 12 is to adopt the UCB formula (Coquelin and Munos, 2007) for the $Simulation$. Line 15 is to run the $Backpropagation$ of MCTS.

Algorithm 3: Triangular estimation algorithm

Input: ground set $S = \{s_1, s_2, \ldots, s_N\}$, objective function f, approximation cost function $\hat{c}$, a cost constraint B
Output: α_c', $\bar{b}$
Initialization : $\alpha_c' = 1, \bar{b} = 1$

1: $S_A \leftarrow \{s_1\}$
2: $s^* \leftarrow argmax_{s \notin S_A} \frac{f(S_A \cup s) - f(X)}{\hat{c}(S_A \cup s) - \hat{c}(S_A)}$
3: $\bar{b} \leftarrow \min\{\bar{b},\ \frac{B}{\alpha_f} \frac{f(S_A \cup s^*) - f(S_A)}{\hat{c}(S_A \cup s^*) - \hat{c}(S_A)} + f(S_A)\}$
4: **for** $i = 2$ to $N-1$ **do**
5: $S_A \leftarrow \{s_1,\ s_i\}$
6: **for** $j = i+1$ to N **do**
7: $S_B \leftarrow \{s_1,\ s_i,\ s_j\}$
8: $\bar{\alpha} \leftarrow$ Calculate α_c of S_B
9: $\alpha_c' \leftarrow \min\{\alpha_c',\ \bar{\alpha}\}$
10: **end for**
11: $s^* \leftarrow argmax_{s \notin S_A} \frac{f(S_A \cup s) - f(S_A)}{\hat{c}(S_A \cup s) - \hat{c}(S_A)}$
12: $\bar{b} \leftarrow \min\{\bar{b},\ \frac{B}{\alpha_f} \frac{f(S_A \cup s^*) - f(S_A)}{\hat{c}(S_A \cup s^*) - \hat{c}(S_A)} + f(S_A)\}$
13: **end for**

The Algorithm 3 is a partial enumeration algorithm to decide the terminal condition of CE-MCTS. The input data is the ground set (S) with $|S| = N$, an objective function (f), an approximation cost function ($\hat{c}$), and a cost budget (B). The output data is α_c' and $\bar{b}$ where $\alpha_c, \alpha_{\hat{c}} \leq \alpha_c' \leq 1$ and $0 \leq f(\tilde{X}) \leq \bar{b} \leq 1$ ($f(\tilde{X})$ is defined as Eq. 5 and is normalized). α_c' and $\bar{b}$ are equal to 1 at the initialization. Line 1 is to set $S_A = \{s_1\}$. Line 2 and 3 are to update $\bar{b}$ according to Lemma 1. Line 5 is to assign $S_A = \{s_1,\ s_i\}$. Line 7 is to assign $S_D = \{s_1,\ s_i,\ s_j\}$. Line 8 is to evaluate α_c of S_B via Definition 2. Line 9 is to update α_c'. Line 11 and 12 are similar to Line 2 and 3.

The computational complexity of CE-MCTS (see Algorithm 2) is as follows: Suppose N is the number of subgoals. $T(m)$ represents the computational complexity of the TSP-solver ($\hat{c}(\cdot)$) to calculate the route cost. The computational complexity of four phases: $Expansion$, $Simulation$, $Selection$, and $Backpropagation$ are $O(T(m)N)$, $O(T(m)N^2)$, $O(T(m)N)$, and $O(N)$, respectively. Hence, the computational complexity of the proposed algorithm is $O(T(m)N^2L)$ where L is the manual parameter to control the search times (e.g., $L = N^2$).

The computational complexity of GCB and EAMC are $O(T(m)N^2)$ and $O(T(m)L)$, respectively. The difference between CE-MCTS and GCB is as follows: GCB is a greedy approach, so GCB outputs a deterministic trajectory. Since CE-MCTS is a sampling based algorithm, CE-MCTS evaluate L trajectories via the search tree. Then, CE-MCTS outcomes the trajectory with the highest coverage value. The difference between CE-MCTS and EAMC is that CE-MCTS utilizes the budget resources via the built distribution. (see Algorithm 1)

6 Experiments

To evaluate the performance of different approaches (e.g., GCB, EAMC, and CE-MCTS), there are four experiments: the maximization coverage problem (EX1), the search problem (EX2), the terminal criterion (EX3), and the computation time problem (EX4). The EX1 is to evaluate the coverage of different approaches in four maps under various routing constraints. The EX2 is to evaluate the search time of different approaches via a drone in the real world. The EX3 is to investigate the feasibility of the terminal criterion to save the computation time. There are two terminal criterions as followings: The EX3-1 is to estimate $f(\tilde{X})$ and $\alpha_c, \alpha_{\hat{c}}$ to verify the CE-MCTS performance. If CE-MCTS finds a solution meets the desired lower bound performance, CE-MCTS terminates. The EX3-2 is to set a manual parameter T: If the algorithm fails to find a better solution for T consecutive times, the algorithm terminates. The EX4 is to evaluate the computation time of different approaches under varying sizes of subgoals.

6.1 Experiment setup

In the EX1, the coverage performance is compared in the four maps: three 2D maps and one 3D map. In the 2D maps (see Fig. 8a–c), the subgoals (S) with $|S| = 48$ are shown in Fig. 8e. The ground set S is generated randomly, and the left top point is the fixed beginning point (the red arrow). In the 3D map, Fig. 8f shows the entire third floor of National Central University's General Education Building and the ground set S. The dimension of this experimental space is 30×60 m^2 with corridors involving narrow passages, several occlusions, and other geometrical features. The map and sensor limit are shown in Table 1. The cost function c is measured by Euclidean distance. Note that, since computing the minimal routing cost is a NP-hard problem, c is approximated by $\hat{c}$ via a TSP-solver with a polynomial time complexity.

In the EX2, the map is a 10 m x 5 m hallway at the Mathematics Department of National Central University (see Fig. 9a). Figure 9b shows the ground set $|S| = 41$, which is generated by the fixed distance. The fixed beginning point is the subgoal #1. The target is a person who appears at 5 different locations (see Fig. 9b).

As shown in Fig. 10a, the robot platform is the Intel ready-to-fly (RTF) drone with a R200 RGBD sensor, an Intel Movidius Neural Compute SDK sensor, and a T265 sensor. The R200 RGBD sensor is a forwarding camera for exploring environments and providing images for the Intel Movidius Neural Compute SDK (NCSDK) sensor to detect the target (see Fig. 10b). The R200 range is 3.5m and FOV is $[60° \times 46° \times 70°]$. The T265 sensor is a forwarding camera for localization. The drone's operating system is Ubuntu 16.04 and runs the Robot Operating System (ROS). The parameters

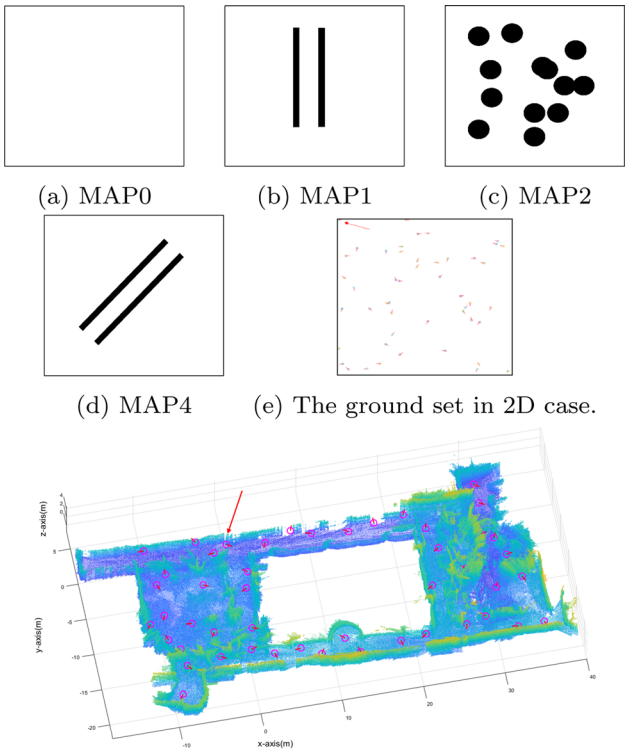


Fig. 8 The simulation environments in EX1 and EX3. **a**–**d** The black grids and the white grids represent obstacles and free areas, respectively. **e**, **f** The circles depict the subgoal locations. The red arrows show the beginning points in the 2D and 3D cases. **a**–**c**, **f** The four maps are adopted for EX1. **d** The map is adopted for EX3. The color of 3D pointcloud points shows the height of points (Color figure online)

of drone hardware setup are shown in Table 2. The cost function is defined as: $\sqrt{(\triangle x)^2 + (\triangle y)^2 + (\triangle z)^2} + (\triangle \theta)\frac{v_{max}}{\omega_{max}}$.

