Static and Dynamic Values of Computation in MCTS

Eren Sezener
DeepMind

Peter Dayan
Max Planck Institute for Biological Cybernetics Tübingen
University of Tübingen

Abstract

Monte-Carlo Tree Search (MCTS) is one of the most-widely used methods for planning, and has powered many recent advances in artificial intelligence. In MCTS, one typically performs computations (i.e., simulations) to collect statistics about the possible future consequences of actions, and then chooses accordingly. Many popular MCTS methods such as UCT and its variants decide which computations to perform by trading-off exploration and exploitation. In this work, we take a more direct approach, and explicitly quantify the value of a computation based on its expected impact on the quality of the action eventually chosen. Our approach goes beyond the *myopic* limitations of existing computation-value-based methods in two senses: (I) we are able to account for the impact of non-immediate (ie, future) computations (II) on non-immediate actions. We show that policies that greedily optimize computation values are optimal under certain assumptions and obtain results that are competitive with the state-of-the-art.

1 INTRODUCTION

Monte Carlo tree search (MCTS) is a widely used approximate planning method that has been successfully applied to many challenging domains such as computer Go [4, 21]. In MCTS, one estimates values of actions by stochastically expanding a search tree—capturing potential future states and actions with their respective values. Most MCTS methods rely on rules concerning how to expand the search tree, typically trading-off exploration and exploitation such as in UCT [11]. However, since no "real" reward accrues during internal search, UCT can be viewed as a heuristic [7, 8, 23]. In this paper, we propose a more direct approach by calculating values of MCTS computations (i.e., tree expansions/simulations).

In the same way that the value of an action in a Markov decision process (MDP) depends on subsequent actions, the value of a computation in MCTS should reflect subsequent computations. However, computing the optimal computation values—the value of a computation under an optimal computation policy—is known to be generally intractable [14, 18]. Therefore, one often resorts to "myopic" approximations of computation values, such as considering the impact of only the immediate computation in isolation from the subsequent computations. For instance, it has been shown that simple modifications to UCT, where myopic computation values inform the tree policy at the root node, can yield significant improvements [8, 23].

In this work, we propose tractable yet non-myopic methods for calculating computation values, going beyond the limitations of existing methods. To this end, we introduce static and dynamic value functions that form lower and upper bounds for state-action values in MCTS, independent of any future computations. We then utilize these functions to define static and dynamic values of computation, capturing the expected change in state values resulting from a computation. We show that the existing myopic computation value definitions in MCTS can be seen as specific instances of static computation values. The dynamic value function, on the other hand, is novel measure, and it enables non-myopic ways of selecting MCTS computations. We prove that policies that greedily maximize static/dynamic computation values are asymptotically optimal under certain assumptions. Furthermore, we also show that they outperform various MCTS baselines empirically.

2 BACKGROUND

In this section we cover some of relevant literature and introduce the notation.

2.1 MONTE CARLO TREE SEARCH

MCTS algorithms function by incrementally and stochastically building a search tree (given an environment model) to approximate state-action values. This incremental growth prioritizes the promising regions of the search space by directing the growth of the tree towards high value states. To elaborate, a *tree policy* is used to traverse the search tree and select a node which is not fully expanded—meaning, it has immediate successors that aren't included in the tree. Then, the node is expanded once by adding one of its unexplored children to the tree, from which a trajectory simulated for a fixed number of steps or until a terminal state is reached. Such trajectories are generated using a *rollout policy*; which is typically fast to compute—for instance random and uniform. The outcome of this trajectory—i.e., cumulative discounted rewards along the trajectory—is used to update the value estimates of the nodes in the tree that lie along the path from the root to the expanded node.

Upper Confidence Bounds applied to trees (UCT) [11] adapts a multi-armed bandit algorithm called UCB1 [1] to MCTS. More specifically, UCT's tree policy applies the UCB1 algorithm recursively down the tree starting from the root node. At each level, UCT selects the most promising action at state s via $\arg\max_{a\in\mathcal{A}_s} \hat{Q}(s,a) + c\sqrt{\frac{2\log N(s)}{N(s,a)}}$ where $\mathcal{A}_s$ is the set of available actions at s, $N(s,a)$ is the number of times the (s,a) is visited, $N(s) := \sum_{a\in\mathcal{A}_s} N(s,a)$, $\hat{Q}(s,a)$ is the average reward obtained by performing rollouts from (s,a) or one of its descendants, and c is a positive constant, which is typically selected empirically. The second term of the UCT-rule assigns higher scores to nodes that are visited less frequently. As such, it can be thought of as an exploration bonus.

UCT is simple and has successfully been utilized for many applications. However, it has also been noted [8, 23] that UCT's goal is different from that of approximate planning. UCT attempts to ensure that the agent experiences little *regret* associated with the actions that are taken during the Monte Carlo simulations that comprise planning. However, since these simulations do not involve taking actions in the environment, the agent actually experience no true regret at all. Thus failing to explore actions based on this consideration could slow down discovery of their superior or inferior quality.

2.2 METAREASONING & VALUE OF INFORMATION

Howard [9] was the first to quantify mathematically the economic gain from obtaining a piece of information. Russell and Wefald [18] formulated the *rational metareasoning* framework, which is concerned with how one should assign values to meta-level actions (i.e., computations). Hay et al. [8], Tolpin and Shimony [23] applied the principles of this framework to MCTS by modifying the tree-policy of UCT at the root node such that the selected child node maximizes the value of information. They showed empirically that such a simple modification can yield significant improvements.

The field of Bayesian optimization has evolved in parallel. For instance, what are known as *knowledge gradients* [19, 24] are equivalent to information/computation value formulations for flat/stateless problems such as multi-armed bandits.

Computation values have also been used to explain human and animal behavior. For example, it has been suggested that humans might leverage computation values to solve planning tasks in a resource efficient manner [13, 20], and animals might improve their policies by "replaying" memories with large computation values [17].

