Monte Carlo Tree Search with Boltzmann Exploration

Michael Painter, Mohamed Baioumy, Nick Hawes, Bruno Lacerda
Oxford Robotics Institute
University of Oxford
{mpainter, mohamed, nickh, bruno}@robots.ox.ac.uk

Abstract

Monte-Carlo Tree Search (MCTS) methods, such as Upper Confidence Bound applied to Trees (UCT), are instrumental to automated planning techniques. However, UCT can be slow to explore an optimal action when it initially appears inferior to other actions. Maximum ENtropy Tree-Search (MENTS) incorporates the maximum entropy principle into an MCTS approach, utilising *Boltzmann policies* to sample actions, naturally encouraging more exploration. In this paper, we highlight a major limitation of MENTS: optimal actions for the maximum entropy objective do not necessarily correspond to optimal actions for the original objective. We introduce two algorithms, Boltzmann Tree Search (BTS) and Decaying ENtropy Tree-Search (DENTS), that address these limitations and preserve the benefits of Boltzmann policies, such as allowing actions to be sampled faster by using the Alias method. Our empirical analysis shows that our algorithms show consistent high performance across several benchmark domains, including the game of Go.

1 Introduction

Planning under uncertainty is a core problem in Artificial Intelligence, commonly modelled as a Markov Decision Process (MDP) or variant thereof. MDPs can be solved using dynamic programming techniques to obtain an optimal policy [3]. However, computing a full optimal policy does not scale to large state-spaces, necessitating the use of heuristic solvers [18, 4] and online, sampling-based, planners based on Monte-Carlo Tree-Search (MCTS), such as the Upper Confidence Bound applied to Trees (UCT) algorithm [22].

The UCT search policy is designed to minimise *cumulative regret*, so manages a trade-off between exploration and exploitation. To exploit, UCT often selects the same action on successive trials, which can result in it getting stuck in local optima. Conversely, Maximum ENtropy Tree Search (MENTS) places a greater emphasis on exploration by combining MCTS with techniques from maximum entropy policy optimisation [38, 16, 17]. MENTS jointly maximises cumulative rewards and policy entropy, where a *temperature* parameter controls the weight of the entropy objective. However, MENTS is sensitive to this temperature parameter, and may not converge to the reward maximising policy or require a prohibitively low temperature to do so.

In this work, we consider scenarios where MCTS methods are used with a simulator to plan how an agent should act. We introduce two algorithms for this scenario that address the above limitations. First, we present Boltzmann Tree Search (BTS) which uses a Boltzmann search policy like MENTS, but optimises for reward maximisation only. Secondly, we introduce Decaying ENtropy Tree Search (DENTS), which adds entropy backups to BTS, but is still *consistent* (i.e. it converges to the reward maximising policy in the limit).

The main contributions of this paper are: (1) Demonstrating that the maximum entropy objective used in MENTS can be misaligned with reward maximisation, thus preventing it from converging to the optimal policy; (2) Introducing two new algorithms, BTS and DENTS, which preserve the benefits

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

of using Boltzmann search policies while being as simple to implement as UCT and MENTS, but converge to the reward maximising policy; (3) Analysing MENTS, BTS and DENTS through the lens of *simple regret* to provide theoretical convergence results; (4) Highlighting and demonstrating that the Alias method [35, 34] can be used with stochastic action selection to improve the asymptotic complexity of running a fixed number of trials over existing MCTS algorithms; and (5) Demonstrating the performance improvements of Boltzmann search policies used in BTS and DENTS in benchmark gridworld environments and the game of Go.

2 Background

2.1 Markov Decision Processes

We define a (finite-horizon) MDP as a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, R, H)$, where $\mathcal{S}$ is the set of states; $\mathcal{A}$ is the set of actions; $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition function where $p(s'|s, a,)$ is the probability of moving to state s' given that action a was taken in state s; $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function; and $H \in \mathbb{N}$ is the finite horizon. Let $\text{Succ}(s, a)$ denote the set of successor states of (s, a), i.e. $\text{Succ}(s, a) = \{s' \in \mathcal{S} \mid p(s'|s, a) > 0\}$.

