2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Perturbation-Based Best Arm Identification for Efficient Task Planning with Monte-Carlo Tree Search

Daejong Jin, Juhan Park, and Kyungjae Lee

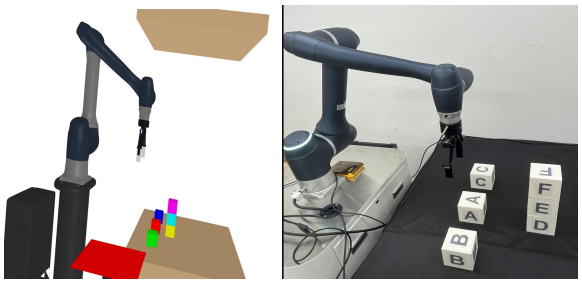
***Abstract*— Combining task and motion planning (TAMP) is crucial for intelligent robots to perform complex and long-horizon tasks. In TAMP, many approaches generally employ Monte-Carlo tree search (MCTS) with upper confidence bound (UCB) for task planning to handle exploration-exploitation trade-off and find globally optimal solutions. However, since UCB basically considers the estimation error caused by noise, the error caused by insufficient optimization of the sub-tree is not represented. Hence, UCB-based approaches have the disadvantage of not exploring underestimated sub-trees. To alleviate this issue, we propose a novel tree search method using perturbation-based best-arm identification (PBAI). We theoretically prove the bound of the simple regret of our method and empirically verify that PBAI finds the optimal task plans faster and more efficiently than the existing algorithms. The source code of our proposed algorithm is available at https://github.com/jdj2261/pytamp.**

I. Introduction

Robots that perform complex and long-horizon tasks require the ability to plan high-level tasks by considering low-level motion plans, which is known as a problem of task and motion planning (TAMP) [1], [2]. Task planning is required to obtain efficient long-horizon plans, such as a sequence of possible robotic operations like *pick*, *place*, or *push* in manipulation domain. Motion planning is also needed to obtain actual low-level movements to perform a task plan. Most standard approaches have integrated a task planner [3]–[5] with a motion planner [6].

With the recent development of efficient motion planners [6], [7], task planning has become a more crucial issue in solving long-horizon planning problems. The main challenge of task planning is the exponentially increasing computational complexity. For example, in a sequential manipulation scenario, as shown in Fig. 1, as the number of interacting objects increases, the possible sequences of operations exponentially grow. For instance, for *pick* motion, the possible combinations increase proportion to the number of objects, and for *place* motion, similarly, the possible choices also increase linearly to the number of objects. In this regard, developing efficient task planners is a crucial issue in TAMP.

To overcome these issues, we focus on developing an efficient Monte-Carlo tree search (MCTS) algorithm for task-level planning. Most approaches [8]–[10] have often utilized the upper confidence bound (UCB) [11], [12] for task-level planning, however, we argue that the confidence bound used in existing methods is not efficient for task-level optimization. In order for the UCT to successfully optimize a task plan, the confidence bound must accurately represent the solution's error caused by two sources: estimation error of the objective function and optimization error of the sub-tree. The former is usually caused by randomness in the problem, such as noisy observations. The latter is caused by insufficient optimization of sub-trees. In general, the confidence bound used in the UCT accurately reflects the former error, not the latter error.



(a) Initial state in simulation (b) Initial state in real world

Fig. 1: Benchmark 1 (Blocks World). A robot should stack alphabetical boxes in one place in alphabetical order taking into account the obstacles and the constraints of the robot.

In this paper, we propose a novel perturbation-based search method for MCTS to consider both estimation and optimization errors. In the existing UCT, the optimization error was not sufficiently considered. Hence, we added the concept of random perturbation that can make underestimated sub-trees to be explored randomly. To apply our algorithm to task planning, we define symbolic states, actions from a given scene, and rewards received after the robot performs each step. Notably, we present a level-wise TAMP framework, similarly to [13] where the proposed MCTS optimizes a task plan.

The main contribution of this paper is twofold: We propose a perturbation-based exploration for MCTS and prove that the convergence speed is $O(T\exp(-T))$ where T is the number of iterations. We would like to emphasize that convergence guarantee is an essential factor of a planning algorithm. Second, we demonstrate our method in four benchmarks which cannot be solved trivially. We verify that the proposed method nearly outperforms other task planners with a large margin regarding the convergence speed and the final sum of rewards.

Daejong Jin, Juhan Park, and Kyungjae Lee are with the Department of Artificial Intelligence, Chung-Ang University, Seoul 06974, Republic of Korea (e-mails: wlseoeo@gmail.com, p3549823@gmail.com, kyungjae.lee@ai.cau.ac.kr)

This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2022-0-00331, Development of emotion recognition/generation-based interacting edge device technology for mental health care, 50%) and (No. 2021-0-01341, AI Graduate School Program, CAU, 50%).

II. Related Work

A. Search Algorithms for Task Planning

One of the major strategies [14]–[16] for task planning has conventionally taken a tree search algorithm. Traditional search algorithms such as Breadth-First search (BFS) or Depth-First search (DFS) iteratively expand a node until a goal node is found. It provides the optimal solution to the problem but is quite exhaustive. In this regard, the computational cost of brute force search increases exponentially as the tree depth and the branching factor increase. Branch-and-bound [17], [18] alleviates this issue. This algorithm is widely used to solve NP-hard discrete optimization problems. The technique is based on the principle of dividing the set of feasible solutions into smaller subsets and evaluating them repeatedly until the best solution is found. For smaller problems, an optimal solution can be found within a given problem. As the problem size increases, the tree may grow to a point where processing times become longer due to the increased number of nodes. A* algorithm [19] searches for the minimum cost path between nodes and additionally employs a heuristic function value (h-value) for each node to predict the future cost, providing an efficient solution. But It can take a long time in case of a complex cost function.