2.3 NOTATION

A finite Markov decision process (MDP) is a 5-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is a finite set of states $\mathcal{A}$ is a finite set of actions, $\mathcal{P}$ is the transition function such that $\mathcal{P}^a_{ss'} = P(s_t = s'|s_t = s, a_t = a)$, where $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$, $\mathcal{R}$ is the expected immediate reward function such that $\mathcal{R}^a_{ss'} = \mathbb{E}[r_{t+1}|s_t = s, a_t = a, s_{t+1} = s']$, where again $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$, γ is the discount factor such that $\gamma \in [0, 1)$.

We assume an agent interacts with the environment via a (potentially stochastic) policy π, such that $\pi(s,a) = P(a_t = a|s_t = s)$. These probabilities typically depend on parameters; these are omitted from the notation. The value of an action a at state s is defined as the expected cumulative discounted rewards following policy π, that is $Q^\pi(s,a) = \mathbb{E}_\pi\left[\sum_{i=0}^\infty \gamma^i r_{t+i}\middle| s_t = s, a_t = a\right]$.

The optimal action value function is defined as $Q^*(s,a) = \max_\pi Q^\pi(s,a)$ for all state-action pairs, and satisfies the *Bellman optimality recursion*:

$$Q^*(s,a) = \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma \max_{a'} Q^*(s',a')\right] .$$

We use $\mathcal{N}(\mu, \Sigma)$ and $\mathcal{N}(\mu, \sigma^2)$ to denote a multivariate and univariate Normal distribution respectively with mean vector/value μ and covariance matrix Σ or scale σ.

3 STATE-ACTION VALUES IN MCTS

To motivate the issues underlying this paper, consider the following example (Figure 1). Here, there are two rooms: one containing two boxes and the other containing five boxes. Each box contains an unknown but i.i.d. amount of money; and you are ultimately allowed to open only one box. However, you do so in stages. First you must choose a room, then you can open one of the boxes and collect the money. Which room should you choose? What if you know ahead of time that you could peek inside the boxes *after* choosing the room?

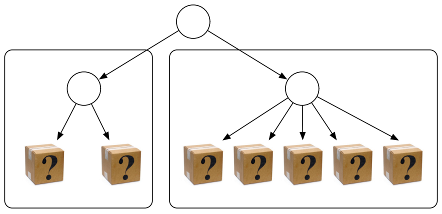


Figure 1: Illustration of the example. There are two rooms, one containing two boxes, another one containing five boxes. There is an unknown amount of money in each box.

In the first case, it doesn't matter which room one chooses, as all the boxes are equally valuable in expectation in absence of any further information. By contrast, in the second case, choosing the room with five boxes is the better option. This is because one can obtain further information by peeking inside the boxes—and more boxes mean more money in expectation, as one has the option to choose the best one.

Formally, let $X = \{x_i\}_{i=1}^{n_x}$ and $Y = \{y_i\}_{i=1}^{n_y}$ be sets of random variables denoting rewards in the boxes of the first and the second room respectively. Assume all the rewards are sampled i.i.d. and $n_x < n_y$. Then we have $\max_{x \in X} \mathbb{E}[x] = \max_{y \in Y} \mathbb{E}[y]$, which is why the two rooms are equally valuable if one has to choose a box blindly. On the other hand, $\mathbb{E}[\max_{x \in X} x] < \mathbb{E}[\max_{y \in Y} y]$, which is analogous to the case where boxes can be peeked in first.

If we consider MCTS with this example in mind, when we want to value an action at the root of the tree, backing up the estimated mean values of the actions lower in the tree may be insufficient. This is because the value of a root action is a convex combination of the "downstream" (e.g., leaf) actions; and, as such, uncertainty in the values of the leaves contributes to the expected value at the root due to Jensen's inequality. We formalize this notion of value as *dynamic value* in the following section and utilize it to define computation values later on.

3.1 STATIC AND DYNAMIC VALUES

We assume a planning setting where the environment dynamic (i.e., $\mathcal{P}$ and $\mathcal{R}$) is known. We could then compute Q^* in principle; however, this is typically computationally intractable. Therefore, we estimate Q^* by performing computations such as random environment simulations (e.g., MCTS rollouts). Note that, our uncertainty about Q^* is not epistemic—environment dynamic is known—but it is computational. In other words, if we do not know Q^*, it is because we haven't performed the necessary computations. In this subsection, we introduce static and dynamic value functions, which are "posterior" estimates of Q^* conditioned on computations.

Let us unroll the Bellman optimality equation for n-steps[1]. For a given "root" state, s_ρ, let $\Gamma_n(s_\rho)$ be the set of leaf state-actions—that is, state-actions can be transitioned to from s_ρ in exactly n-steps. Let Q_0^* be a random function denoting our prior beliefs about the optimal value function Q^* over $\Gamma_n(s_\rho)$. We then use $Q_n^*(s,a)$ to denote the (Bayes-)optimal state-action value, which we define as a function of Q_0^*:

$$Q_n^*(s,a) = \begin{cases} Q_0^*(s,a) & \text{if } n = 0 \\ \sum_{s'} \mathcal{P}_{ss'}^a [\mathcal{R}_{ss'}^a + & \text{else} \\ \quad \gamma \max_{a' \in \mathcal{A}_{s'}} Q_{n-1}^*(s',a')] & \end{cases},$$

where $\mathcal{A}_{s'}$ is the set of actions available at s'.