There are three steps in the search process. First, the trajectory π is computed offline subjects to the cost constraints B. Second, π is imported to the drone. Third, the drone begins searching for the target through π.

Along with π, the drone updates the probability of target detection (Z) according to the outcome of NCSDK via the Bayes filter. When arriving at each subgoal, the drone hovers for 2 s to update $P(Z = 1)$.[3] If the target is found ($P(Z = 1) \geq P_{thr}$), the drone lands; otherwise, the drone continues to search until the search is terminated. Once the drone finds the target, it will land on the ground, and the flight time is recorded as the search time. For each approach, the drone searches for the target 5 times at each location.

In the EX3, the obstacle map (see Fig. 8d) is adopted to verify the terminal conditions. The map size is 600 $\times$ 600 (units). There are two cost functions: routing cost, and cardinality cost. The routing cost constraints are 1000 and

[3] $P(Z = 1)$ denotes the probability of the target is in the camera view. P_{thr} is the detection probability threshold.

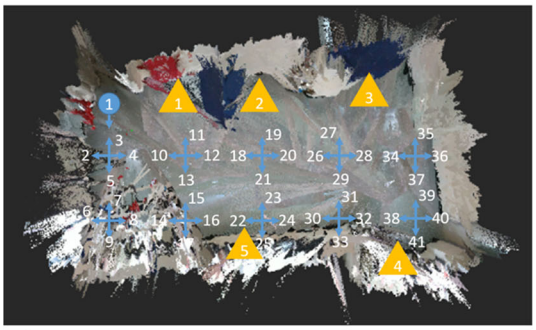
Table 1 The map and sensor parameters of EX1

Type	Map size	$\|S\|$	Sensor range	Sensor FOV
2D	600 × 600 (units)	48	150 (units)	114.6°
3D	60 × 30 × 3 (m)	48	4.5 (m)	[60° × 46° × 70°]

Table 2 The parameters of CE-MCTS and drone hardware setup

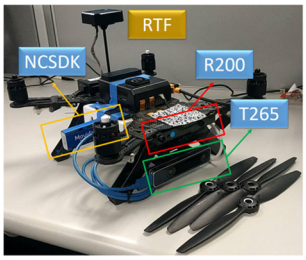
Parameter	Description	Value
N	Population size	$(\|S\|-1)(\|S\|-1)$
ρ	Elites rate	0.02
α	Adaption rate	0.1
k	CEM termination	5
Altitude	Flight altitude	1 (m)
ω_{max}	Limitation of angular velocity	35 (°/s)
v_{max}	Limitation of linear velocity	1 (m/s)

(a) The environment of EX2.

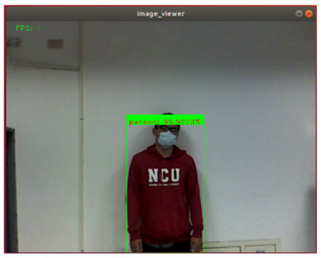


(b) The top view of EX2 map is built by RTABMAP.

Fig. 9 **a** The map of EX2. **b** The decimal numbers with the blue arrows represent the subgoal information, and the left-top blue subgoal #1 is the fixed beginning point. The yellow triangles indexed by the decimal numbers represent the target locations (Color figure online)



(a) The RTF and sensors.



(b) The detection of NCSDK.

Fig. 10 **a** The red box indicates the R200, the green box indicates the T265, and the yellow box indicates the NCSDK. **b** The R200 provides the RGB images. There is a person with the red sweater standing there, and then the NCSDK detects the target in the image. The light green box represents the target location in the image with the detection probability (Color figure online)

2000 units, and cardinality cost constraints are 20 and 30. The maximal generation number is 20 in EX3-1. In EX3-2, it is the terminal condition to set $T_{CE-MCTS} = 5$ and $T_{EAMC} = 10$.

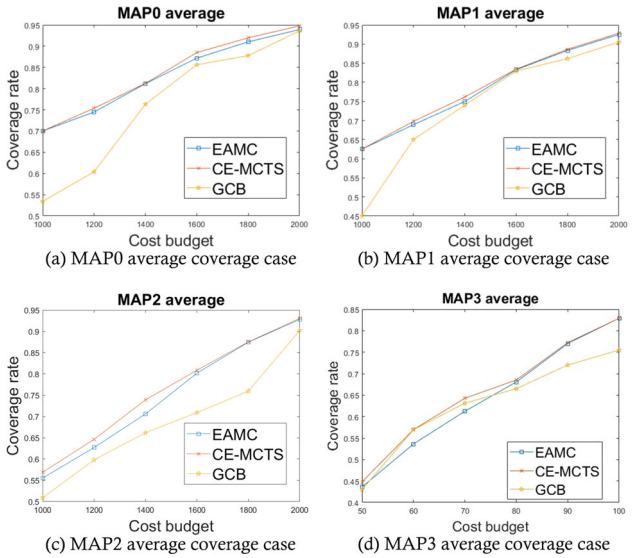
Note that, CEM termination (k) is set 5 (see Table 2) and $T_{CE-MCTS} = 5$. In other words, the CE-MCTS runs at least 10 generations. Under this settings, EAMC and CE-MCTS run at least 10 generations. The other setups are the same as the EX1 setup.

The parameters of CE-MCTS for EX1, EX2 and EX4 are shown in Table 2. The $termination()$ in Algorithm 1 is defined as follows:

$$termination() = \begin{cases} 1, & \gamma_t = \gamma_{t-1} = \cdots = \gamma_{t-k} \\ 0, & otherwise \end{cases}$$

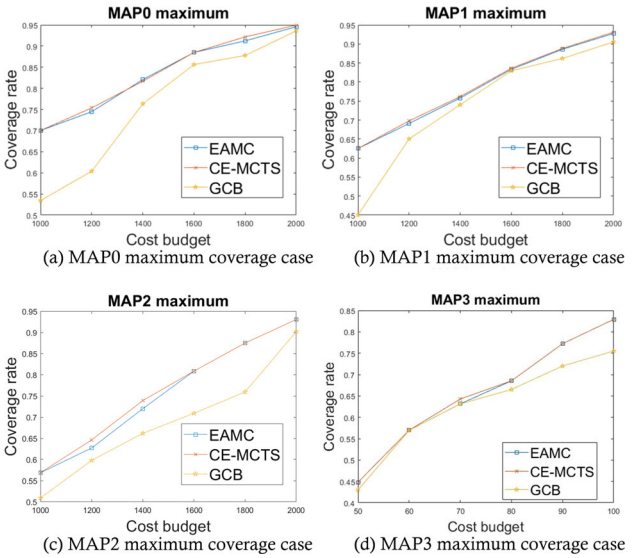
where γ_t represents the low bound in t-generation. The γ_t is equal to the objective value low bound of the best $\lceil N \times \rho \rceil$ samples. In other words, if the low bound stays and keeps for k-generation, the CEM stops and turns to MCTS (see Algorithm 2).

In the EX4, the experiment runs with Intel i7-8700K CPU on MATALB R2019b and the empty map (see Fig. 8a) is



(a) MAP0 average coverage case (b) MAP1 average coverage case
(c) MAP2 average coverage case (d) MAP3 average coverage case

Fig. 11 The average coverage result of EX1. The x-axis and the y-axis represent cost budget constraints and coverage rate, respectively



(a) MAP0 maximum coverage case (b) MAP1 maximum coverage case
(c) MAP2 maximum coverage case (d) MAP3 maximum coverage case

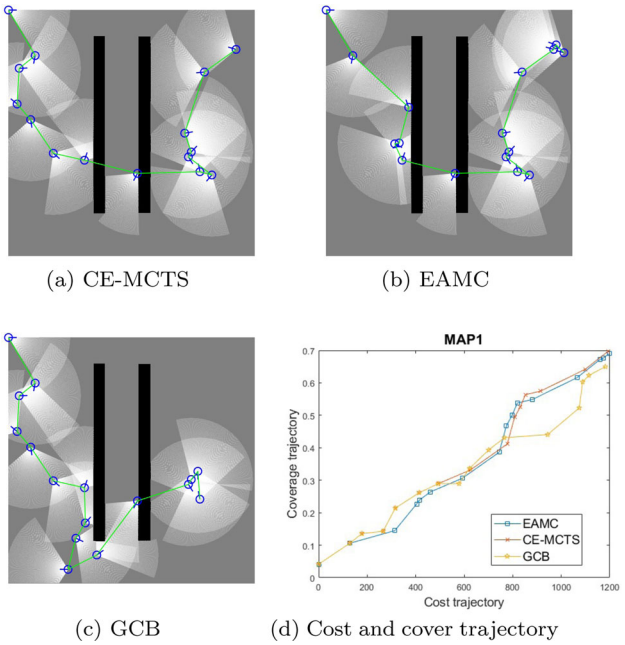
Fig. 12 The maximum coverage result of EX1. The x-axis and the y-axis represent cost budget constraints and coverage rate, respectively

adopted to verify the computation time. The number of subgoals is 16–20 and the cost function is routing cost.