A policy π maps a state and timestep to a distribution over actions, and we denote the probability of executing a at state s and timestep t as $\pi(a|s, t)$. Let s_t denote the state after t time-steps and a_t the action selected at s_t, according to π. The expected value V^π and expected state-action value Q^π of π are defined as:

$$V^\pi(s, t) = \mathbb{E}_\pi \left[\sum_{i=t}^{H} R\left(s_i, a_i\right) \Big| s_t = s \right], \tag{1}$$

$$Q^\pi(s, a, t) = R(s, a) + \mathbb{E}_{s' \sim p(\cdot|s,a)}[V^\pi(s', t + 1)]. \tag{2}$$

The goal is to find the *optimal policy* π^* with the maximum expected reward: $\pi^* = \arg\max_\pi V^\pi$. The optimal value functions are then defined as $V^* = V^{\pi^*}, Q^* = Q^{\pi^*}$. For an MDP, there always exists an optimal policy π^* which is deterministic [24].

2.2 Maximum entropy policy optimization

In planning and reinforcement learning, the agent usually aims to maximise the expected sum of rewards. In maximum entropy policy optimisation, the objective is augmented with the expected entropy of the policy [16, 38]. Formally, this is expressed as:

$$V^\pi_{\text{sft}}(s, t) = \mathbb{E}_\pi \left[\sum_{i=t}^{H} R\left(s_i, a_i\right) + \alpha \mathcal{H}\left(\pi\left(\cdot|s_i, i\right)\right) \Big| s_t = s \right], \tag{3}$$

where $\alpha \geq 0$ is a temperature parameter, and $\mathcal{H}$ is the Shannon entropy function. The temperature determines the relative importance of the entropy against the reward and thus controls the stochasticity of the optimal policy. The conventional reward maximisation objective can be recovered by setting $\alpha = 0$.

An optimal value function for maximum entropy optimization is obtained using the *soft* Bellman optimality equations [17]:

$$Q^*_{\text{sft}}(s, a, t) = R(s, a) + \mathbb{E}_{s' \sim p(\cdot|s,a)} \left[V^*_{\text{sft}}(s', t) \right], \tag{4}$$

$$V^*_{\text{sft}}(s, t) = \alpha \log \sum_{a \in \mathcal{A}} \exp(Q^*_{\text{sft}}(s, a, t)/\alpha), \tag{5}$$

which corresponds to a standard Bellman backup, with the $\max$ replaced by a *softmax*, shown in Equation (5). The optimal soft policy $\pi^*_{\text{sft}} = \arg\max_\pi V^\pi_{\text{sft}}$ can be computed directly [26] as follows:

$$\pi^*_{\text{sft}}(a|s, t) = \exp((Q^*_{\text{sft}}(s, a, t) - V^*_{\text{sft}}(s, t))/\alpha). \tag{6}$$

Note that the soft policy is always stochastic for any $\alpha > 0$. Henceforth, we will use *soft value* to refer to value functions indexed with 'sft', *(optimal) soft policy* to refer to policies of the form given in Equation (6), and *standard value* and *(optimal) standard policy* for values and policies of the form given in Section 2.1, unless it is clear from the context.

For the remainder of this paper we will drop the timestep t from policies and value functions to simplify notation.

2.3 Monte-Carlo tree search

MCTS methods build a search tree $\mathcal{T}$ using Monte-Carlo trials. Each trial is split into two phases: starting from the root node, actions are chosen according to a *search policy* and states sampled from the transition distribution until the first state not in $\mathcal{T}$ is reached. A new node is added to $\mathcal{T}$ and its value is initialised using some function V^{init}, often using a *rollout policy* to select actions until the time horizon H is reached. In the second phase, the return for the trial is back-propagated up (or 'backed up') the tree to update the values of nodes in $\mathcal{T}$. For a reader unfamiliar with MCTS, we refer to [6] for a review of the MCTS literature, as many variants of MCTS exist and may vary from our description.