We assume it is possible to obtain noisy evaluations of Q^* for leaf state-actions by performing computations such as trajectory simulations. We further assume that the process by which a state-action value is sampled is a given, and we are interested in determining which state-action to sample from. Therefore, we associate each computation with a single state-action in $\Gamma_n(s_\rho)$; but, the outcome of a computation might be informative for multiple leaf values if they are dependent. Let $\bar{\omega} := (s,a) \in \Gamma_n(s_\rho)$ be a *candidate computation*. We denote the unknown outcome of this computation at time t with random variable $O_{\bar{\omega}t}$ (or equivalently, O_{sat}), which we assume to be $O_{\bar{\omega}t} = Q_0^*(\bar{\omega}) + \epsilon_t$ where ϵ_t is an unknown noise term with a known distribution and is i.i.d. sampled for each t. If we associate a candidate computation $\bar{\omega}$ with its unknown outcome $O_{\bar{\omega}t}$ at time t, we refer to the resulting tuple as a *closure* and denote it as $\Omega_t := (\bar{\omega}, O_{\bar{\omega}t})$. Finally, we denote a *performed computation* at time t, by dropping the bar, as $\omega_t := (\bar{\omega}, o_{\bar{\omega}t})$ where $o_{\bar{\omega}t}$ (or equivalently, o_{sat}) is the observed outcome of the computation that we assume to be $o_{\bar{\omega}t} \sim O_{\bar{\omega}t}$, and thus $\omega_t \sim \Omega_t$. In the context of MCTS, o_{sat} will be the cumulative discounted reward

[1]Obtaining, what is sometimes referred to as the n-step Bellman equation.

of a simulated trajectory from (s,a) at time t. We will obtain these trajectories using an adaptive, asymptotically optimal sampler/simulator (e.g., UCT), such that $\lim_{t\to\infty}\mathbb{E}[O_{sat}] = Q^*(s,a)$. This means $\{O_{sat}\}_t$ is a non-stationary stochastic process in practice; yet, we will treat it as a stationary process, as reflected in our i.i.d. assumption.

Let $\omega_{1:t}$ be a sequence of t performed computations concerning arbitrary state-actions in $\Gamma_n(s_\rho)$ and s_ρ be the current state of the agent on which we can condition Q^*_0. Because $\omega_{1:t}$ contains the necessary statistics to compute the posterior leaf values, we will sometimes refer to it as the *knowledge state*. We denote the resulting posterior values for a $(s,a)\in\Gamma_n(s_\rho)$ as $Q^*_0(s,a)|\omega_{1:t}$ and the joint values of leaves as $Q^*_0|\omega_{1:t} = (Q^*_0(s,a)|\omega_{1:t} : (s,a)\in\Gamma_n(s_\rho))$.

We define the *dynamic value function* as the expected value the agent should assign to an action at s_ρ given $\omega_{1:t}$, assuming it could resolve all of the remaining uncertainty about posterior leaf state-action values $Q^*_0|\omega_{1:t}$.

Definition 1. The **dynamic value function** is defined as $\psi_n(s,a|\omega_{1:t}) := \mathbb{E}_{Q^*_0|\omega_{1:t}}[\Upsilon_n(s,a|\omega_{1:t})]$ where,

$$\Upsilon_n(s,a|\omega_{1:t}) := \begin{cases} Q^*_0(s,a)|\omega_{1:t} & \text{if } n=0 \\ \sum_{s'}\mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'}+ & \text{else} \\ \gamma\max_{a'}\Upsilon_{n-1}(s',a'|\omega_{1:t})] & \end{cases}$$

where $\mathcal{A}_{s'}$ is the set of actions available at s'.

The 'dynamic' in the term reflects the fact that the agent may change its mind about the best actions available at each state within n-steps; yet, this is reflected and accounted for in ψ_n. A useful property of ψ_n is that is time-consistent in the sense that it does not change with further computations in expectation. Let $\Omega_{1:k} := \{(\bar{\omega}_i, O_{\bar{\omega}_i i})\}_{i=1}^k$ be a sequence of k closures. Then the following equality holds for any $\Omega_{1:k}$:

$$\psi_n(s_\rho,a|\omega_{1:t}) = \mathbb{E}_{\Omega_{1:k}}[\psi_n(s_\rho,a|\omega_{1:t}\Omega_{1:k})], \qquad (1)$$

due to the law of total expectation, where $\omega_{1:t}\Omega_{1:k}$ is a concatenation. This might seem paradoxical: why perform computations if action values do not change in expectation? The reason is that we care about the maximum of dynamic values over actions at s_ρ, which increases in expectation as long as computations resolve some uncertainty. Formally, $\max_{a\in\mathcal{A}_{s_\rho}}\psi_n(s_\rho,a|\omega_{1:t}) \leq \mathbb{E}_{\Omega_{1:k}}[\max_{a\in\mathcal{A}_{s_\rho}}\psi_n(s_\rho,a|\omega_{1:t}\Omega_{1:k})]$, due to Jensen's inequality, just as in the example of the boxes.

Dynamic values capture one extreme: valuation of actions assuming perfect information in the future. Next, we consider the other extreme, valuation under zero information in the future, which is given by the *static value function*.

Definition 2. We define the **static value function** as

$$\phi_n(s,a|\omega_{1:t}) := \begin{cases} \mathbb{E}[Q^*_0(s,a)|\omega_{1:t}] & \text{if } n=0 \\ \sum_{s'}\mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'}+ & \text{else} \\ \gamma\max_{a'}\phi_{n-1}(s',a'|\omega_{1:t})] & \end{cases}$$

where $\mathcal{A}_{s'}$ is the set of actions available at s'.

In other words, $\phi_n(s_\rho,a)$ captures how valuable (s_ρ,a) would be if the agent were to take n actions before running any new computations. In Figure 2, we graphically contrast dynamic and static values, where the difference is the stage at which the expectation is taken. For the former, it is done at the level of the root actions; for the latter, at the level of the leaves.

Going back to our example with the boxes, dynamic value of a room assumes that you open all the boxes after entering the room, whereas the static value assumes you do not open any boxes. What can we say about the in-between cases: action values under a finite number of future computations? Assume we know that the agent will perform k computations before taking an action at s_ρ. The optimal allocation of these k computations to leaf nodes is known to be intractable even in a simpler bandit setting [16]. That said, for any allocation (and for any finite k), static and dynamic values will form lower and upper bounds on expected action values nevertheless. We formalize this for a special case below.