6.2 EX1: 2D and 3D coverage maximization

In the EX1, coverage performance in the four maps: three 2D maps and one 3D map (see Fig. 8a–c, f) is as follows: The parameters are shown in Table 2. As shown in Fig. 8d, f, the ground set $|S| = 48$ in both 2D and 3D cases, and the population size $N = (48-1) \times (48-1) = 2209$. In 2D and 3D cases, the coverage performance is compared under 1000–2000 units, and 50–100 ms, respectively. Since EAMC and CE-MCTS are randomized algorithms, the experiments are run 3 times independently and the mean values are compared. Each time, CE-MCTS and EAMC run 48 generations. The trajectory with the highest coverage value is recorded in each generation. The results of mean coverage are shown in Figs. 15, 16, 17 and 18. The CE-MCTS outperforms EAMC and GCB after 5 generations under different maps and constraints. Moreover, as shown in Fig. 11, CE-MCTS $\geq$ EAMC $\geq$ GCB in the mean values. The maximum coverage of CE-MCTS, EAMC, and GCB is shown in Fig. 12. It shows that CE-MCTS and EAMC are competitive in the maximum values. Figures 13 and 14a–c demonstrate the outcome of CE-MCTS, EAMC, and GCB, respectively. Figures 13d and 14d show the cost and coverage value trajectories of each approach.

These experiments demonstrate that CE-MCTS outperforms EAMC and GCB in the coverage maximization problem.

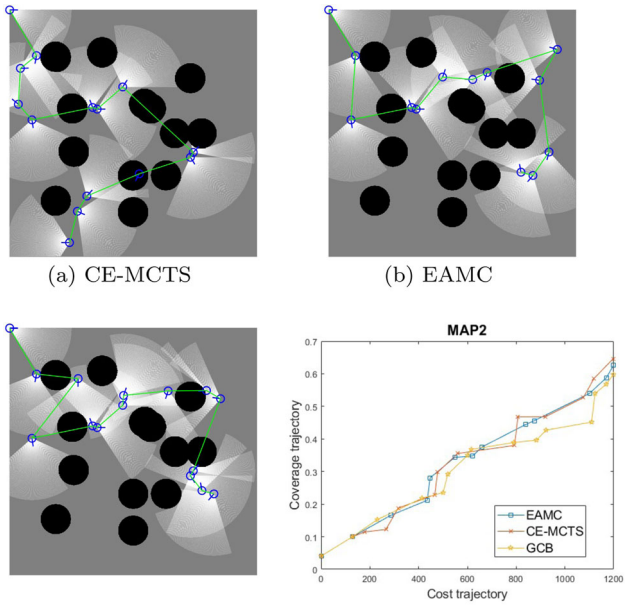


(a) CE-MCTS (b) EAMC
(c) GCB (d) Cost and cover trajectory

Fig. 13 **a–c** Depict the trajectories with the highest coverage value of CE-MCTS, EAMC, and GCB under $B = 1200$ in MAP1. **d** The x-axis and y-axis represent cost and coverage value, respectively

6.3 EX2: search problem

In the EX2, the search performance of three approaches (CE-MCTS, EAMC, and GCB) is compared in the real world. The goal of the drone is to find the target ($P(Z = 1) \geq P_{thr}$) as soon as possible under the cost constraint B. The parameters $B = 20(m)$ and $P_{thr} = 0.95$ are adopted in this experiment. The CE-MCTS, EAMC, and GCB compute the trajectory



(a) CE-MCTS (b) EAMC (c) GCB (d) Cost and cover trajectory

Fig. 14 **a**–**c** Depict the trajectories with the highest coverage value of CE-MCTS, EAMC, and GCB under $B = 1200$ in MAP2. **d** The x-axis and y-axis represent cost and coverage value, respectively

Table 3 The mean of successful search time in EX2

Location	CE-MCTS	EAMC	GCB
#1	**53.63 (s)**	–	63.97
#2	**68.22**	89.20	76.52
#3	84.01	**74.11**	88.23
#4	93.09	**57.99**	109.37
#5	41.21	**37.85**	59.6

Bold values indicate the best performance
The dash line represents the drone does not find the target

offline. At each location, the drone flies 5 times to search for the target with CE-MCTS, EAMC, and GCB, respectively.

As shown in Fig. 19, the expected time till decision ($\mathbb{E}[TTD]$) is compared. GCB is the worst in $\mathbb{E}[TTD+]$ while EAMC is the worst in $\mathbb{E}[TTD-]$. In Table 3, CE-MCTS outperforms EAMC in locations #1 and #2. Moreover, EAMC does not detect the target at location #1 every time (see Fig. 20, Tables 4, 5). Alternatively, EAMC saves time to check location #2. Hence, EAMC outperforms CE-MCTS at locations #3, #4, and #5. As shown in Table 5, CE-MCTS finds the target at each location successfully. In Table 6, CE-MCTS outperforms EAMC and GCB in the mean time of search. These experiments demonstrate CE-MCTS outperforms EAMC and GCB in search problems.

Table 4 The mean of failed search time in EX2

Location	CE-MCTS	EAMC	GCB
#1	–	108.40	–
#2	–	–	–
#3	–	–	–
#4	–	–	–
#5	–	–	107.24

The smaller is the better. The dash line represents that the drone finds the target every time

6.4 EX3: the feasibility of terminal criterion

There are two parts in EX3 to investigate the terminal condition of CE-MCTS. In EX3-1, the goal is to verify if $\bar{b}$ can be adopted as the terminal condition of CE-MCTS. In EX3-2, it is to set the manual parameter T as the terminal criteria for the algorithm.

The EX3-1 and EX3-2 are run once independently to verify CE-MCTS performance under different cost functions (e.g., cardinality and routing). In the cardinality case, there are two budget constraints (e.g., $B = 20$ and $B = 30$). Notice that, in cardinality constraint case, the greedy approach (Fisher et al., 1978) can be viewed as a special case of GCB (Zhang and Vorobeychik, 2016).

In EX3-1, as Fig. 21a, c show, since the cost function is cardinality, obviously $\alpha_c = \alpha_{\hat{c}} = \psi(n) = 1$ and $\kappa_c = 0$. Hence, in Theorem 1, $f(\tilde{X}) = OPT$. Moreover, as $\bar{b} \geq f(\tilde{X})$, $\bar{b}$ is the highest value. As Fig. 21b, d show, the ratios of CE-MCTS to $\bar{b}$ are higher than $\frac{1}{2}(1-\frac{1}{e})$ and lower than 1. Hence, if the threshold $\eta = 0.31$, the CE-MCTS terminates after the first generation. The outcome is over 95% $f(\tilde{X})$ (see Fig. 21b, d). In other words, $\bar{b}$ can be adopted for the terminal criterion.

In the routing case, there are two budget constraints (e.g., $B = 1000$ and $B = 2000$). As Fig. 22a shows, since α_c and $\alpha_{\hat{c}}$ are not close to 1, $\bar{b}$ is not the highest value. Therefore, the percentage of $\bar{b}$ can not be the terminal criterion. Note that, $\alpha_c, \alpha_{\hat{c}} \leq \alpha'_c = 0.7465$ in both two constraints. Due to the limited sampling in Algorithm 3, Fig. 22c shows that $\bar{b}$ is still equal 1. Figure 22b, d show Theorem 4 holds.

The EX3-1 verifies Theorem 4 and Lemma 2 hold. The EX3-1 also demonstrates that when α_c and $\alpha_{\hat{c}}$ are close to 1 (e.g., cardinality), $\bar{b}$ can be adopted for the terminal criterion. Alternatively, the EX3-2 compares CE-MCTS, EAMC and GCB via setting the manual parameter T.