B. Monte-Carlo Tree Search

Many TAMP approaches have employed the MCTS as a task-level planner [8]–[10], [20], [21]. A goal of MCTS is to find the optimal sequence of actions using a random sampling technique in the discrete action space. The main challenge in MCTS is to control the trade-off between exploration (select unvisited actions) and exploitation (select the known-best actions). To control the exploration-exploitation trade-off, the upper confidence bound for tree (UCT) algorithm [12] has employed the principle of optimism in the face of uncertainty as an exploration method. The confidence bound represents the potential to find a better value that may be underestimated by an estimation error. The UCT explores action space by considering estimated values and their confidence bounds. However, considering the trade-off is inefficient from the perspective of optimization. In optimization, the quality of the final solution found as a result of the optimization is the most crucial factor instead of controlling the trade-off during optimization. In this regard, the best arm identification (BAI) framework [22] is often used to focus on finding the optimal action. BAI-UCB [23] have first applied BAI framework to the UCT and guarantee the faster convergence than the original UCT. However, we would like to emphasize that the confidence bound used in UCT and BAI-UCB is derived from Hoeffding inequality, which mainly considers the estimation errors instead of the optimization error of the sub-tree. In this paper, we overcome this issue by adding a random perturbation.

III. Problem Formulation

In this section, we formulate a problem of task and motion planning (TAMP). To represent a task-level plan, we introduce a symbolic state $m \in \mathcal{M}$ and symbolic action $u \in \mathcal{U}$ where $\mathcal{M}$ is a set of symbolic states and $\mathcal{U}$ is a set of symbolic actions. A symbolic state indicates a logical mode of the system, and a symbolic action indicates changing the logical mode to another logical mode. For example, a symbolic state expresses the discrete state of the system such as *the object $\mathcal{O}_i$ is on the object $\mathcal{O}_j$* or *a robot is holding the object O_i*. After taking a symbolic action u, a symbolic state m will be moved to another symbolic state $m' = \mathcal{T}(m, u)$ where $\mathcal{T}(m, u)$ is a transition function. For example, a symbolic action expresses the change of dynamics or kinematics of the system such as *grasping the object O_i* or *placing the object O_i on the object O_j* is a symbolic action. Furthermore, another goal of TAMP is to find motion plans that are a sequence of continuous states $x \in \mathcal{X}$, where x represents the robot's configurations and its surroundings, such as objects in object manipulation.

Then, the problem of TAMP is to find a sequence of symbolic state and action pairs and corresponding configuration trajectories from initial configuration x_0 and initial state m_0, and can be formulated as a form of a Logic-Geometric Program (LGP) [13], [24] as follows,

$$\begin{aligned} \max_{K, \left\{u_k,\, x(T_{k-1}:T_k)\right\}_{k=1}^{K}} & \sum_{k=1}^{K} \left[\int_{T_{k-1}}^{T_k} r_{\mathrm{p}}(x(t))dt + r_{\mathrm{f}}(x(T_k)) \right] \\ \text{s. t.} \quad & K \leq H, x(0) = x_0, T_k - T_{k-1} \leq T \\ \forall k \in [1:K] \quad & x(T_k) \in \mathcal{C}_{\mathrm{switch}}(m_{k-1}, u_k), \\ \forall k \in [1:K] \quad & m_k = \mathcal{T}\left(m_{k-1}, u_k\right), \\ \forall t \in [T_{k-1}:T_k) \quad & x(t) \in \mathcal{C}_{\mathrm{path}}(m_{k-1}), \end{aligned} \tag{1}$$

where H is the maximum length of symbolic transitions, T indicates the maximum length of a robot trajectory per symbolic action, $\mathcal{C}_{\mathrm{path}}(m_{k-1})$ and $\mathcal{C}_{\mathrm{switch}}(m_{k-1}, u_k)$ are a set of feasible configurations depending a symbolic state and action pair, $r_{\mathrm{p}}(x)$ is a reward function of a trajectory of configuration, and $r_{\mathrm{f}}(x)$ is a reward function of a final configuration at the end of the trajectory.

A. Overview

We address the problem (1) by combining a Monte-Carlo tree search (MCTS) with a geometric motion planner. First, we separate optimization variables into two groups. The first group consists of $(m_{k-1}, u_k, x(T_k))$. The second group is a trajectory of configuration $x(T_{k-1} : T_k)$. Note that if for all k, first groups are given, then the second groups can be simply optimized by using a conventional geometric motion planner such as a sample-based planner [25]–[27], or trajectory optimization methods [28], [29].

Given the geometric motion planner, we reformulate the original problem into the Markov decision problem. First, we define a state as $s_k := (m_k, x(T_k))$ and define an action as $a_k := (u_k, x(T_{k+1}))$. Based on these definitions, we can further define a deterministic transition function as $\mathcal{P}(s_k, a_k) := (\mathcal{T}(m_k, u_k), x(T_{k+1}))$. The original problem can be reduced to the following high-level planning problem,

$$\max_{K, a_{1:K}} \sum_{k=1}^{K} \bar{R}(s_k, a_k, \mathcal{P}\left(s_k, a_k\right)) \text{ s. t. } K \leq H, \tag{2}$$