Proposition 1. *Assume an agent at state s_ρ and knowledge state $\omega_{1:t}$ decides to perform $\bar{\omega}_{1:k}$, a sequence of k candidate computations, before taking n actions. Then the expected future value of $a\in\mathcal{A}_{s_\rho}$ prior to observing any of the k-computation outcomes is equal to $\mathbb{E}_{\Omega_{1:k}}[\phi_n(s_\rho,a|\omega_{1:t}\Omega_{1:k})]$, where $\Omega_{1:k} = \{(\bar{\omega}_i, O_{\bar{\omega}_i i})\}_{i=1}^k$. Then,*

$$\psi_n(s_\rho,a|\omega_{1:t}) \geq \mathbb{E}_{\Omega_{1:k}}[\phi_n(s_\rho,a|\omega_{1:t}\Omega_{1:k})] \geq \phi_n(s_\rho,a|\omega_{1:t}),$$

where both bounds are tight.

The proof is provided in the Appendix.

4 VALUE OF COMPUTATION

In this section, we use the static and dynamic value functions to define computation values. We show that these computation values have desirable properties and that policies greedily-maximizing these values are optimal in certain senses. Lastly, we compare our definitions to the existing computation value definitions.

Definition 3. We define the **value of computation** at state s_ρ for a sequence of candidate computations $\bar{\omega}_{1:k}$ given a static or dynamic value function $f\in\{\phi_n,\psi_n\}$

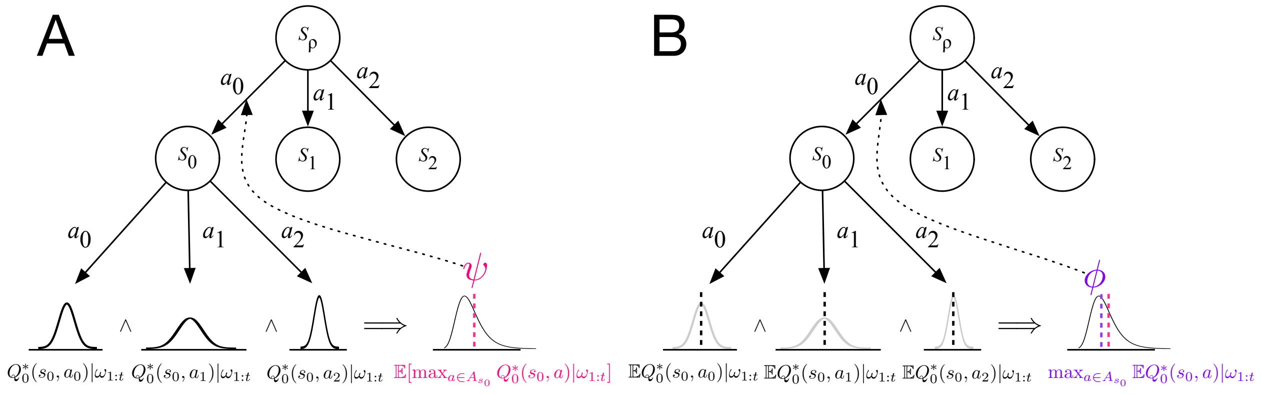

A
s_ρ
a_0
a_1
a_2
s_0
s_1
s_2
a_0
a_1
a_2
ψ
$\wedge$
$\wedge$
$\Longrightarrow$
$Q_0^*(s_0,a_0)|\omega_{1:t}$
$Q_0^*(s_0,a_1)|\omega_{1:t}$
$Q_0^*(s_0,a_2)|\omega_{1:t}$
$\mathbb{E}[\max_{a\in A_{s_0}} Q_0^*(s_0,a)|\omega_{1:t}]$
B
s_ρ
a_0
a_1
a_2
s_0
s_1
s_2
a_0
a_1
a_2
ϕ
$\wedge$
$\wedge$
$\Longrightarrow$
$\mathbb{E}Q_0^*(s_0,a_0)|\omega_{1:t}$
$\mathbb{E}Q_0^*(s_0,a_1)|\omega_{1:t}$
$\mathbb{E}Q_0^*(s_0,a_2)|\omega_{1:t}$
$\max_{a\in A_{s_0}} \mathbb{E}Q_0^*(s_0,a)|\omega_{1:t}$

Figure 2: Graphical illustration dynamic (ψ_n) and static (ϕ_n) value functions for $n = 2$. We ignore immediate rewards and the discounting for simplicity. In Panel A, dynamic values (given by ψ_2) are obtained by calculating the *expected maximum* of all state-action values (given by Q_0^*) lying 2-steps away. Whereas, the static values (given by ϕ_2) are obtained by calculating the *maximum of expectations* of state-action values, as shown in Panel B.

and a knowledge state $\omega_{1:t}$ as

$$\mathrm{VOC}_f(s_\rho, \bar{\omega}_{1:k}|\omega_{1:t}) = \mathbb{E}_{\Omega_{1:k}}\left[\max_{a\in\mathcal{A}_{s_\rho}} f(s_\rho, a|\omega_{1:t}\Omega_{1:k})\right] - \max_{a\in\mathcal{A}_{s_\rho}} f(s_\rho, a|\omega_{1:t}) \,,$$

where $\bar{\omega}_{1:k}$ specifies the state-actions in $\Omega_{1:k}$, that is, $\Omega_{1:k} = \{(\bar{\omega}_i, O_{\bar{\omega}_i i})\}_{i=1}^k$ where $\bar{\omega}_i$ is the ith element of $\bar{\omega}_{1:k}$.

We refer to computation-policies that choose computations based on greedy maximization one by one (i.e., $k = 1$) of VOC as VOC(ϕ_n)-greedy and VOC(ψ_n)-greedy depending on which value function is utilized. We assume these policies stop if and only if $\forall\bar{\omega}$: $\mathrm{VOC}_f(s_\rho, \bar{\omega}|\omega_{1:t}) = 0$. Our greedy policies consider and select computations one-by-one. Alternatively, one can perform a forward search over future computation sequences, similar to the search over future actions sequences in Guez et al. [6]. However, this adds another meta-level to our metareasoning problem; thus, further increasing the computational burden.