In EX3-2, the results are shown in Tables 7 and 8. The CE-MCTS outperforms EAMC and GCB. The total number of CE-MCTS generations compared to EX1 decreases over 50%. Notice that, the dash line represents the algorithm terminates before the X-th generation, and the star symbol means the GCB samples only one solution. The EX3-2 shows

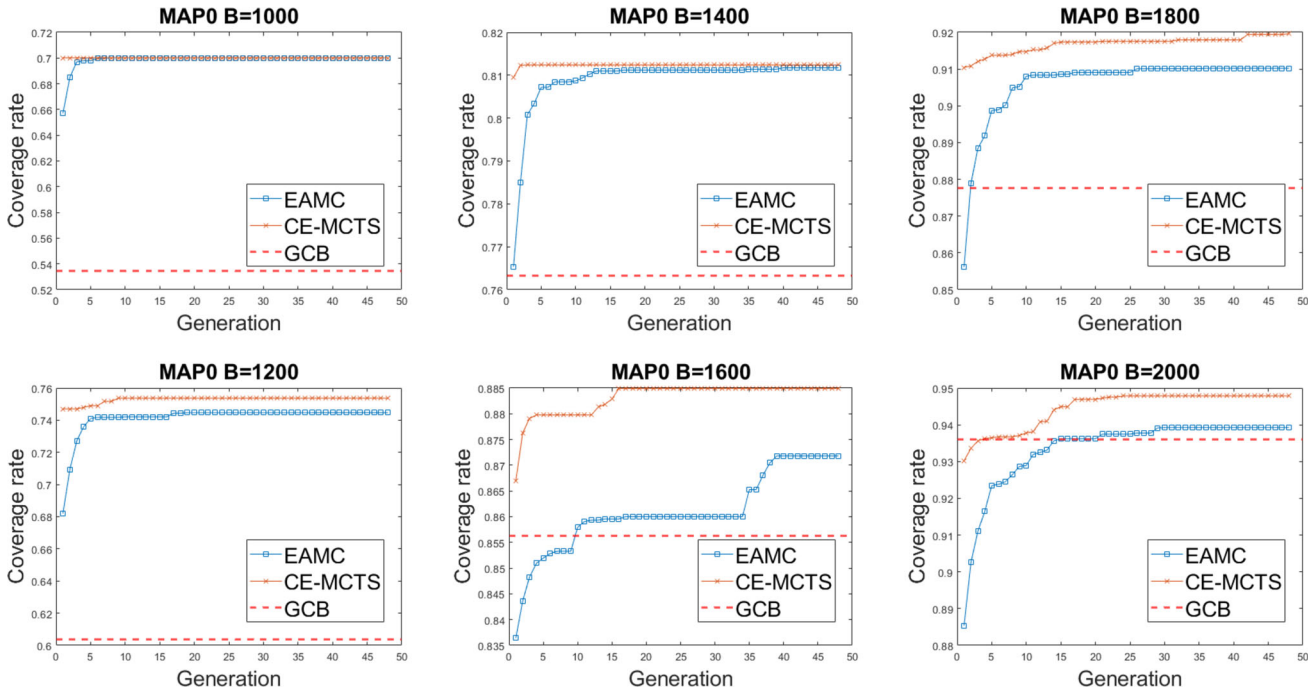


Fig. 15 The performance of CE-MCTS is compared with EAMC and GCB in MAP0 under 1000–2000 routing cost constraints. The x-axis represents the generation number, and total generation size is N. The y-axis represents coverage rate normalized by $f(S)$. **a**–**f** show the results in MAP0

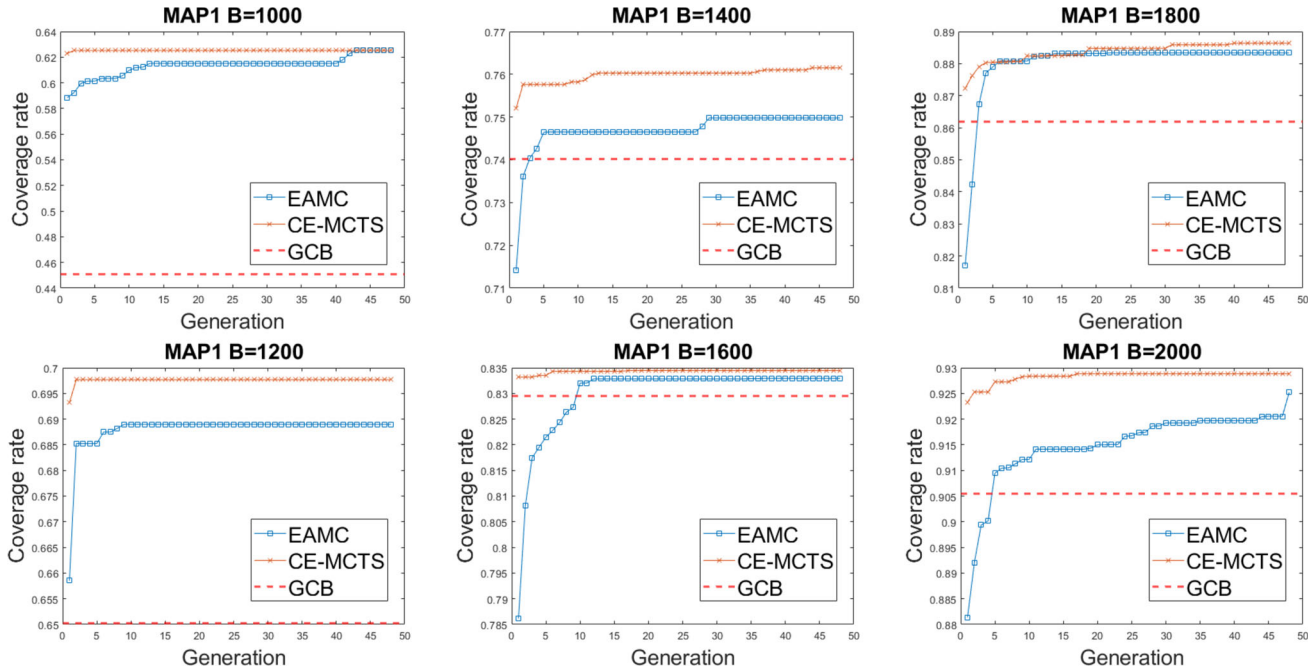


Fig. 16 The performance of CE-MCTS is compared with EAMC, and GCB in MAP1 under 1000–2000 routing cost constraints. The x-axis represents the generation number, and total generation size is N. The y-axis represents coverage rate normalized by $f(S)$. **a**–**f** show the results in MAP1

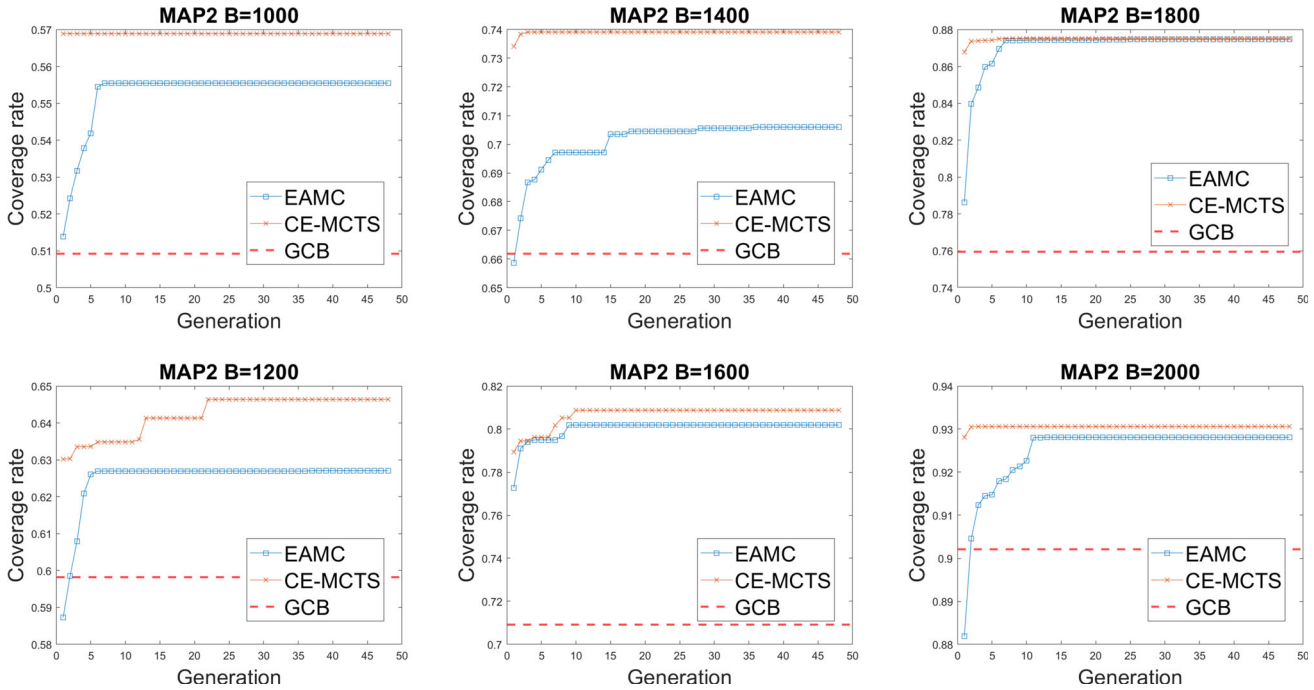


Fig. 17 The x-axis represents the generation number, and total generation size is N. The y-axis represents the normalized coverage rate. **a**–**f** show the results in MAP2