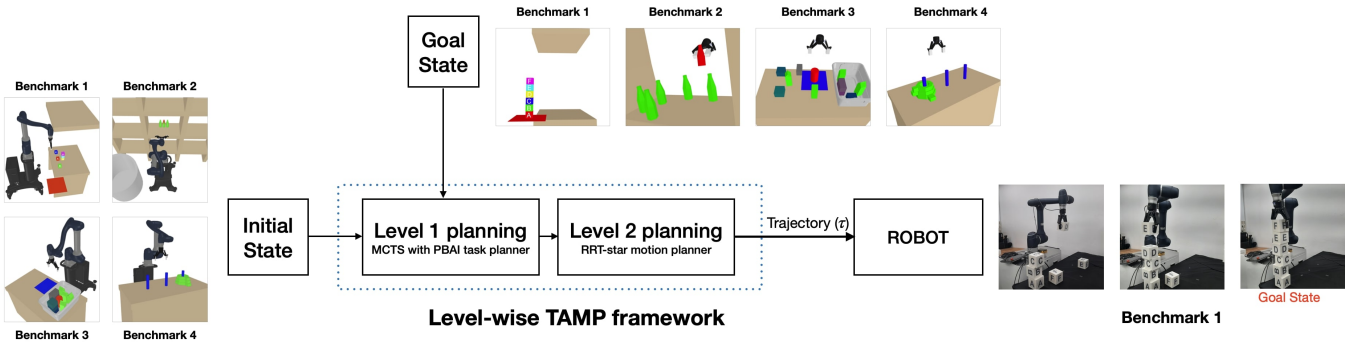


Fig. 2: Overview of our proposed level-wise TAMP framework. A robot moves along the optimized plans generated by our framework when given a task goal in an initial state.

where $\bar{R}(s_k, a_k, s_{k+1}) := \int_{T_{k-1}}^{T_k} r_{\mathrm{p}}(x(t))dt + r_{\mathrm{f}}(x(T_k))$ and $s_{k+1} := \mathcal{P}(s_k, a_k)$. Consequently, we propose a novel exploration method for MCTS to optimize high-level planning. Hence, the entire optimization is separated into two levels. In the first level, high-level plans are computed by optimizing $r(s_k, a_k, s_{k+1}) := r_{\mathrm{f}}(x(T_k))$ with the proposed MCTS and it returns candidates of high-level plans. In the second level, we check the feasibility and optimize continuous trajectories for given high-level plans by considering $\bar{R}(s_k, a_k, s_{k+1})$. We present an overview of our proposed algorithm in Fig. 2, and provide detailed explanations in Section IV-C.

IV. Perturbation-based Best-Arm Identification for Monte-Carlo Tree Search

In this section, we propose a perturbation-based best-arm identification (PBAI) method. The best-arm identification (BAI) has often been applied to a multi-armed bandit [30], and tree search problems [23]. The BAI focuses on finding the best action after finishing all iterations instead of minimizing cumulative regrets during the optimization procedure. This property of BAI makes it suitable for TAMP problems since finding feasible high-level plans is more important than not losing many rewards during the optimization. Especially, we proposed a novel BAI method by using a random perturbation, which will be explained in Section IV-B. Note that the proposed method is the first approach to employ the perturbation method for both BAI and TAMP problems.

A. Value Estimation

The goal of MCTS is to find a sequence of optimal actions that have the optimal values defined by the optimal Bellman equation as $Q_\star^{(k)}(s,a) := \bar{r}(s,a) + V_\star^{(k+1)}(\mathcal{P}(s,a))$ and $V_\star^{(k)}(s) := \max_{a\in\mathcal{A}} Q_\star^{(k)}(s,a)$, where $\bar{r}(s,a) := r(s,a,\mathcal{P}(s,a))$ and k indicates the kth step (or depth k in a search tree). In MCTS, we estimate $Q_\star^{(k)}$ and $V_\star^{(k)}$ using a Monte-Carlo simulation. For each node in a tree, we estimate the rewards as $\hat{R}(s,a) := \sum_{i=1}^{n^{(k)}(s,a)} R_i / n^{(k)}(s,a)$ where $n^{(k)}(s,a)$ is the number of the times (s,a) has been visited at depth k and R_i is reward at the ith visit. Note that while we address a deterministic reward in this paper, we propose the most general case, including the noisy reward setting. Then, by using the estimated rewards, estimated values can be computed by $\hat{Q}^{(k)}(s,a) := \hat{R}(s,a) + \hat{V}^{(k+1)}(\mathcal{P}(s,a))$ and $\hat{V}^{(k)}(s) := \max_{a\in\mathcal{A}} \hat{Q}^{(k)}(s,a)$. From this estimation rule, we can observe a source of the estimation error of $\hat{Q}^{(k)}(s,a)$. The error of $\hat{V}^{(k+1)}$ will be propagated to $\hat{Q}^{(k)}$. Hence, if $\hat{V}^{(k+1)}$ is insufficiently optimized, $\hat{Q}^{(k)}$ can be underestimated. The proposed method alleviates this issue by using a random perturbation.

Algorithm 1: Perturbed Best Arm Identification (PBAI)

1: Input: $\{\hat{Q}(s,a)\}_{a\in\mathcal{A}}$, $\{n^{(k)}(s,a)\}_{a\in\mathcal{A}}$
2: Output: $I^{(k)}(s)$
3: Sample perturbations $g^{(k)}(s,a)$ for each action $a \in \mathcal{A}$
4: Compute $B^{(k)}(s,a)$ for each action $a \in \mathcal{A}$
5: Compute $b^{(k)}(s) = \arg\min_a B^{(k)}(s,a)$ and $u^{(k)}(s) = \arg\max_{a\neq b^{(k)}(s)} U^{(k)}(s,a)$
6: Select $I^{(k)}(s) = \arg\max_{a\in\{b^{(k)}(s), u^{(k)}(s)\}} \beta^{(k)}(s,a)$