Two critical choices in designing an MCTS algorithm are the search policy (which needs to balance exploration and exploitation) and the backups (how values are updated). MCTS algorithms are often designed to achieve *consistency* (i.e. convergence to the optimal action in the limit), which implies that running more trials will increase the probability that the optimal action is recommended.

To simplify notation we assume that each node in the search tree corresponds to a unique state, so we may represent nodes using states. Our algorithms and results do not make use of this assumption, and generalise to when this assumption does not hold.

UCT UCT [22] applies the upper confidence bound (UCB) in its search policy to balance exploration and exploitation. The nth trial of UCT operates as follows: let $\mathcal{T}$ be the current search tree and let $\tau = (s_0, a_0, ..., a_{h-1}, s_h)$ denote the trajectory of the nth trial, where $s_h \notin \mathcal{T}$ or $h = H$. At each node s_t the UCT search policy π_{UCT} will select a random action that has not previously been selected, otherwise, it will select the action with maximum UCB value:

$$\pi_{\text{UCT}}(s) = \max_{a \in \mathcal{A}} \bar{Q}(s,a) + c\sqrt{\frac{\log N(s)}{N(s,a)}}, \tag{7}$$

where, $\bar{Q}(s,a)$ is the current empirical Q-value estimate, $N(s)$ (and $N(s,a)$) is how many times s has been visited (and action a selected) and c is an exploration parameter. Then, s_h is added to the tree: $\mathcal{T} \leftarrow \{s_h\} \cup \mathcal{T}$. The backup consists of updating empirical estimates for $t = h-1, ..., 0$:

$$\bar{Q}(s_t, a_t) \leftarrow \bar{Q}(s_t, a_t) + \frac{\bar{R}(t) - \bar{Q}(s_t, a_t)}{N(s_t, a_t) + 1}, \tag{8}$$

where $\bar{R}(t) = V^{\text{init}}(s_h) + \sum_{i=t}^{h-1} R(s_i, a_i)$, and $V^{\text{init}}(s_h) = \sum_{i=h}^{H} R(s_i, a_i)$ if using a rollout policy.

MENTS MENTS [37] combines maximum entropy policy optimization [16, 38] with MCTS. Algorithmically, it is similar to UCT. The two differences are: (1) the search policy follows a stochastic Boltzmann policy, and (2) it uses soft values that are updated with dynamic programming backups. The MENTS search policy π_{MENTS} is given by:

$$\pi_{\text{MENTS}}(a|s) = (1 - \lambda_s)\rho_{\text{MENTS}}(a|s) + \frac{\lambda_s}{|\mathcal{A}|}, \tag{9}$$

$$\rho_{\text{MENTS}}(a|s) = \exp\left(\frac{1}{\alpha}\left(\hat{Q}_{\text{sft}}(s,a) - \hat{V}_{\text{sft}}(s)\right)\right) \tag{10}$$

where $\lambda_s = \min(1, \epsilon/\log(e + N(s)))$, $\epsilon \in (0, \infty)$ is an exploration parameter and $\hat{V}_{\text{sft}}(s)$ (and $\hat{Q}_{\text{sft}}(s,a)$) are the current soft (Q-)value estimates. The soft value of the new node is initialised $\hat{V}_{\text{sft}}(s_h) \leftarrow V^{\text{init}}(s_h)$ and the soft values are updated with backups for $t = h-1, ..., 0$:

$$\hat{Q}_{\text{sft}}(s_t, a_t) \leftarrow R(s_t, a_t) + \sum_{s' \in \text{Succ}(s,a)} \left(\frac{N(s')}{N(s_t, a_t)} \hat{V}_{\text{sft}}(s')\right), \tag{11}$$

$$\hat{V}_{\text{sft}}(s_t) \leftarrow \alpha \log \sum_{a \in \mathcal{A}} \exp\left(\frac{1}{\alpha} \hat{Q}_{\text{sft}}(s_t, a)\right). \tag{12}$$

Each $\hat{Q}_{\text{sft}}(s,a)$ is initialised using another function $Q^{\text{init}}_{\text{sft}}(s,a)$ (but is typically zero).