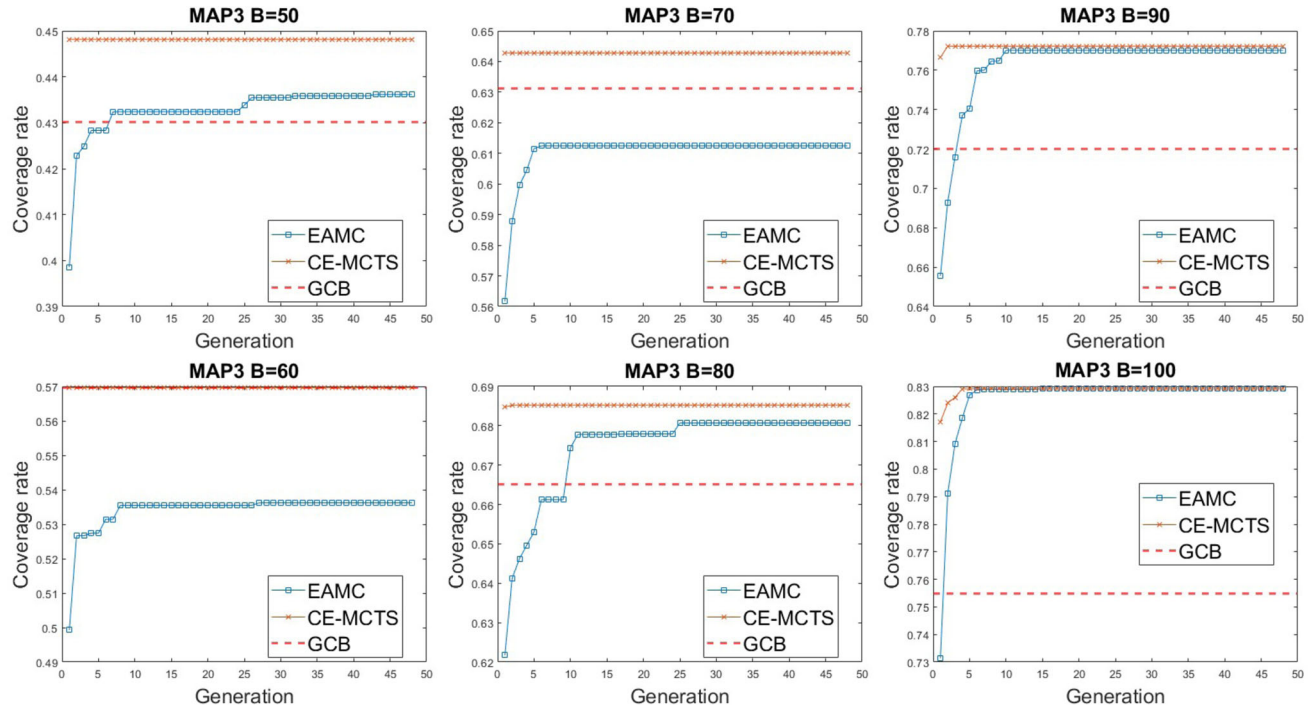


Fig. 18 The performance of CE-MCTS is compared with EAMC, and GCB in MAP3 under 50–100 (m) routing cost constraints. The x-axis represents the generation number, and total generation size is N. The y-axis represents the normalized coverage rate. **a**–**f** show the results in MAP3

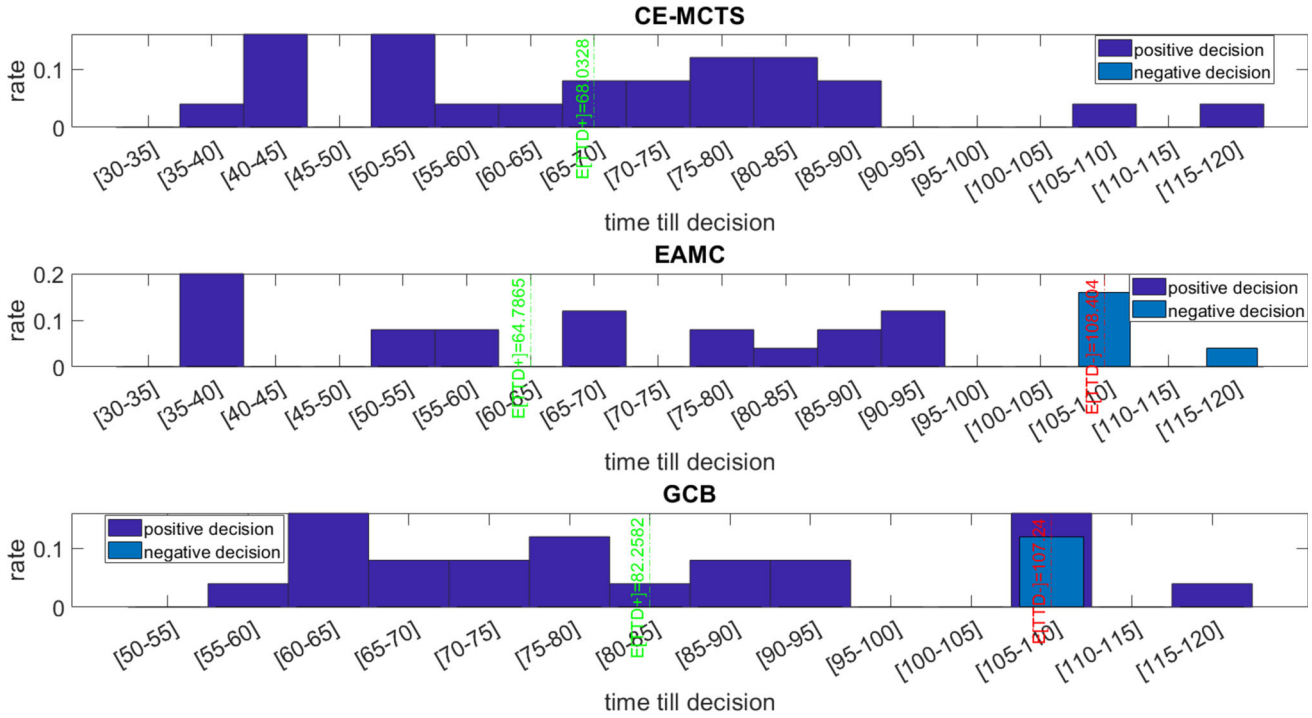
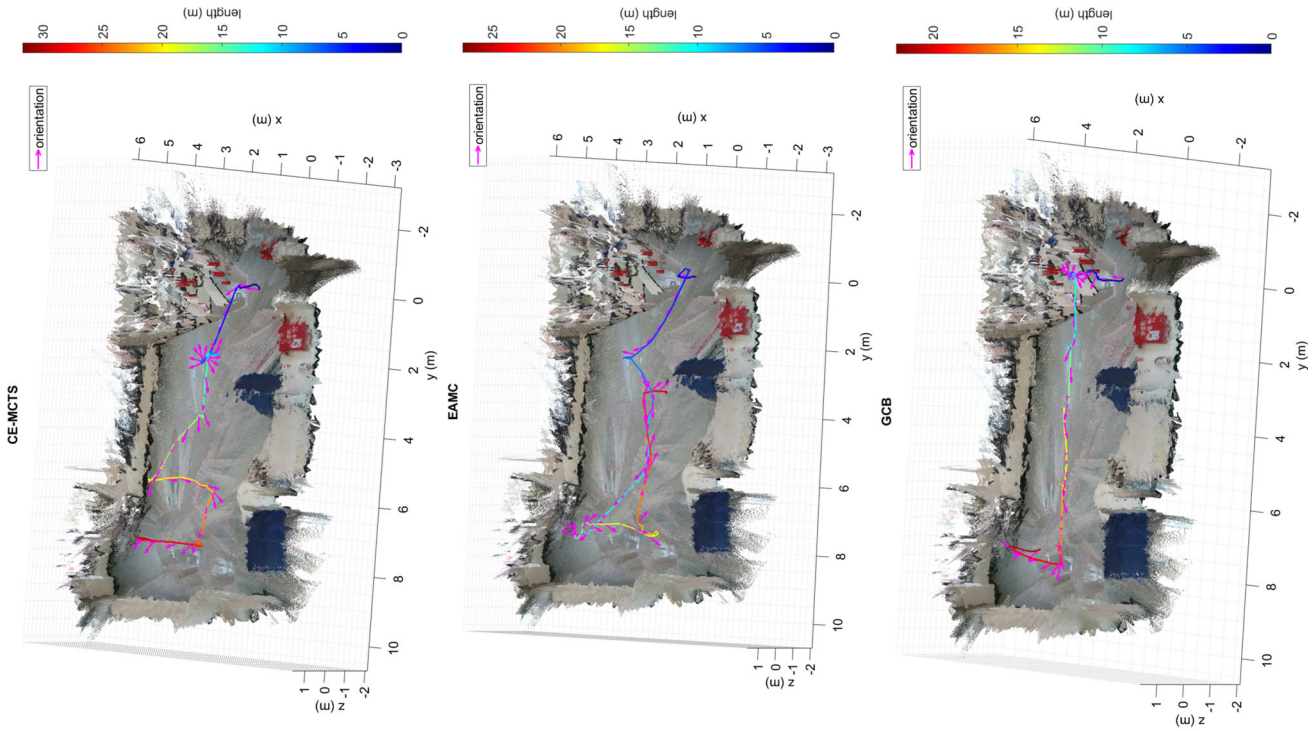