B. Action Selection

For an exploration strategy, we propose the perturbation-based best-action identification. For each state node s at depth k, we select the action a to be explored. Then, designing the action selection rule is the key idea for the efficient exploration of MCTS. To choose the action to be explored, we first compute the lower and upper bounds of the value estimation as follows,

$$U^{(k)}(s,a) := \hat{Q}^{(k)}(s,a) + \beta^{(k)}(s,a) \tag{3}$$
$$L^{(k)}(s,a) := \hat{Q}^{(k)}(s,a) - \beta^{(k)}(s,a) \tag{4}$$

where $\beta^{(k)}(s,a)$ is a perturbed confidence interval. We add a random perturbation to the confidence bound defined as follows,

$$\beta^{(k)}(s,a) := \sqrt{\frac{2R_{\max}^2 \sqrt{T} g^{(k)}(s,a)}{n^{(k)}(s,a)}} + 2\epsilon^{(k+1)}, \tag{5}$$

where $g^{(k)}(s,a)$ is a positive random perturbation sampled from the sub-Gaussian distribution with mild conditions and $\epsilon^{(k+1)}$ is a tolerance of estimation error at the next depth. More detail conditions on $R_{\max}$, $g^{(k)}$ and $\epsilon^{(k+1)}$ can be found in Theorem 1 and 2. By using the perturbed confidence

Algorithm 2: PBAI Task and Motion Planning

Global variables: $\mathcal{T}, \mathcal{A}, \mathcal{P}, r, T, K, \gamma$
Input: s_0
Initialize a tree $\mathcal{T}(s_0) = \{U = \emptyset, \hat{Q}(s, \cdot) = -\infty, 0\}$
for $t = 1, \cdots, T$ **do**
  Level-One-Planning($s_0, 0$)
  if $\mathcal{T}(s_0)$.feasible_high_level_plan == True **then**
    Level-Two-Planning()
  end if
end for
return $\tau = \{joint(\arg\max_i \mathcal{T}(s_0).\hat{Q}(x_0, a(:)))\}$

Algorithm 3: Level-One-Planning

Global variables: $\mathcal{T}, \mathcal{A}, \mathcal{P}, r, T, K, \gamma$
Input: s, k
Initialize $r_{\text{goal}}, r_{\text{inf}}, \mathcal{T}(s_0)$.feasible_high_level_plan = False
if $s ==$ goal **then**
  $\mathcal{T}(s)$.feasible_high_level_plan = True
  return r_{goal}
end if
if $s ==$ infeasible or $k == K$ **then**
  return r_{inf}
end if
if Child of $\mathcal{T}(s)$ is empty **then**
  $\mathcal{T}(s).\hat{Q}(s, a) = -\infty$ and $\mathcal{T}(s).N(s, a) = 0$ for all $a \in \mathcal{A}$
  a = Sample $a \in \mathcal{A}$
else
  $\mathcal{T}(s).N(s) = \mathcal{T}(s).N(s) + 1$
  a = PBAI($\mathcal{T}(s).\hat{Q}, \mathcal{T}(s).N$)
end if
$\mathcal{T}(s).U = \mathcal{T}(s).U \cup \{a\}$
$s' = \mathcal{P}(s, a)$
$\hat{Q}_{new} = r(s, a, s') + \gamma$Level-One-Planning($s', k + 1$)
$\mathcal{T}(s).\hat{Q}(s, a) = \max(\hat{Q}_{new}, \mathcal{T}(s).\hat{Q}(s, a))$

Algorithm 4: Level-Two-Planning

Global variables: $\mathcal{T}, \mathcal{A}, \mathcal{P}, r, T, K, \gamma$
P = get_sub_optimal_plans($\mathcal{T}$)
for $p \in P$ **do**
  Get reward $r_{\text{p}} =$ RRT-STAR(p)
  $\hat{Q}_{new} = r_{\text{p}} + \mathcal{T}(s).\hat{Q}(s, a))$
  $\mathcal{T}(s).\hat{Q}(s, a) = \max(\hat{Q}_{new}, \mathcal{T}(s).\hat{Q}(s, a))$
end for

bound, we compute the length of the confidence interval for each action a as follows,

$$B^{(k)}(s, a) := \max_{a' \neq a} U^{(k)}(s, a') - L^{(k)}(s, a). \tag{6}$$

Then, we denote the possibly optimal action at s as $b^{(k)}(s) := \arg\min_a B^{(k)}(s, a)$. Note that $b^{(k)}(s)$ may be an optimal action with high probability since $B^{(k)}(s, a)$ indicates the maximum gap between the lower bound of a and the upper bounds of other actions not including a. Furthermore, we define $u^{(k)}(s) := \arg\max_{a \neq b^{(k)}(s)} U^{(k)}(s, a)$, which is the best possible action except for $b^{(k)}(s)$. Then, our method selects the action which has the maximum confidence bounds among $b^{(k)}(s)$ and $u^{(k)}(s)$.

$$I^{(k)}(s) = \arg\max_{a \in \{b^{(k)}(s), u^{(k)}(s)\}} \beta^{(k)}(s, a). \tag{7}$$

The entire algorithm is summarized in Algorithm 1. We would like to emphasize that the source of noise of estimated values consists of randomness in the sample-based kinematic planner and randomness of the estimated value of the subtree. However, general UCT cannot properly represent confidence. Hence, by using perturbation, the randomness in confidence bound finds a better solution than UCT.

C. Proposed Algorithm

The entire proposed method is summarized in Algorithm 2, which combines Level-One-Planning (Algorithm 3) and Level-Two-Planning (Algorithm 4). The main algorithm takes the initial state s_0, $\mathcal{T}$ (a tree), $\mathcal{A}$ (a set of actions), $\mathcal{P}$ (a transition function), r (a reward function), T (the maximum horizon), K (the maximum depth), and γ (the discount factor) as input variables. After initializing the tree in the initial state, Level-One-Planning is iteratively performed for T rounds. If a feasible high-level plan is obtained during Level-One-Planning, then, Level-Two-Planning is performed to optimize kinematic trajectories. The resulting output is a sequence of symbolic states, actions, and corresponding trajectories that maximizes the Q-values.