We analyze these greedy policies in terms of the *Bayesian simple regret*, which we define as the difference between two values. The first is the maximum value the agent could expect to reap assuming it can perform infinitely many computations, thus resolving all the uncertainty, before committing to an immediate action. Given our formulation, this is identical to $\mathbb{E}_{Q_0^*|\omega_{1:t}}\left[\max_{a\in\mathcal{A}_{s_\rho}} \Upsilon_n(s_\rho, a|\omega_{1:t})\right]$ and thus is independent of the agent's action policy and the (future) computation policy for a given knowledge state $\omega_{1:t}$. Furthermore, it remains constant in expectation as the knowledge state expands. The second term in the regret is the maximum static/dynamic action value assuming the agent cannot expand its knowledge state before taking an action.

Definition 4. Given a knowledge state $\omega_{1:t}$, we define **Bayesian simple regret** at state s_ρ as

$$R_f(s_\rho, \omega_{1:t}) = \mathbb{E}\left[\max_a \Upsilon_n(s_\rho, a|\omega_{1:t})\right] - \max_a f(s_\rho, a|\omega_{1:t}),$$

where $f \in \{\phi_n, \psi_n\}$.

Based on this definition, we have the following.

Proposition 2. VOC(ϕ_n)*-greedy and* VOC(ψ_n)*-greedy choose the computation that maximize expected decrease in* $R_{\phi_n}(s_\rho, \omega_{1:t})$ *and* $R_{\psi_n}(s_\rho, \omega_{1:t})$ *respectively.*

The proof is provided in the appendix.

We refer to policies that choose the regret-minimizing computation as being *one-step optimal*. Note that this result is different than what is typically referred to as myopic optimality. Myopia refers to considering the impact of a single computation only, whereas VOC(ψ)-greedy policy accounts for the impact of possible future computations that succeed the immediate action.

Proposition 3. *Given an infinite computation budget,* VOC(ϕ_n)*-greedy and* VOC(ψ_n)*-greedy policies will find the optimal action at the root state.*

Proof sketch. Both policies will perform all computations infinitely many times as shown for the flat setting in Ryzhov et al. [19]. Thus, dynamic and static values at the leaves (ie, for $\Gamma_n(s_\rho)$) will converge to the true optimal values (given by Q^*), so will the downstream values. $\square$

We refer to such policies as being *asymptotically optimal*.

4.1 ALTERNATIVE VOC DEFINITIONS

A common [8, 10, 18, 23] formulation for the value of computation is

$$\mathrm{VOC}'{}_f(s_\rho, \bar{\omega}_{1:k}|\omega_{1:t}) = \mathbb{E}_{\Omega_{1:k}}\left[\max_{a\in\mathcal{A}_{s_\rho}} f(s_\rho, a|\omega_{1:t}\Omega_{1:k}) - f(s_\rho, \alpha|\omega_{1:t}\Omega_{1:k})\right], \qquad (2)$$

where $\alpha := \arg\max_a f(s_\rho, a|\omega_{1:t})$ and f is a value function as before.

The difference between this and Definition 3 is that the second term in VOC' conditions f also on $\Omega_{1:k}$. This might seems intuitively correct. VOC' is positive if and only if the policy at s_ρ changes with some probability, that is, $P(\arg\max_{a\in\mathcal{A}_{s_\rho}} f(s_\rho, a|\omega_{1:t}\Omega_{1:k}) \neq \alpha) > 0$. However, this approach can be too myopic as it often takes multiple computations for the policy to change [8]. Note that, this is particularly troublesome for static values ($f = \phi_n$), which commonly arise in methods such as UCT that estimate mean returns of rollouts.

Proposition 4. $\mathrm{VOC}'(\phi_n)$*-greedy is neither one-step optimal nor asymptotically optimal.*

By contrast, dynamic value functions escape this problem.

Proposition 5. *For any* $\bar{\omega}_{1:k}$ *and* $\omega_{1:t}$ *we have* $\mathrm{VOC}'{}_{\psi_n}(s_\rho, \bar{\omega}_{1:k}|\omega_{1:t}) = \mathrm{VOC}_{\psi_n}(s_\rho, \bar{\omega}_{1:k}|\omega_{1:t})$.

Both propositions are proved in the Appendix.

5 VALUE OF COMPUTATION IN MCTS

We now introduce a MCTS method based on VOC-greedy policies we introduced. For this, as done in other information/computation-value-based MCTS methods [8, 23], we utilize UCT as a "base" policy—meaning we call UCT as a subroutine to draw samples from leaf nodes. Because UCT is adaptive, these samples will be drawn from a non-stationary stochastic process in practice; yet, we will treat them as being i.i.d. .

We introduce the model informally, and provide the exact formulas and a pseudocode in the Appendix. We assume no discounting, i.e., $\gamma = 1$, and zero immediate rewards within n steps of the root node for simplicity here, though as we show in the Appendix, the results trivially generalize.

We assume $Q_0^* \sim \mathcal{N}(\mu_0, \Sigma_0)$ where μ_0 is a prior mean vector and Σ_0 is a prior covariance matrix. We assume both these quantities are known—but it is possible to also assume a Wishart prior over Σ_0 or to employ optimization methods from the Gaussian process literature (e.g., maximizing the likelihood function via gradient descent). We assume computations return evaluations of Q_0^* with added Normal noise with known parameters. Then, the posterior value function $Q_0^*|\omega_{1:t}$ can be computed in $\mathcal{O}(t)$ for an isotropic prior covariance, in $\mathcal{O}(tm^2)$ using recursive update rules for multivariate Normal priors , where $m = |Q_0^*|$ is the number of leaf nodes, or in $\mathcal{O}(t^3)$ using Gaussian process priors. We omit the exact form of the posterior distribution here as it is a standard result.