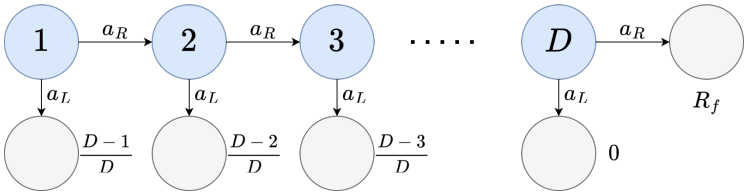


Figure 1: An illustration of the *(modified) D-chain problem*, where 1 is the starting state, transitions are deterministic and values next to states represent rewards for arriving in that state.

2.4 Simple regret

UCB [1] is frequently used in MCTS methods to minimise *cumulative regret* during the tree search. Cumulative regret is most appropriate in scenarios where the actions taken during tree search have an associated real-world cost. However, MCTS methods often use a simulator during the tree search, where the only significant real-world cost is associated with taking the recommended action after the tree search. In such scenarios, simple regret [7, 8] is more appropriate for analysing the performance of algorithms, as it only considers the cost of the actions that are actually executed. Under simple regret, algorithms are not penalised for under-exploiting during the search, thus can explore more, which leads to better recommendations by allowing algorithms to confirm that bad actions are indeed of lower value.

We consider the problem of MDP planning as a sequential decision problem, where for each round n:

1. the forecaster algorithm produces a search policy π^n and samples a trajectory $\tau \sim \pi^n$,
2. the environment returns the rewards $R(s_t, a_t)$ for each s_t, a_t pair in τ,
3. the forecaster algorithm produces a recommendation policy ψ^n,
4. if environment sends stop signal, then end, else return to step 1.

The *simple regret* of the forecaster on it's nth round is then:

$$\text{reg}(s, \psi^n) = V^*(s) - V^{\psi^n}(s). \tag{13}$$

In MENTS the recommendation policy suggested in [37] can be written:

$$\psi_{\text{MENTS}}(s) = \arg\max_{a \in \mathcal{A}} \hat{Q}_{\text{sft}}(s, a). \tag{14}$$

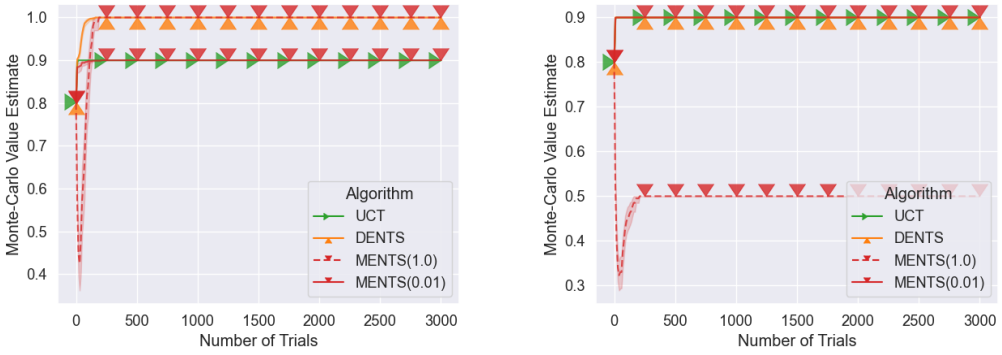
We can now formally define *consistency*: an algorithm is *consistent* if and only if its recommendation policy ψ^n converges to an expected simple regret of zero: $\mathbb{E}[\text{reg}(s, \psi^n)] \to 0$ as $n \to \infty$. Note that because we are considering randomised algorithms, there is a distribution over the possible recommendation policies that could have been produced. If a policy has a simple regret of zero then it implies it is an optimal policy.