Level-One-Planning uses MCTS that recursively expands state nodes and action nodes based on action selection rules and the state transition model. Then, the recursive function will stop if a goal state is reached, or the current state is infeasible, or the maximum depth is reached. In the state node, if child action nodes are empty, then, actions are randomly sampled, otherwise, the PBAI method is used to obtain the next actions. All nodes maintain the best values found during iterations and the values are updated if better solutions with higher Q-values are found.

Level-Two-Planning obtains a set of trajectories if a feasible high-level plan is obtained through Level-One-Planning. For each high-level plan, RRT* motion planning computes r_p after finding trajectories. The final Q-value is updated by adding r_p to the previously updated $\hat{Q}(s, a)$ obtained from Level-One-Planning.

D. Theoretical Analysis

In this section, we analyze the convergence speed of the proposed planning method. We first provide convergence speed at the leaf node of the tree. At the leaf node, the planning problem is reduced to a simple MAB problem. Then, we extend the result at the leaf node to arbitrary depth k. We define the sum of expected rewards of MCTS at the state s after n visits as $\bar{V}_n^{(k)}(s) := \sum_{h=k}^{K} r(s_h, a_h)$, where $s_k := s$, a_h is the best action computed by MCTS, and $s_{h+1} := \mathcal{P}(s_h, a_h)$. Then, we define the simple regret of the proposed MCTS,

$$\text{regret}_n^{(k)}(s) := V_\star^{(k)}(s) - \bar{V}_n^{(k)}(s). \tag{8}$$

The problem of selecting optimal action from leaf nodes is equivalent to a MAB problem since the leaf node will

no longer be extended. Hence, the simple regret of MCTS becomes the simple regret of MAB as $\text{regret}_n^{(k)}(s) := \max_{a'} r(s,a') - r(s,b^{(k)}(s))$ where $b^{(k)}(s)$ is the best action computed by a PBAI at the leaf node. Before stating the main theorem, we first define a nice event as $\mathcal{E}(s) := \{\forall a \in \mathcal{A},\ |\hat{R}(s,a) - r(s,a)| < \beta^{(k)}(s,a),\ g^{(k)}(s,a) \leq (n - N_{\mathcal{A}})\epsilon^2/(8N_{\mathcal{A}}R_{\max}^2\sqrt{T})\}$ where $N_{\mathcal{A}}$ is the number of action and n is the number of visits at the node s. We first state the following useful lemmas on $\mathcal{E}$.

Lemma 1. *Let $B^{(k)}(s) = U^{(k)}(s,u^{(k)}(s)) - L^{(k)}(s,b^{(k)}(s))$. Then, $B^{(k)}(s,b^{(k)}(s)) = B^{(k)}(s)$ holds.*

Lemma 2. *At each visit $t \in [1,\cdots,N^{(k)}(s)]$, the following statements hold.*

- *If $u^{(k)}(s)$ is pulled, $L^{(k)}(s,u^{(k)}(s)) \leq L^{(k)}(s,b^{(k)}(s))$*
- *If $b^{(k)}(s)$ is pulled, $U^{(k)}(s,u^{(k)}(s)) \leq U^{(k)}(s,b^{(k)}(s))$*

Lemma 3. *If an action a is pulled at visit n, then, $B^{(k)}(s) < 2\beta^{(k)}(s,a)$.*

Lemma 4. *On event $\mathcal{E}$, for any action $a \neq a_\star$, we have $B^{(k)}(s,a) \geq \max_{a'} r(s,a') - r(s,a)$.*

Lemma 5. *On event $\mathcal{E}$, if $a \in \{b^{(k)}(s), u^{(k)}(s)\}$ is pulled, we have,*

$$B^{(k)}(s) \leq \min\left(0, -\Delta^{(k)}(s,a) + 2\beta^{(k)}(s,a)\right) + 2\beta^{(k)}(s,a)$$

The proofs of Lemma 1, 2, 3, 4, and 5 are found in the supplementary material [31], but, the Lemmas can be easily proved by applying the same techniques in [22]. However, we would like to emphasize that the results in [22] have been extended to the version of perturbed exploration in our work, which is the main difference from [22]. By using these lemmas, we prove the following theorem. The proofs of Theorem 1, 2 and Corollary 1 details can be found in the supplementary material [31].

Theorem 1. *Assume that s is a leaf node at depth k and it is visited n times after T iterations. If we sample the perturbation from sub-Gaussian with $\mu_n = (n - N_{\mathcal{A}})\epsilon^2/(8N_{\mathcal{A}}R_{\max}^2\sqrt{T}) + \sqrt{T}$ and $\sigma_n^{-2} = 1 + 2\ln(N_{\mathcal{A}}T)/T$, then, the simple regret at the leaf node s satisfies,*

$$\mathbb{P}\left(\text{regret}_n^{(k)}(s) > \epsilon\right) \leq O\left(N_{\mathcal{A}}T\exp\left(-\frac{T}{2} - \frac{\epsilon^2}{8N_{\mathcal{A}}R_{\max}^2}\right)\right),$$

where $N_{\mathcal{A}}$ is the number of discrete actions and $R_{\max}$ is an upper bound of reward.

Corollary 1. *Assume that s is a leaf node at depth k and it is visited n times after T iterations. An estimation error of the optimal value satisfies,*

$$\mathbb{P}\left(|\max_a r(s,a) - \hat{R}(s,b^{(k)}(s))| > 2\epsilon\right) \leq O\left(N_{\mathcal{A}}Te^{-\frac{T}{2} - \frac{\epsilon^2}{8N_{\mathcal{A}}R_{\max}^2}}\right). \quad (9)$$

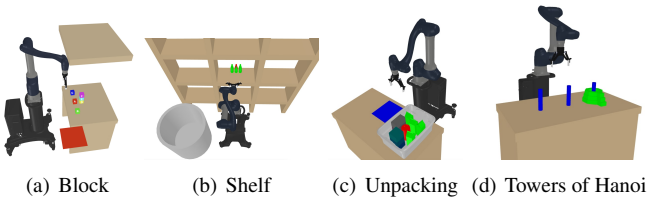
Now, we derive the convergence rate for the entire tree by using mathematical induction.