Fig. 19 $\mathbb{E}[TTD]$ of CE-MCTS, EAMC, and GCB. The x-axis and y-axis represent the interval(s) and probability, respectively. The red and green texts show $\mathbb{E}[TTD-]$ and $\mathbb{E}[TTD+]$ for each method (Color figure online)



(a) The flight path of CE-MCTS.

(b) The flight path of EAMC.

(c) The flight path of GCB.

Fig. 20 The three axes represent geographical positions. The color lines show the full trajectory of each method and the color represents the cumulative routing cost responding to the color bar. The pink arrows show the orientation of the drone (Color figure online)

Table 5 The success rate for detection results of EX2

Location	CE-MCTS (%)	EAMC (%)	GCB (%)
#1	100	0	100
#2	100	100	100
#3	100	100	100
#4	100	100	100
#5	100	100	40

The higher is the better

Table 6 The average flight time

	CE-MCTS	EAMC	GCB
Mean (s)	**68.03**	73.51	85.26
Std	21.17	25.24	18.83

Bold value indicates the best performance
The smaller is the better

Table 7 The coverage results of EX3-2 in MAP4 under cardinality case[4]

	Generation	1	10	15	Result
MAP4 B=20					
CE-MCTS	16	0.9232	**0.9431**	**0.9431**	**0.9431**
EAMC	15	0.9113	**0.9431**	**0.9431**	**0.9431**
GCB	*	**0.9250**	–	–	0.9250
MAP4 B=30					
CE-MCTS	15	0.9937	**0.9949**	**0.9949**	**0.9949**
EAMC	15	0.9733	**0.9949**	**0.9949**	**0.9949**
GCB	*	**0.9949**	–	–	0.9949

Bold values indicate the best performance
In $B = 30$ case, the GCB result is 0.99486 lower than the CE-MCTS and EAMC results (0.99492)

Table 8 The coverage results of EX3-2 in MAP4 under routing case

	Generation	1	10	15	Result
MAP4 B=1000					
CE-MCTS	14	**0.6689**	**0.6748**	–	**0.6748**
EAMC	24	0.6010	0.6623	**0.6748**	**0.6748**
GCB	*	0.5186	–	–	0.5186
MAP4 B=2000					
CE-MCTS	19	**0.9311**	**0.9440**	**0.9440**	**0.9440**
EAMC	21	0.9160	0.9434	**0.9440**	**0.9440**
GCB	*	0.8867	–	–	0.8867

Bold values indicate the best performance

that T is a manual parameter for the trade-off between performance and computation time.

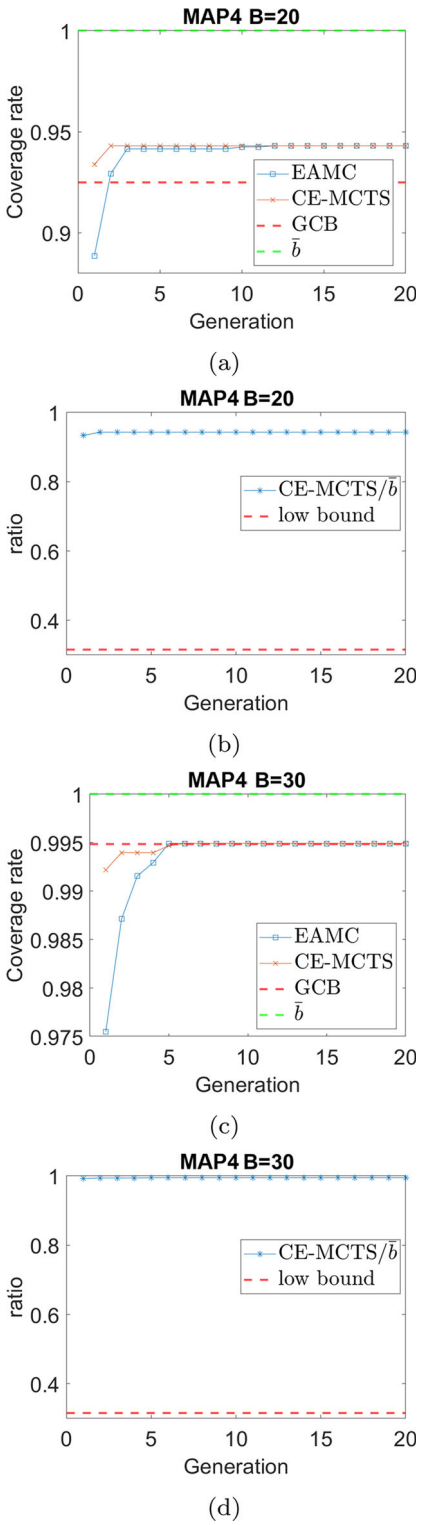


Fig. 21 The cost function is cardinality cost and budgets are 20 and 30. In **a**, **c**, the x-axis and y-axis represent the generation index and coverage rate normalized by $f(S)$, respectively. The green dash line $\bar{b}$ represents estimation $f(X)$, where $\bar{b} \geq f(X)$. In **b**, **d**, the x-axis and y-axis represent the generation index and ratio of CE-MCTS to $\bar{b}$, respectively. The red dash line shows the lower bound of CE-MCTS from Theorem 4 (Color figure online)

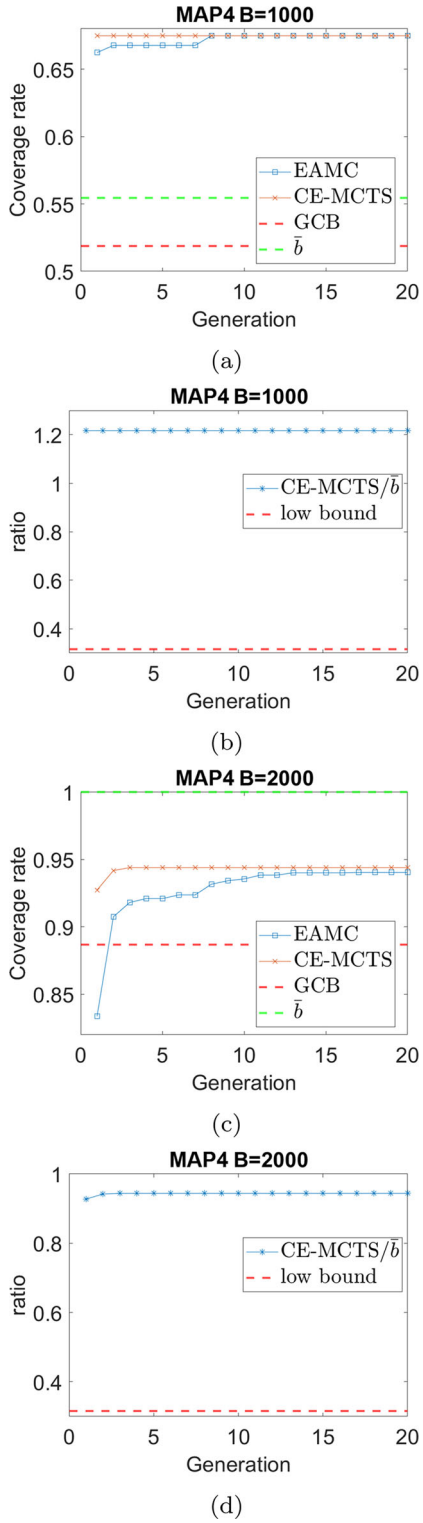


Fig. 22 The cost function is routing cost and budgets are 1000 and 2000. In **a**, **c**, the x-axis and y-axis represent the generation index and coverage rate normalized by $f(S)$, respectively. The green dash line $\bar{b}$ represents estimation $f(\tilde{X})$, where $\bar{b} \geq f(\tilde{X})$. In **b**, **d**, the x-axis and y-axis represent the generation index and ratio of CE-MCTS to $\bar{b}$, respectively. The red dash line shows the lower bound of CE-MCTS from Theorem 4 (Color figure online)