3 Limitations of prior MCTS methods

In this section we use the *D-chain problem* introduced in [9] (Figure 1) to highlight the limitations of UCT and MENTS. In the D-chain problem, when an agent chooses action a_L, from some state d, it moves to an absorbing state and receives a reward of $(D - d)/D$. In state D, action a_R corresponds to an absorbing state with reward $R_f = 1$. The optimal standard policy always selects action a_R.

In the 10-chain problem ($D = 10$), UCT will recommend action a_L from state 1 (Figure 2a). UCT requires $\Omega(\exp(...\exp(1)...))$ many trials (D composed exponential functions) to recommend the optimal policy that reaches the reward of $R_f = 1$ [9]. This highlights the first limitation mentioned in Section 1: UCT quickly disregards action a_R at the initial state, to exploit the reward of 0.9.

When MENTS is run on the 10-chain problem, with the help of the entropy term it quickly finds the final reward of $R_f = 1$ (Figure 2a). However, consider the *modified 10-chain* with $R_f = 1/2$ instead. Repeated applications of Equations (4) and (5) for $\alpha = 1$ gives the optimal soft values of $Q^*_{\text{sft}}(1, a_R) = \log(\exp(1/2) + \sum_{i=0}^{8} \exp(i/10)) \approx 2.74$ and $Q^*_{\text{sft}}(1, a_L) = 0.9$. So in the modified



(a) 10-chain. (b) Modified 10-chain.

Figure 2: A comparison of MENTS, DENTS and UCT when run on the (modified) 10-chain.

10-chain problem, we have $Q^*_{\text{sft}}(1, a_R) > Q^*_{\text{sft}}(1, a_L)$ with $\alpha = 1$, whereas $Q^*(1, a_R) < Q^*(1, a_L)$. Thus, when MENTS converges, it will recommend the wrong action with respect to the standard objective (Figure 2b), i.e. it is not consistent. The modified 10-chain is an example of Proposition 3.1, which states that MENTS will not always converge to the standard optimal policy.

Proposition 3.1. *There exists an MDP $\mathcal{M}$ and temperature α such that $\mathbb{E}[\text{reg}(s_0, \psi^n_{\text{MENTS}})] \not\to 0$ as $n \to \infty$. That is, MENTS is not consistent.*

Proof. Proof is by example with $\alpha = 1$ in the modified 10-chain (Figure 1). □

We can reduce the value of α to decrease the importance of entropy in the soft objective $\mathbb{E}\left[R\left(s_t, a_t\right) + \alpha\mathcal{H}\left(\pi\left(\cdot|s_t\right)\right)\right]$. If α is small enough, then MENTS recommendations can converge to the optimal standard policy (Theorem 3.2). Hence, MENTS with a low temperature can solve the modified 10-chain problem (Figure 2b). However, in practice, a low temperature will often cause MENTS to not sufficiently explore, as demonstrated in the original D-chain (Figure 2a).

Theorem 3.2. *For any MDP $\mathcal{M}$, after running n trials of the MENTS algorithm with $\alpha \leq \Delta_{\mathcal{M}}/3H\log|\mathcal{A}|$, there exists constants $C, k > 0$ such that: $\mathbb{E}[\text{reg}(s_0, \psi_{\text{MENTS}})] \leq C\exp(-kn)$, where $\Delta_{\mathcal{M}} = \min\{Q^*(s,a) - Q^*(s,a') | Q^*(s,a) \neq Q^*(s,a'), s \in \mathcal{S}, a, a' \in \mathcal{A}, t \in \mathbb{N}\}$.*

Proof outline. We can show $\hat{Q}_{\text{sft}}(s,a) \xrightarrow{p} Q^*_{\text{sft}}(s,a)$ (Corollary E.12.1) similarly to Thoerem 4.2. The bound on α is required to ensure that $\pi^*(s_0) = \pi^*_{\text{sft}}(s_0)$. □

In conclusion, similar MDPs can require vastly different temperatures for MENTS to be effective. We discuss MENTS sensitivity to the temperature parameter further in Appendix D.2, and demonstrate this parameter sensitivity in the Frozen Lake environment (Section 5.1) in Figure 27 in the appendix.