Theorem 2. *Let $\{\epsilon^{(k)}\}_{k=1}^H$ be the set of a threshold of tolerance for every depth such that $\epsilon^{(k)} - 4\epsilon^{(k+1)} > 0$ and $\epsilon^{(k)} - 4\epsilon^{(k+1)} > \epsilon^{(k+1)} - 4\epsilon^{(k+2)}$ hold for every depth k. For any state s at any depth k, assume that the state s is visited at n time after T iterations. If we sample the perturbation from sub-Gaussian with $\mu_n = (n - N_{\mathcal{A}})(\epsilon^{(k)} - 4\epsilon^{(k+1)})^2/(8N_{\mathcal{A}}R_{\max}^2\sqrt{T}) + \sqrt{T}$ and $\sigma_n^{-2} = 1 + 2\ln(N_{\mathcal{A}}T)/T$, then, the simple regret at s satisfies,*

$$\mathbb{P}\left(\text{regret}_n^{(k)}(s) > \epsilon^{(k)}\right) \leq O\left((N_{\mathcal{A}}T)^{H+1-k}e^{-\frac{T}{2} - \frac{\left(\epsilon^{(k)} - 4\epsilon^{(k+1)}\right)^2}{8N_{\mathcal{A}}R_{\max}^2}}\right)$$

and the estimation error satisfies,

$$\mathbb{P}\left(\hat{V}^{(k)}(s) - V_\star^{(k)}(s) > 2\epsilon^{(k)}\right) \leq O\left((N_{\mathcal{A}}T)^{H+1-k}e^{-\frac{T}{2} - \frac{\left(\epsilon^{(k)} - 4\epsilon^{(k+1)}\right)^2}{8N_{\mathcal{A}}R_{\max}^2}}\right).$$



(a) Block (b) Shelf (c) Unpacking (d) Towers of Hanoi

Fig. 3: Illustration of benchmark environments.

V. Experimental Results

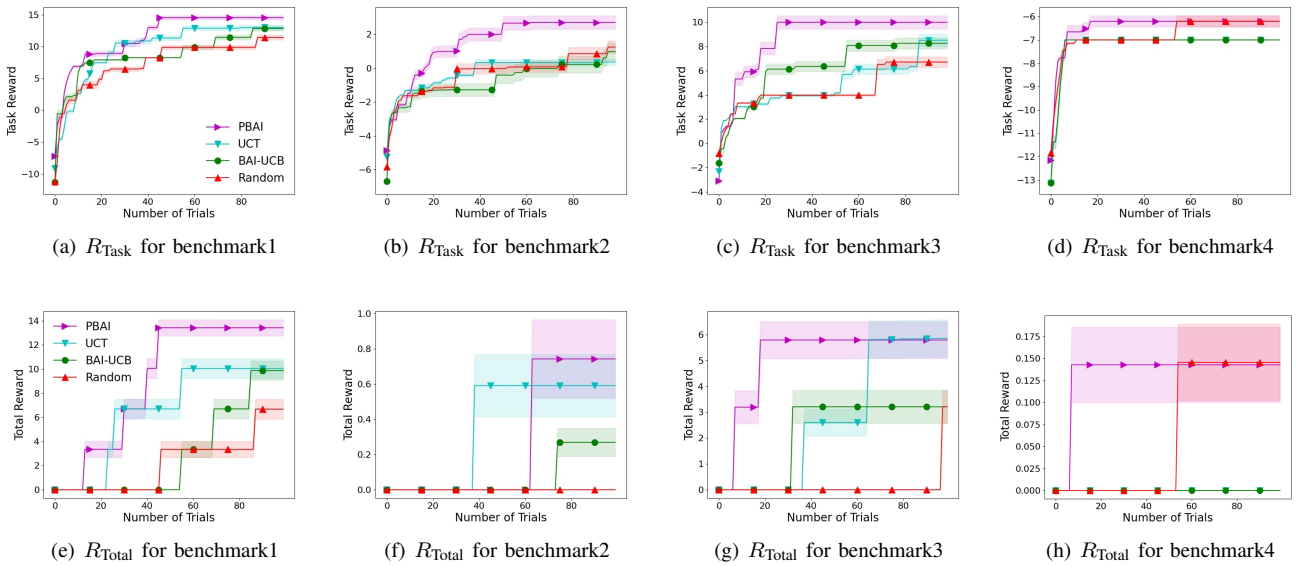
A. Benchmark Environments and Evaluation Metrics

We validate our proposed method in four benchmarks: Blocks, Shelf, Unpacking, and Towers of Hanoi. *Blocks* and *Towers of Hanoi* were created by referring to [32]. All benchmarks have the following reward function: 0 for feasible pick, r_{goal} for reaching a goal state, and r_{inf} for infeasible actions.

1) Blocks World (Fig. 3(a)): There are six boxes in alphabetical order. The goal is to stack boxes on the table in alphabetical order on the tray. The additional reward is as follows: $r_a(1/k)$ for stacking in alphabetical order, $-r_a(1/k)$ for stacking in non alphabetical order, where $r_{\text{goal}} = 5$, $r_{\text{inf}} = -10$, $r_a = 20$, and k is current tree depth.