Table 9 The computation time result of EX4 and the unit is seconds

	16	17	18	19	20
MAP0 B=1000					
CE-MCTS	286.2	381.7	469.3	643.9	806.1
EAMC	13.4	14.3	17.7	25.9	32.1
GCB	0.2	0.2	0.2	0.3	0.3

6.5 EX4: the comparison of computation time

In the EX4, the comparison of computation time in the empty map (see Fig. 8a) is as follows: The ground set $|S|$ is 16 to 20 and the routing cost constraint is 1000 units. The parameters are shown in Table 2. The experiments are run once to evaluate the computation time of different approaches under various sizes of $|S|$. Note that, CE-MCTS and EAMC are randomized algorithms sampling $|S|(|S|-1)(|S|-1)$ trajectories, and GCB is a deterministic algorithm, which generates only one trajectory.

As shown in Table 9, GCB $\leq$ EAMC $\leq$ CE-MCTS in computation time due to the difference of computational complexity. The numerical results demonstrate that although the performance of CE-MCTS is better than that of GCB and EAMC, the computational time of CE-MCTS is higher than that of GCB abd EAMC.

7 Conclusions and future work

This research proposes the CE-MCTS approach to solve the maximal coverage problem with routing constraints. The CE-MCTS is a sampling-based method consisting of CEM and MCTS. The experiments show that the proposed CE-MCTS algorithms outperform the benchmark approaches (e.g., EAMC and GCB). Furthermore, the proposed TE algorithm gives the terminal criterion of sampling-based methods under certain conditions.

The future work of this research is as follows: First, CE-MCTS adopts CEM for selecting subgoals with higher coverage. CEM is also applied to the TSP problem (De Boer et al., 2005). Considering both maximal coverage and TSP problems could explore new solutions for these problems. Second, CE-MCTS adopts the EAMC approach for theoretical guarantees in the sampling part. However, the theoretical analysis for the expanding strategy in CE-MCTS is not investigated. If it is explored, there could be a theoretical bound over $\frac{\alpha_f}{2}(1-\frac{1}{e^{\alpha_f}})\widetilde{OPT}$. Finally, in map exploration problems, the coverage functions are unknown without an oracle function to be queried. A promising approach is to utilize

the invariant properties of coverage functions in the Fourier domain (Tseng, 2021).

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s10514-024-10156-6.

Author Contributions Conceptualization, P-TL and K-ST; methodology, P-TL and K-ST; software, P-TL and K-ST; validation, P-TL and K-ST; formal analysis, P-TL and K-ST; investigation, P-TL and K-ST; resources, P-TL and K-ST; data collection, P-TL; writing–original draft preparation, P-TL; writing–review and editing, K-ST; visualization, P-TL; supervision, K-ST; project administration, K-ST; funding acquisition, K-ST.

Funding This research was supported by Taiwan Swarm Innovation Inc., Taiwan MOST Grant 108-2221-E-008-074-MY3, 111-2221-903 E-008-097, and NSTC 112-2221-E-008-075.

Data Availability Data sharing not applicable to this article as no datasets were generated or analysed during the current study.

Declarations

Ethics approval The research does not involve human participants, their data or biological material and it does not involve animals.

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

Appendix A: Mathematical formulation of CEM

The mathematical formulation of CEM is as follows: First, the probability l representing $f(x)$ is greater than or equal to the threshold (γ). When l is small (e.g., $l \leq 10^{-5}$), it is called a rare event. Mathematically, l can be expressed as

$$l = \mathbb{P}_u(f(\mathbf{X}) \geq \gamma) = \mathbb{E}_u I_{f(\mathbf{X})\geq\gamma}, \tag{15}$$

where I is an indicator function and x is sampled by the density $h(\cdot; u)$ with the parameter u. Alternatively, l can be derived as

$$l = \int I_{f(x)\geq\gamma} \frac{h(x;u)}{g(x)} g(x)\, dx = \mathbb{E}_g I_{f(\mathbf{X})\geq\gamma} \frac{h(\mathbf{X};u)}{g(\mathbf{X})}, \tag{16}$$

where g is another density. Notice that, in Eq. 16, the expectation is taken by g. An unbiased estimator of l is

$$\hat{l} = \frac{1}{N}\sum_{i=1}^{N} I_{f(x_i)\geq\gamma} \frac{h(x_i;u)}{g(x_i)}. \tag{17}$$

Second, density estimation of l is

$$g^*(x) := \frac{I_{f(x)\geq\gamma} h(x;u)}{l}. \tag{18}$$

However, l is unknown. The direct way is to select g from the family of densities $h(\cdot; v)$.

Third, a measurement between two density functions g and h is the $Kullback-Leibler\ distance$ (KLD) (so called the cross-entropy between g and h). The KLD is defined as:

$$\mathcal{D}(g,h) = \mathbb{E}_g \ln \frac{g(\mathbf{X})}{h(\mathbf{X})} = \int g(x)\ln g(x)\, dx - \int g(x)\ln h(x)\, dx. \tag{19}$$

Hence, minimizing KLD between g^* in Eq. 18 and $h(\cdot; v)$ is equal to maximize $\int g^*(x)\ln h(x; v)\, dx$. Replacing $g^*(x)$ with Eq. 18, it is equivalent to

$$\max_v \int \frac{I_{f(x)\geq\gamma} h(x;u)}{l} \ln h(x;v)\, dx = \max_v \mathbb{E}_u I_{f(x)\geq\gamma} \ln h(\mathbf{X}; v). \tag{20}$$

Finally, v can be obtained by solving Eq. 20.

References

Amigoni, F., & Gallo, A. (2005). A multi-objective exploration strategy for mobile robots. In *IEEE international conference on robotics and automation* (pp. 3850–3855).

Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine Learning, 47*(2), 235–256.

Bian, C., Feng, C., Qian, C., & Yu, Y. (2020). An efficient evolutionary algorithm for subset selection with general cost constraints. *Proceedings of the AAAI Conference on Artificial Intelligence, 34*(4), 3267–3274.

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

Bircher, A., Kamel, M., Alexis, K., Oleynikova, H., & Siegwart, R. (2018). Receding horizon path planning for 3d exploration and surface inspection. *Autonomous Robots, 42*(2), 291–306.

Brock, O., Trinkle, J., & Ramos, F. (2009). Proofs and experiments in scalable, near-optimal search by multiple robots. Robotics: Science and Systems IV, pp. 206–213.

Cao, Z. L., Huang, Y., & Hall, E. L. (1988). Region filling operations with random obstacle avoidance for mobile robots. *Journal of Robotic systems, 5*(2), 87–102.

Chaslot, G., Bakkes, S., Szita, I., & Spronck, P. (2008). Monte-Carlo tree search: A new framework for game AI. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 4*(1), 216–217.

Cheng, P. Keller, J., & Kumar, V. (2008). Time-optimal UAV trajectory planning for 3d urban structure coverage. In *IEEE/RSJ international conference on intelligent robots and systems* (pp. 2750–2757).

Cieslewski, T., Kaufmann, E., & Scaramuzza, D. (2017). Rapid exploration with multi-rotors: A frontier selection method for high speed flight. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 2135–2142).

Conforti, M., & Cornuéjols, G. (1984). Submodular set functions, matroids and the greedy algorithm: Tight worst-case bounds and some generalizations of the Rado-Edmonds theorem. *Discrete Applied Mathematics, 7*(3), 251–274.

Connolly, C. (1985). The determination of next best views. *IEEE International Conference on Robotics and Automation, 2*, 432–435.

Coquelin, P. -A., & Munos, R. (2007). Bandit algorithms for tree search. arXiv:0703-062 [CS].

Corah, M., & Michael, N. (2019). Distributed matroid-constrained submodular maximization for multi-robot exploration: Theory and practice. *Autonomous Robots, 43*(2), 485–501.