4 Boltzmann search

We now introduce two algorithms that utilise Boltzmann search policies similar to MENTS and admit bounded simple regrets that converge to zero without restrictive constraints on parameters. Thus, they do not suffer from sensitivity to parameter selection that MENTS does. Both algorithms use action selection and value backups that are easy to implement and use. We designed these algorithms with consistency in mind, which in practice, means that if we run more trials then we (with high probability) will recommend a better solution (note that Proposition 3.1 implies that this is not always the case for MENTS).

4.1 Boltzmann Tree Search

Our first approach, put simply, replaces the use of soft values in MENTS with *Bellman* values. We call this algorithm *Boltzmann Tree Search* (BTS). BTS promotes exploration through the stochastic

Boltzmann search policy, like MENTS, while using backups that optimise for the standard objective, like UCT. The search policy π_{BTS} and backups for the nth trial are given by:

$$\pi_{\text{BTS}}(a|s) = (1 - \lambda_s)\rho_{\text{BTS}}(a|s) + \frac{\lambda_s}{|\mathcal{A}|}, \tag{15}$$

$$\rho_{\text{BTS}}(a|s) \propto \exp\left(\frac{1}{\alpha}\left(\hat{Q}(s,a)\right)\right), \tag{16}$$

$$\hat{Q}(s_t, a_t) \leftarrow R(s_t, a_t) + \sum_{s' \in \text{Succ}(s_t, a_t)} \left(\frac{N(s')}{N(s_t, a_t)}\hat{V}(s')\right), \tag{17}$$

$$\hat{V}(s_t) \leftarrow \max_{a \in \mathcal{A}} \hat{Q}(s_t, a), \tag{18}$$

for $t = h - 1, ..., 0$, where $\hat{V}$ and $\hat{Q}$ are the current Bellman (Q-)value estimates, $\lambda_s = \min(1, \epsilon/\log(e + N(s)))$, $\epsilon \in (0, \infty)$ is an exploration parameter and α is a search temperature (unrelated to entropy). Each $\hat{V}(s)$ and $\hat{Q}(s,a)$ are initialised using V^{init} and Q^{init} functions similarly to MENTS. The Bellman values are used for recommendations:

$$\psi_{\text{BTS}}(s) = \arg\max_{a \in \mathcal{A}} \hat{Q}(s,a). \tag{19}$$

By using Bellman backups, we can guarantee that the BTS recommendation policy converges to the optimal standard policy for any temperature α, given enough time. In other words, BTS is consistent.

Theorem 4.1. *For any MDP $\mathcal{M}$, after running n trials of the BTS algorithm with a root node of s_0, there exists constants $C, k > 0$ such that for all $\varepsilon > 0$ we have $\mathbb{E}[\text{reg}(s_0, \psi_{\text{BTS}})] \leq C\exp(-kn)$, and also $\hat{V}(s_0) \xrightarrow{p} V^*(s_0)$ as $n \to \infty$.*

Proof outline. This result is a special case of Theorem 4.2 by setting $\beta(m) = 0$. □

4.2 Decaying Entropy Tree Search

Secondly, we present Decaying ENtropy Tree Search (DENTS), which can effectively interpolate between the MENTS and BTS algorithms. DENTS also uses the dynamic programming backups from equations (17) and (18), but adds an *entropy backup*. The *entropy values* are weighted by a bounded non-negative function $\beta(N(s))$ in the DENTS search policy π_{DENTS}:

$$\pi_{\text{DENTS}}(a|s) = (1 - \lambda_s)\rho_{\text{DENTS}}(a|s) + \frac{\lambda_s}{|\mathcal{A}|}, \tag{20}$$