2) Shelf World (Fig. 3(b)): Six bottles are placed on a shelf, one of which is a red bottle called a goal bottle. A robot aims to grab the red bottle with the gripper and place it into a round basket. The additional reward is $-1 \cdot \mathbb{I}[d \leq 0.2] + 1 \cdot \mathbb{I}[d > 0.2]$ where $\mathbb{I}$ is an indicator, d is the y-axis distance between the center of target bottle and currently replaced bottle, $r_{\text{goal}} = 5$, and $r_{\text{inf}} = -5$.

3) Unpacking World (Fig. 3(c)): Boxes of different sizes are stacked inside a plastic storage box containing green milk cartons and a red can, and a blue tray is floating next to the storage box. This benchmark aims to grab the can and put it on the blue tray. The additional reward is -1 for re-grasping an object already switched, 2 otherwise, where $r_{\text{goal}} = 10$ is set to, and $r_{\text{inf}} = -5$.



(a) R_{Task} for benchmark1 (b) R_{Task} for benchmark2 (c) R_{Task} for benchmark3 (d) R_{Task} for benchmark4

(e) R_{Total} for benchmark1 (f) R_{Total} for benchmark2 (g) R_{Total} for benchmark3 (h) R_{Total} for benchmark4

Fig. 4: Comparison of R_{Task} (Top) and R_{Total} (Bottom) for each benchmark.

4) Towers of Hanoi world (Fig. 3(d)): According to the Towers of Hanoi rules, a larger disk cannot be placed on top of a smaller disk. The game aims to build stacks in order of size. we set to $r_{\text{goal}} = 15$, and $r_{\text{inf}} = -5$.

We evaluated four exploration methods per benchmark for the metric: PBAI (Ours), UCT, BAI-UCB, and Random. Each algorithm was evaluated on the benchmark tasks for ten random seeds. The hyperparameter is optimized by using brute force search. We used two metrics to evaluate our method's performance: reward and success rate. For reward metric, we compared two rewards: $R_{\text{Task}} := \sum_{k=1}^{H} r(s_k, a_k, s_{k+1})$ and $R_{\text{Total}} := \sum_{k=1}^{H} \bar{R}(s_k, a_k, s_{k+1})$, where R_{Task} is a sum of task planning rewards. R_{total} is a total reward considering the robot trajectory. The success rate is the percentage of successful planning attempts for benchmark 1.

B. Results and Analysis

All results are shown in Fig. 4. First, we compare the task rewards of each algorithm. For benchmark 1, Fig. 4(a) shows that PBAI finds the optimal solution much faster than other methods. Especially, R_{Task} of PBAI is 10.59% higher than the baseline UCT, which is the second-best algorithm. Similarly, Fig. 4(b) and 4(c) also shows that R_{Total} of PBAI increases the fastest and reaches the best task reward compared to other algorithms. Unlike other benchmarks, as shown in Fig. 4(d), all algorithms show similar performances for benchmark 4. In this regard, the proposed PBAI nearly outperforms other task planning methods in terms of both maximum task rewards and the fastest convergence speed.

For total rewards, a similar tendency to task rewards can be observed. As shown at the bottom of Fig. 4, we would like to emphasize that the PBAI method is the most efficient in finding the optimal trajectory. Especially, Fig. 4(h) shows that UCB and BAI-UCB methods sometimes show worse results than random methods. We believe that this result supports that the general confidence bound cannot help the exploration for the tree search since it cannot represent the optimization errors of sub-trees.

TABLE I: Success Rates (Benchmark 1)

Algorithm	Simulation		Real World	
	Trials	Rates (%)	Trials	Rates (%)
PBAI (Ours)	68/83	81.9	16/20	80
UCT	47/83	56.6	11/20	55
BAI-UCB	39/83	47.0	9/20	45
Random	11/83	13.3	3/20	15

The results of the success rate in simulation and real robot are shown in Table I. In simulations, the random algorithm has the worst performance 13.3%, and UCT has a better performance than BAI-UCB by 9.6%. Our algorithm achieves a success rate of 81.9% and outperforms the baseline UCT by 25.3% for benchmark 1. Similarly, in real experiments, we observe that the significantly higher success rate of our method compared to other methods outperforms. The proposed method achieves the highest success rate in real-world experiments.

VI. Conclusions

We proposed an efficient and effective task planning framework for Task and Motion Planning (TAMP) that incorporates a random perturbation to the confidence bounds. The Perturbation-based Best Arm Identification (PBAI) algorithm was shown to be optimal through theoretical analysis, and its superior performance over other search-based approaches was demonstrated in empirical evaluations on four benchmarks. We believe that our algorithm can be applied to real-world scenarios, such as cooking and furniture assembly, to help a robot find efficient task and motion plans.

References

[1] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Integrated task and motion planning in belief space. *International Journal of Robotics Research*, 32(9-10):1194–1227, 2013.

[2] Siddharth Srivastava, Eugene Fang, Lorenzo Riano, Rohan Chitnis, Stuart Russell, and Pieter Abbeel. Combined task and motion planning through an extensible planner-independent interface layer. *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014*, pages 639–646, 2014.

[3] George Konidaris, Leslie Pack Kaelbling, and Tomas Lozano-Perez. From skills to symbols: Learning symbolic representations for abstract high-level planning. *Journal of Artificial Intelligence Research*, 61:215–289, 2018.

[4] Daniel E Koditschek. Robot planning and control via potential functions. *The robotics review*, page 349, 1989.

[5] Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning. *Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling, Nancy, France, October 26-30, 2020*, pages 440–448, 2020.

[6] Mohamed Elbanhawi and Milan Simic. Sampling-based robot motion planning: A review. *Ieee access*, 2:56–77, 2014.

[7] Jay Kamat, Joaquim Ortiz de Haro, Marc Toussaint, Florian T. Pokorny, and Andreas Orthey. BITKOMO: combining sampling and optimization for fast convergence in optimal motion planning. *CoRR*, abs/2203.01751, 2022.