For computing the $\mathrm{VOC}(\phi_n)$-greedy policy we need to evaluate how the expected values at the leaves change with a candidate computation $\bar{\omega} = (s, a)$, i.e., $\mathbb{E}_\Omega[\mathbb{E}_{Q_0^*|\omega_{1:t}\Omega}[Q_0^*|\omega_{1:t}\Omega]]$ where $\Omega = (\bar{\omega}, O_{\bar{\omega}t+1})$. Note that, $O_{\bar{\omega}t+1}$ conditioned on $\omega_{1:t}$, gives the posterior predictive distribution for rollout returns from (s, a), and is normally distributed. Thus, $\mathbb{E}_{Q_0^*|\omega_{1:t}\Omega}[Q_0^*|\omega_{1:t}\Omega]$ is a multivariate random Normal variable of dimension m. The maximum of this variable, i.e. $\max \mathbb{E}_{Q_0^*|\omega_{1:t}\Omega}[Q_0^*|\omega_{1:t}\Omega]$, is a piecewise linear function in $O_{\bar{\omega}t+1}$ and thus its expectation can be computed exactly in $\mathcal{O}(m^2 \log m)$ as shown in Frazier et al. [5]. If an isotropic prior covariance is assumed, the computations simplify greatly as VOC_{ϕ_n} reduces to the expectation of a truncated univariate normal distribution, can be computed in $\mathcal{O}(1)$ given the posterior distributions and the leaf node with the highest expected value. If the transitions are stochastic, then the same method can be utilized whether the covariance is isotropic or anisotropic, with an extra averaging step over transition probabilities at each node, increasing the computational costs.

Computing the $\mathrm{VOC}(\psi_n)$-policy is much harder on the other hand, even for a deterministic state-transition function, because we need to calculate the expected maximum of possibly correlated random variables, $\mathbb{E}[\max Q_0^*|\omega_{1:t}]$. One could resort to Monte Carlo sampling. Alternatively, assuming an isotropic prior over leaf values, we can obtain the following by adapting a bound on expected maximum of random variables [12, 15]:

$$\mathbb{E}[\max Q_0^*|\omega_{1:t}] \leq \lambda_{s_\rho t} := c + \sum_{(s',a')\in\Gamma_n(s_\rho)} \left[(\sigma_{s'a't})^2 F_{s'a't}(c) + (\mu_{s'a't} - c)[1 - F_{s'a't}(c)]\right]$$

where $\mu_{s'a't}$ and $\sigma_{s'a't}$ are posterior mean and variances, that is $Q_0^*(s', a')|\omega_{1:t} \sim \mathcal{N}(\mu_{s'a't}, (\sigma_{s'a't})^2)$, $F_{s'a't}$ is the CDF of $Q_0^*(s', a')|\omega_{1:t}$, and c is a real number. The tightest bound is realized for a c that satisfies

$\sum_{(s',a')\in\Gamma_n(s_\rho)}[1 - F_{s'a't}(c)] = 1$, which can be found by root-finding methods.

The critical question is then how $\lambda_{s_\rho t}$ changes with an additional sample from (s', a'). For this, we use the local sensitivity, $\partial\lambda_{s_\rho t}/\partial n_{s'a't}$ as a proxy, where $n_{s'a't}$ is the number of samples drawn from (s', a') until time t. We give the closed form equation for this partial derivative along with some of its additional properties in the Appendix. Then we can approximately compute the VOC(ψ_n)-greedy policy by choosing the computation that maximizes the magnitude of $\partial\lambda_{s_\rho t}/\partial n_{s'a't}$. This approach only works if state-transitions are deterministic as it enables us to collapse the root action values into a single $\max$ of leaf values. If the state transitions are stochastic, this is no longer possible as averaging over state transitions probabilities is required. Alternatively, one can sample deterministic transition functions and average $\partial\lambda_{s_\rho t}/\partial n_{s'a't}$ over the samples as an approximation.

Our VOC-greedy MCTS methods address important limitations of VOC$'$-based methods [8, 23]. VOC-greedy does not suffer from the early stopping problem that afflicts VOC$'$-based. It is also less myopic in the sense that it can incorporate the impact of computations that may be performed in the future if dynamic value functions are utilized. Lastly, our proposal extends VOC calculations to non-root actions, as determined by n.

6 EXPERIMENTS

We compare the VOC-greedy policies against UCT [11], VOI-based [8], Bayes UCT [22], and Thompson sampling for MCTS (DNG-MCTS) [2] in two different environments, bandit-trees and peg solitaire, where the environment dynamics are provided to each method.

Bayes UCT computes approximate posterior action values and uses a rule similar to UCT to select child nodes. DNG-MCTS also estimates the posterior action values but instead utilizes Thompson sampling recursively down the tree. We use the same conjugate Normal prior structure for the Bayesian algorithms: VOC-greedy, Bayes UCT, and DNG-MCTS[2]. The prior and the noise parameters are tuned for each method via grid search using the same number of evaluations, as well as the exploration parameters of UCT and VOI-based.

VOI-based, VOC-greedy, and Bayes UCT are hybrid methods, using one set of rules for the top of the search tree and UCT for the rest. We refer to this top part of the tree as the partial search tree (PST). By construction, VOI-based utilizes a PST of height 1. We implement the latter two methods using PSTs of height 4 in bandit-trees and of 2 in peg solitaire. These heights are determined based on the branching factors of the environments and the total computation budgets, such that each leaf node is sampled a few (5-8) times on average. For the experiments we explain next, we tune the hyperparameters of all the policies using grid search.

6.1 BANDIT-TREES

The first environment in which we evaluate the MCTS policies is an MDP composed of a complete binary tree of height d, similar to the setting presented in Tolpin and Shimony [23] but with a deeper tree structure and stochastic transitions. The leaves of the tree are noisy "bandit arms" with unknown distributions. Agents perform "computations" to draw samples from the arms, which is analogous to performing rollouts for evaluating leaf values in MCTS. At each state, the agents select an action from $\mathcal{A} = \{\texttt{LEFT}, \texttt{RIGHT}\}$ (denoting the desired subtree of height $d-1$) and transition there with probability .75 and to the other subtree with probability .25. In Figure 3, we illustrate a bandit tree of height 3.