$$\rho_{\text{DENTS}}(a|s) \propto \exp\left(\frac{1}{\alpha}\left(\hat{Q}(s,a) + \beta(N(s))\mathcal{H}_Q(s,a)\right)\right), \tag{21}$$

$$\mathcal{H}_V(s_t) \leftarrow \mathcal{H}(\pi_{\text{DENTS}}(\cdot|s_t)) + \sum_{a \in \mathcal{A}} \pi_{\text{DENTS}}(a|s_t)\mathcal{H}_Q(s_t, a), \tag{22}$$

$$\mathcal{H}_Q(s_t, a_t) \leftarrow \sum_{s' \in \text{Succ}(s_t, a_t)} \frac{N(s')}{N(s_t, a_t)}\mathcal{H}_V(s'), \tag{23}$$

for $t = h - 1, ..., 0$, where $\mathcal{H}_V(s)$ and $\mathcal{H}_Q(s,a)$ are the *entropy values* of the search policy rooted at s and (s,a) respectively, and α, λ_s are the same as for BTS, as described in Section 4.1. Initial values are the same as Section 4.1, and the entropy values are initialised to zero. In DENTS we can view $\hat{Q}(s,a) + \beta(N(s))\mathcal{H}_Q(s,a)$ as a soft value for (s,a). Hence, by setting $\beta(m) = \alpha$, the DENTS search will mimic the MENTS search (demonstrated in Appendix D.4), and if $\beta(m) = 0$ then the algorithm reduces to the BTS algorithm. By using a decaying function for β we amplify values using entropy as an exploration bonus early in the search while allowing for more exploitation later. Recommendations still use Bellman values:

$$\psi_{\text{DENTS}}(s) = \arg\max_{a \in \mathcal{A}} \hat{Q}(s,a). \tag{24}$$

Because the recommendation policy ψ_{DENTS} uses the Bellman values, we can guarantee that it will converge to the optimal standard policy, and is consistent for any β.

Theorem 4.2. *For any MDP $\mathcal{M}$, after running n trials of the DENTS algorithm with a root node of s_0, if β is a bounded function, then there exists constants $C, k > 0$ such that for all $\varepsilon > 0$ we have $\mathbb{E}[\text{reg}(s_0, \psi_{\text{DENTS}})] \leq C \exp(-kn)$, and also $\hat{V}(s_0) \xrightarrow{p} V^*(s_0)$ as $n \to \infty$.*

Proof outline. Let $0 < \eta < \pi_{\text{DENTS}}(a|s)$ for all s, a, which exists because $\exp(\cdot) > 0$ and $1/|\mathcal{A}| > 0$ in Equation 20 (Lemma E.1), which can be used with Hoeffding bounds to show for appropriate constants and any event E that $\{\Pr(E) \leq C_0 \exp(-k_0 \varepsilon^2 N(s_t))\}$ iff $\{\Pr(E) \leq C_1 \exp(-k_1 \varepsilon^2 N(s_t, a_t))\}$ iff $\{\Pr(E) \leq C_2 \exp(-k_2 \varepsilon^2 N(s_{t+1}))\}$. We can then show $\Pr(|\hat{Q}(s_0, a) - Q^*(s_0, a)| > \varepsilon) < C_3 \exp\left(-k_3 \varepsilon^2 n\right)$ by induction, where the base case $\hat{V}(s_{H+1}) = V^*(s_{H+1}) = 0$ holds vacuously (Lemmas E.10, E.16 and Theorem E.17). Let $\Delta_{\mathcal{M}}$ be a small constant (Equation (122)) such that $\forall s, a. |\hat{Q}(s_0, a) - Q^*(s_0, a)| \leq \Delta_{\mathcal{M}}/2 \implies \arg\max_a \hat{Q}(s, a) = \arg\max_a Q^*(s, a)$. Setting $\varepsilon = \Delta_{\mathcal{M}}/2$ gives a bound on $\Pr(\psi_{\text{DENTS}} \neq \pi^*)$, which can then be used in the definition of simple regret to give the result. □

4.3 Using the Alias method

The Alias method [35, 34] can be used to sample from a categorical distribution with m categories in $O(1)$ time, with a preprocessing step of $O(m)$ time. Given any stochastic search policy, we can sample actions in amortised $O(1)$ time, by computing an alias table every $|\mathcal{A}|$ visits to a node, and then sampling from that table. Note that when using the Alias method we are making a trade off between using the most up to date policy and the speed of sampling actions.