Costa, A., Jones, O. D., & Kroese, D. (2007). Convergence properties of the cross-entropy method for discrete optimization. *Operations Research Letters, 35*(5), 573–580.

Coulom, R. (2006). Efficient selectivity and backup operators in Monte-Carlo tree search. In *International conference on computers and games* (pp. 72–83).

De Boer, P.-T., Kroese, D. P., Mannor, S., & Rubinstein, R. Y. (2005). A tutorial on the cross-entropy method. *Annals of Operations Research, 134*(1), 19–67.

Deng, D., Duan, R., Liu, J., Sheng, K., & Shimada, K. (2020). Robotic exploration of unknown 2d environment using a frontier-based automatic-differentiable information gain measure. In *IEEE/ASME international conference on advanced intelligent mechatronics (AIM)* (pp. 1497–1503).

Deng, D., Xu, Z., Zhao, W., & Shimada, K. (2020). Frontier-based automatic-differentiable information gain measure for robotic exploration of unknown 3d environments. arXiv:2011.05288.

Englot, B., & Hover, F. (2012). Sampling-based coverage path planning for inspection of complex structures. *Proceedings of the International Conference on Automated Planning and Scheduling, 22*, 29–37.

Feige, U. (1998). A threshold of ln n for approximating set cover. *Journal of Applied and Computational Mechanics (JACM), 45*(4), 634–652.

Fisher, M. L., Nemhauser, G. L., & Wolsey, L. A. (1978). An analysis of approximations for maximizing submodular set functions—II. Polyhedral Combinatorics (pp. 73–87).

Friedrich, T., & Neumann, F. (2015). Maximizing submodular functions under matroid constraints by evolutionary algorithms. *Evolutionary Computation, 23*(4), 543–558.

Galceran, E., & Carreras, M. (2023). A survey on coverage path planning for robotics. *Robotics and Autonomous Systems, 61*, 1258–1276.

González-Banos, H. H., & Latombe, J.-C. (2002). Navigation strategies for exploring indoor environments. *The International Journal of Robotics Research, 21*(10–11), 829–848.

Grossman, T., & Wool, A. (1997). Computational experience with approximation algorithms for the set covering problem. *European Journal of Operational Research, 101*(1), 81–92.

Guillaume, M., Winands, M. H., Szita, I., & van den Herik, H. J. (2008). Cross-entropy for Monte-Carlo tree search. *International Computer Games Association, 31*(3), 145–156.

James, S., Konidaris, G., & Rosman, B. (2017). An analysis of Monte Carlo tree search. *AAAI Conference on Artificial Intelligence, 31*, 3576–3582.

Khuller, S., Moss, A., & Naor, J. S. (1999). The budgeted maximum coverage problem. *Information Processing Letters, 70*(1), 39–45.

Kocsis, L., & Szepesvári, C. (2006). Bandit based Monte-Carlo planning. In *European conference on machine learning* (pp. 282–293).

Krause, A., & Guestrin, C. (2005). A note on the budgeted maximization of submodular functions.

Krause, A., & Guestrin, C. (2007). Near-optimal observation selection using submodular functions. *Conference on Artificial Intelligence (AAAI) Nectar track, 7*, 1650–1654.

Krause, A., Guestrin, C., Gupta, A., & Kleinberg, J. (2006). Near-optimal sensor placements: Maximizing information while minimizing communication cost. In *Proceedings of the 5th international conference on Information processing in sensor networks* (pp. 2–10).

Lanillos, P., Besada-Portas, E., Pajares, G., & Ruz, J. J. (2012). Minimum time search for lost targets using cross entropy optimization. In *IEEE/RSJ international conference on intelligent robots and systems* (pp. 602–609).

Lin, S., & Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations Research, 21*(2), 498–516.

Lu, B. -X., & Tseng, K. -S. (2020). 3d map exploration via learning submodular functions in the Fourier domain. In *International conference on unmanned aircraft systems (ICUAS)* (pp. 1199–1205).

Lu, B.-X., & Tseng, K.-S. (2022). 3d map exploration using topological Fourier sparse set. *Journal of Intelligent and Robotic Systems, 104*, 75.

Luperto, M., Antonazzi, M., Amigoni, F., & Borghese, N. A. (2020). Robot exploration of indoor environments using incomplete and inaccurate prior knowledge. *Robotics and Autonomous Systems, 133*, 103622.

Moerland, T. M., Broekens, J., Plaat, A., & Jonker, C. M. (2020). The second type of uncertainty in Monte Carlo tree search. arXiv:2005.09645.

Nemhauser, G. L., Wolsey, L. A., & Fisher, M. L. (1978). An analysis of approximations for maximizing submodular set functions—I. *Mathematical Programming, 14*(1), 265–294.

Qian, C., Shi, J.-C., Yu, Y., & Tang, K. (2017). On subset selection with general cost constraints. *International Joint Conference on Artificial Intelligence, 17*, 2613–2619.

Qian, C., Yu, Y., & Zhou, Z.-H. (2015). Subset selection by pareto optimization. In *Advances in neural information processing systems* (pp. 1774–1782).

Roostapour, V., Neumann, A., Neumann, F., & Friedrich, T. (2019). Pareto optimization for subset selection with dynamic cost constraints. *Proceedings of the AAAI Conference on Artificial Intelligence, 33*(01), 2354–2361.

Rubinstein, R. (1999). The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability, 1*(2), 127–190.

Schadd, M. P., Winands, M. H., Tak, M. J., & Uiterwijk, J. W. (2012). Single-player Monte-Carlo tree search for SameGame. *Knowledge-Based Systems, 34*, 3–11.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature, 529*(7587), 484–489.

Singh, A., Krause, A., Guestrin, C., Kaiser, W. J., & Batalin, M. A. (2006). Efficient planning of informative paths for multiple robots. In *International joint conference on artificial intelligence (IJCAI)* (pp. 2204–2211).

Sviridenko, M. (2004). A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters, 32*(1), 41–43.

Tseng, K.-S. (2021). Transfer learning of coverage functions via invariant properties in the Fourier domain. *Autonomous Robots, 45*(4), 519–542.

Tseng, K.-S., & Mettler, B. (2017). Near-optimal probabilistic search via submodularity and sparse regression. *Autonomous Robots, 41*(1), 205–229.

Tseng, K.-S., & Mettler, B. (2018). Near-optimal probabilistic search using spatial Fourier sparse set. *Autonomous Robots, 42*(2), 329–351.

Umari, H., & Mukhopadhyay, S. (2017). Autonomous robotic exploration based on multiple rapidly-exploring randomized trees. In

IEEE/RSJ international conference on intelligent robots and systems (IROS) (pp. 1396–1402).
Xiao, C., Huang, R., Mei, J., Schuurmans, D., & Müller, M. (2019). Maximum entropy Monte-Carlo planning. *Advances in Neural Information Processing Systems, 32*, 9520–9528.
Yamauchi, B. (1997). A frontier-based approach for autonomous exploration. In *IEEE international symposium on computational intelligence in robotics and automation (CIRA)* (pp. 146–151).
Yasutomi, F., Yamada, M., & Tsukamoto, K. (1988). Cleaning robot control. In *Proceedings. IEEE international conference on robotics and automation* (pp. 1839–1841).
Zhang, H., & Vorobeychik, Y. (2016). Submodular optimization with routing constraints. *Proceedings of the AAAI Conference on Artificial Intelligence, 30*(1), 819–825.
Zhou, B., Zhang, Y., Chen, X., & Shen, S. (2021). Fuel: Fast UAV exploration using incremental frontier structure and hierarchical planning. *IEEE Robotics and Automation Letters, 6*(2), 779–786.

Pao-Te Lin is a graduate student in the Department of Mathematics, National Central University, Taiwan. He received his B.S. degree in Mathematics from National University of Kaohsiung, Taiwan, in 2020. His current research interests include map exploration and Monte Carlo tree search.

Kuo-Shih Tseng is an assistant professor in the Department of Mathematics, National Central University, Taiwan. He received his B.S. degree in Mechanical Engineering from Chung Yuan Christian University, Taiwan, in 2002, and his M.S. degree in Bio-Industrial Mechatronics Engineering from National Taiwan University, Taiwan, in 2004. He received his second M.S. degree and Ph.D. in the Computer Science and Engineering from University of Minnesota in 2013 and 2016, respectively. From 2004 to 2009, he was an associated researcher with the Intelligent Robotics Technology Division, Mechanical and System Research Laboratories, Industrial Technology Research Institute (ITRI), Hsinchu, Taiwan. From 2009 to 2011, he was a researcher with KYH Co., Ltd., Taiwan. His current research interests include robotic search, human search behavior analysis, and reinforcement learning.