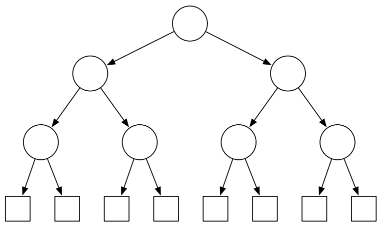


Figure 3: A bandit tree of height 3. Circles denote states, and squares denote bandit arms.

At each time step t, agents sample one of the arms, and update their value estimates at the root state s_ρ. We measure the simple objective[3] regret at state s_ρ at t, which we define as $\max_{a\in\mathcal{A}} Q^*(s_\rho, a) - Q^*(s_\rho, \pi_t(s_\rho))$, for a deterministic policy $\pi_t : s_\rho \to \mathcal{A}$ which depends on the knowledge state acquired by performing t many computations.

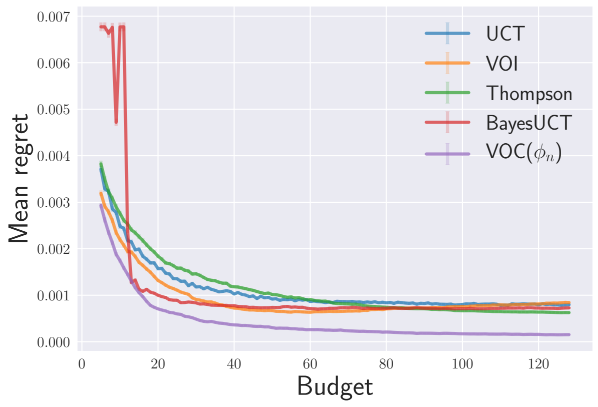
We sample the rewards of the bandit arms from a multivariate Normal distribution, where the covariance is obtained either from a radial basis function or from a white noise kernel. The noise of arms/computations follow an i.i.d. Normal distribution. We provide the exact environment parameters in the Appendix.

[2]In the original paper [2], the authors use Dirichlet-Normal-Gamma priors, but we resort to Normal priors to preserve consistency among all the Bayesian policies.

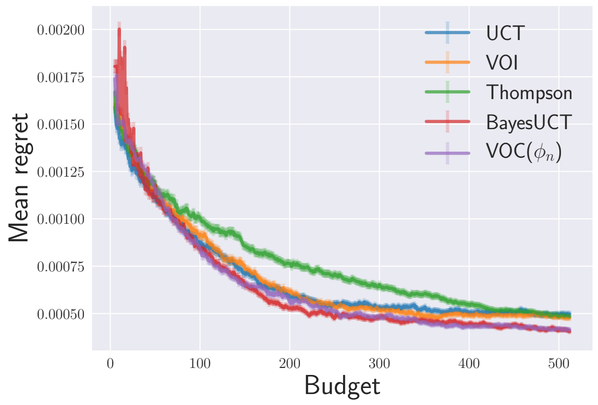
[3]We call it objective regret because it is based on the ground truth (Q^*) as opposed to Bayesian regret, which is based on the estimates of the ground-truth.

Figure 4.a shows the results in the case with correlated bandit arms. These correlations are exploited in our implementation of VOC(ϕ_n)-greedy (via an anisotropic Normal prior over the leaf values of the PST). Note that, we aren't able to incorporate this extra assumption in other Bayesian methods. Bayes UCT utilizes a specific approximation for propagating values up the tree. Thompson sampling would require a prior over all state-actions in the environment which is complicated due to the parent-child dependency among the nodes as well as computationally prohibitive. Because computing the VOC(ψ_n)-greedy policy is very expensive if state transitions are stochastic, we only implement VOC(ϕ_n)-greedy for this environment, but implement both for the next environment.

We see that VOC(ϕ_n)-greedy outperforms all other methods. Note that this is a low-sample density setting: there are $2^7 = 128$ bandit arms and each arm gets sampled on average once as the maximum budget (see x-axis) is 128 as well. This is why many of the policies do not seem to have converged to the optimal solution. The outstanding performance of VOC(ϕ_n)-greedy is partially due to its ability of exploiting correlations. In order to control for this, Figure 4b shows the results in a case in which the bandit rewards are actually uncorrelated (i.e., sampled from an isotropic Normal distribution). As we can see VOC(ϕ_n)-greedy and Bayes UCT performs equally well, and better than the other policies. This implies that the good performance of VOC(ϕ_n)-greedy does not depend wholly on its ability to exploit the correlational structure.



(a) Bandit-trees with correlated bandit arms.



(b) Bandit-trees with uncorrelated bandit arms.

Figure 4: Mean regret as a function of the computation budget for bandit-trees with *correlated* (panel a) *uncorrelated* (panel b) expected bandit rewards, averaged over $10k$ and $5k$ random bandit reward seeds respectively.

6.2 PEG SOLITAIRE

Peg solitaire—also known as Solitaire, Solo, or Solo Noble—is a single-player board game, where the objective for our purposes is to remove as many pegs as possible from the board by making valid moves. We use a 4×4 board, with 9 pegs randomly placed.

In the implementation of VOC-greedy policies, we assume an anisotropic prior over the leaf nodes. As shown in Figure 5, VOC(ϕ_n)-greedy has the best performance for small budget ranges, which is in line with our intuition as ϕ_n is a more accurate valuation of action values for small computation budgets. For large budgets, we see that VOC(ψ_n)-greedy performs as well as Thompson sampling, and better than the rest.

7 DISCUSSION

This paper offers principled ways of assigning values to actions and computations in MCTS. We address important limitations of existing methods by extending computation values to non-immediate actions while accounting for the impact of non-immediate future computations. We show that MCTS methods that greedily maximize computation values have desirable properties and are more sample-efficient in practice than many popular existing methods. The major drawback of our proposal is that computing VOC-greedy policies might be expensive, and may only worth doing so if rollouts (i.e., environment simulations) are computationally expensive. That said, we believe efficient derivates of our methods are possible, for instance by using graph neural networks to directly learn static/dynamic action or computation values.