[8] Chaitanya Mitash, Abdeslam Boularias, and Kostas E Bekris. Improving 6d pose estimation of objects in clutter via physics-aware monte carlo tree search. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3331–3338. IEEE, 2018.

[9] Haoran Song, Joshua A Haustein, Weihao Yuan, Kaiyu Hang, Michael Yu Wang, Danica Kragic, and Johannes A Stork. Multi-object rearrangement with monte carlo tree search: A case study on planar nonprehensile sorting. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9433–9440. IEEE, 2020.

[10] Chanyeol Yoo, Samuel Lensgraf, Robert Fitch, Lee M Clemon, and Ramgopal Mettu. Toward optimal fdm toolpath planning with monte carlo tree search. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4037–4043. IEEE, 2020.

[11] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

[12] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. *In Proc. of the 17th European Conference on Machine Learning*, 4212:282–293, 2006.

[13] Marc A Toussaint, Kelsey Rebecca Allen, Kevin A Smith, and Joshua B Tenenbaum. Differentiable physics and stable modes for tool-use and manipulation planning. 2018.

[14] Richard E. Korf. Depth-first iterative-deepening: An optimal admissible tree search. *Artif. Intell.*, 27(1):97–109, 1985.

[15] Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Ffrob: An efficient heuristic for task and motion planning. In H. Levent Akin, Nancy M. Amato, Volkan Isler, and A. Frank van der Stappen, editors, *the Eleventh International Workshop on the Algorithmic Foundations of Robotics, WAFR*, volume 107 of *Springer Tracts in Advanced Robotics*, pages 179–195. Springer, 2014.

[16] Javier G. Martin, José Ramón Domínguez Frejo, Ramón A. García, and Eduardo F. Camacho. Multi-robot task allocation problem with multiple nonlinear criteria using branch and bound and genetic algorithms. *Intell. Serv. Robotics*, 14(5):707–727, 2021.

[17] Jonathan Binney and Gaurav S Sukhatme. Branch and bound for informative path planning. In *2012 IEEE international conference on robotics and automation*, pages 2147–2154. IEEE, 2012.

[18] Ailsa H Land and Alison G Doig. An automatic method for solving discrete programming problems. In *50 Years of Integer Programming 1958-2008*, pages 105–132. Springer, 2010.

[19] Bing Fu, Lin Chen, Yuntao Zhou, Dong Zheng, Zhiqi Wei, Jun Dai, and Haihong Pan. An improved a* algorithm for the industrial robot path planning with high success rate and short length. *Robotics and Autonomous Systems*, 106:26–37, 2018.

[20] Jennifer E. King, Vinitha Ranganeni, and Siddhartha S. Srinivasa. Unobservable monte carlo planning for nonprehensile rearrangement tasks. In *IEEE International Conference on Robotics and Automation, ICRA*, pages 4681–4688. IEEE, 2017.

[21] Chris Paxton, Yotam Barnoy, Kapil D. Katyal, Raman Arora, and Gregory D. Hager. Visual robot task planning. In *International Conference on Robotics and Automation, ICRA 2019, Montreal, QC, Canada, May 20-24, 2019*, pages 8832–8838. IEEE, 2019.

[22] Victor Gabillon, Mohammad Ghavamzadeh, and Alessandro Lazaric. Best arm identification: A unified approach to fixed budget and fixed confidence. In Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 3221–3229, 2012.

[23] Emilie Kaufmann and Wouter M. Koolen. Monte-carlo tree search by best arm identification. *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4897–4906, 2017.

[24] Marc Toussaint and Manuel Lopes. Multi-bound tree search for logic-geometric programming in cooperative manipulation domains. In *2017 IEEE International Conference on Robotics and Automation, ICRA 2017, Singapore, Singapore, May 29 - June 3, 2017*, pages 4044–4051. IEEE.

[25] James J. Kuffner Jr. and Steven M. LaValle. Rrt-connect: An efficient approach to single-query path planning. *Proceedings of the 2000 IEEE International Conference on Robotics and Automation, ICRA 2000, April 24-28, 2000, San Francisco, CA, USA*, pages 995–1001, 2000.

[26] Michael S Branicky, Michael M Curtiss, Joshua Levine, and Stuart Morgan. Sampling-based planning, control and verification of hybrid systems. *IEE Proceedings-Control Theory and Applications*, 153(5):575–590, 2006.

[27] Steven M LaValle et al. Rapidly-exploring random trees: A new tool for path planning. 1998.

[28] Marc Toussaint. Robot trajectory optimization using approximate inference. pages 1049–1056, 2009.

[29] Michael Posa, Cecilia Cantu, and Russ Tedrake. A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research*, 33(1):69–81, 2014.

[30] Jean-Yves Audibert, Sébastien Bubeck, and Rémi Munos. Best arm identification in multi-armed bandits. In Adam Tauman Kalai and Mehryar Mohri, editors, *COLT 2010 - The 23rd Conference on Learning Theory, Haifa, Israel, June 27-29, 2010*, pages 41–53. Omnipress, 2010.

[31] Daejong Jin, Juhan Park, and Kyungjae Lee. Perturbation-based best arm identification for efficient task planning with monte-carlo tree search (longer version). `https://drive.google.com/file/d/1LlCZQUdJsOG-a94Ug6J3VE-DVpik1fWc/view`, 2023. [Online; accessed 03-March-2023].

[32] Fabien Lagriffoul, Neil T Dantam, Caelan Garrett, Aliakbar Akbari, Siddharth Srivastava, and Lydia E Kavraki. Platform-independent benchmarks for task and motion planning. *IEEE Robotics and Automation Letters*, 3(4):3765–3772, 2018.