Practical applications aside, the study of computation values might provide tools for a better understanding of MCTS policies, for instance, by providing notions of regret and optimality for computations, similar to what already exists for actions (i.e., Q^*).

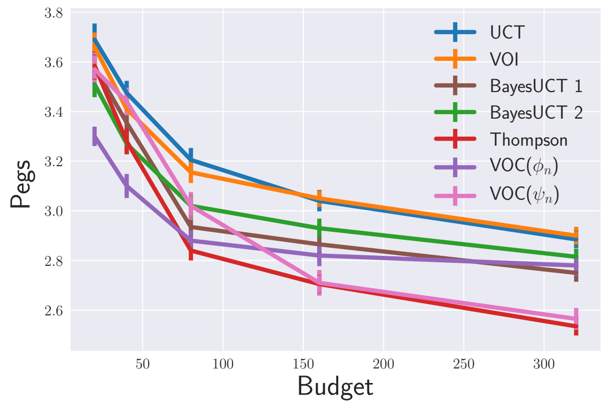


Figure 5: The average number of pegs remaining on the board as a function of the computation budget, averaged over 200 random seeds. The bars denote the mean squared errors.

Acknowledgements. We are grateful to Marcus Hutter, whose feedback significantly improved our presentation of this work. We also thank the anonymous reviewers for their valuable feedback and suggestions. ES conducted this work during a research visit to the Gatsby Unit at University College London and during his studies at the Bernstein Center for Computational Neuroscience Berlin. PD was funded by the Gatsby Charitable Foundation, the Max Planck Society, and the Alexander von Humboldt Foundation.

References

[1] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2-3):235–256, 2002.

[2] Aijun Bai, Feng Wu, and Xiaoping Chen. Bayesian mixture modelling and inference based thompson sampling in monte-carlo tree search. In *Proceedings of the Advances in Neural Information Processing Systems (NIPS-13)*, pages 1646–1654, Lake Tahoe, United States, 2013.

[3] David Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, New York, NY, USA, 2012.

[4] Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *Proceedings of the 5th International Conference on Computers and Games*, CG'06, pages 72–83, Berlin, Heidelberg, 2007. Springer-Verlag.

[5] Peter Frazier, Warren Powell, and Savas Dayanik. The knowledge-gradient policy for correlated normal beliefs. *INFORMS Journal on Computing*, 21 (4):599–613, 2009.

[6] Arthur Guez, David Silver, and Peter Dayan. Scalable and efficient bayes-adaptive reinforcement learning based on monte-carlo tree search. *J. Artif. Int. Res.*, 48(1):841–883, October 2013.

[7] Nicholas Hay and Stuart J. Russell. Metareasoning for monte carlo tree search. Technical Report UCB/EECS-2011-119, EECS Department, University of California, Berkeley, Nov 2011.

[8] Nicholas Hay, Stuart Russell, David Tolpin, and Solomon Eyal Shimony. Selecting computations: Theory and applications. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, UAI'12, pages 346–355, Arlington, Virginia, United States, 2012. AUAI Press.

[9] Ronald A. Howard. Information value theory. *IEEE Transactions on Systems Science and Cybernetics*, SSC-2:22–26, 1966.

[10] Mehdi Keramati, Amir Dezfouli, and Payam Piray. Speed/accuracy trade-off between the habitual and the goal-directed processes. *PLOS Computational Biology*, 7(5):1–21, 05 2011.

[11] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Proceedings of the 17th European Conference on Machine Learning*, ECML'06, pages 282–293, Berlin, Heidelberg, 2006. Springer-Verlag.

[12] T. L. Lai and Herbert Robbins. Maximally dependent random variables. *Proceedings of the National Academy of Sciences of the United States of America*, 73(2):286–288, 1976.

[13] Falk Lieder, Dillon Plunkett, Jessica B. Hamrick, Stuart J. Russell, Nicholas J. Hay, and Thomas L. Griffiths. Algorithm selection by rational metareasoning as a model of human strategy selection. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 2870–2878, Cambridge, MA, USA, 2014. MIT Press.

[14] Christopher H. Lin, Andrey Kolobov, Ece Kamar, and Eric Horvitz. Metareasoning for planning under uncertainty. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 1601–1609. AAAI Press, 2015.

[15] Andrew M. Ross. Computing bounds on the expected maximum of correlated normal variables. *Methodology and Computing in Applied Probability*, 12:111–138, 03 2010.

[16] Omid Madani, Daniel J. Lizotte, and Russell Greiner. The budgeted multi-armed bandit prob-

lem. In John Shawe-Taylor and Yoram Singer, editors, *Learning Theory*, pages 643–645, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[17] Marcelo G. Mattar and Nathaniel D. Daw. Prioritized memory access explains planning and hippocampal replay. *Nature Neuroscience*, 21(11): 1609–1617, 2018.

[18] S. Russell and E. Wefald. *Do the right thing. Studies in limited rationality*. MIT Press, 1991.

[19] Ilya O. Ryzhov, Warren B. Powell, and Peter I. Frazier. The knowledge gradient algorithm for a general class of online learning problems. *Operations Research*, 60(1):180–195, 2012.

[20] Can Eren Sezener, Amir Dezfouli, and Mehdi Keramati. Optimizing the depth and the direction of prospective planning using information values. *PLOS Computational Biology*, 15(3):1–21, 03 2019.

[21] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, January 2016.

[22] Gerald Tesauro, V T Rajan, and Richard Segal. Bayesian inference in monte-carlo tree search. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, UAI'10, pages 580–588, Arlington, Virginia, United States, 2010. AUAI Press.

[23] David Tolpin and Solomon Eyal Shimony. MCTS based on simple regret. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.*, 2012.

[24] Jian Wu, Matthias Poloczek, Andrew G Wilson, and Peter Frazier. Bayesian optimization with gradients. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5267–5278. Curran Associates, Inc., 2017.