In Appendix C.1 we discuss this idea in more detail, and give an informal analysis of the complexity to run n trials. BTS, DENTS and MENTS can run n trials in $O(n(H \log(|\mathcal{A}|) + |\mathcal{A}|))$ time when using the Alias method, as opposed to the typical complexity of $O(nH|\mathcal{A}|)$.

4.4 Limitations and benefits

The main limitations of BTS and DENTS are as follows: (1) the DENTS decay function β can be non-trivial to set and tune; (2) the focus on simple regret and exploration means they are not appropriate to use when actions taken during the tree search/planning phase have a real-world cost; (3) the backups implemented directly as presented above are computationally more costly than computing the average returns that UCT uses; (4) when it is desirable for an agent to follow the maximum entropy policy, then MENTS would be preferable, for example if the agent needs to explore to learn and discover an unknown environment.

The main benefits of using a stochastic policy for action selection are: (1) they allow the Alias method (Section 4.3) to be used to speed up trials; (2) they naturally encourage exploration as actions are sampled randomly, which is useful for discovering sparse or delayed rewards and for confirming that actions with low values do in fact have a low value; and (3) the entropy of a stochastic policy can be computed and used as an exploration bonus. In Appendix C.3 we summarise and compare the differences between the algorithms considered in this work in more detail.

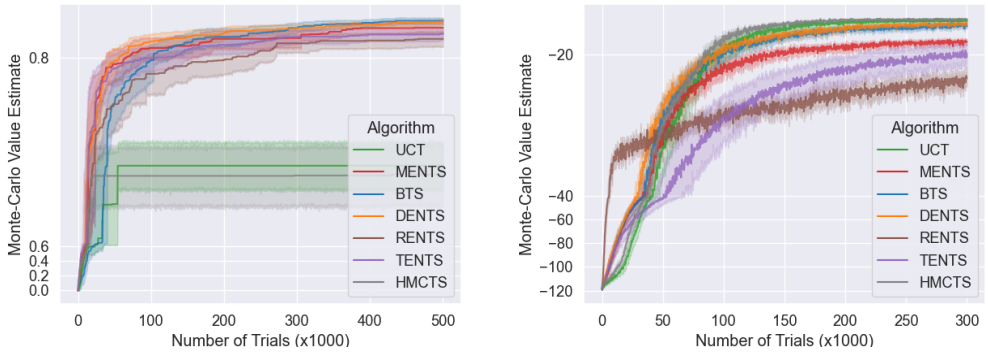
5 Results

This section compares the proposed BTS and DENTS against MENTS and UCT on a set of goal-based MDPs and in the game of Go. For additional baselines, we also compare with the RENTS and TENTS algorithms [10], which use *relative* and *Tsallis* entropy in place of Shannon entropy respectively, and the H-MCTS algorithm [20] which combines UCT and Sequential Halving.

5.1 Gridworlds

To evaluate an algorithm with search tree $\mathcal{T}$, we complete the partial recommendation policy ψ_{alg} as follows:

$$\psi(a|s) = \begin{cases} 1 & \text{if } s \in \mathcal{T} \text{ and } a = \psi_{\text{alg}}(s), \\ 0 & \text{if } s \in \mathcal{T} \text{ and } a \neq \psi_{\text{alg}}(s), \\ \frac{1}{|\mathcal{A}|} & \text{otherwise.} \end{cases} \tag{25}$$



(a) 8x12 Frozen Lake.

(b) 6x6 Sailing Problem.

Figure 3: Results for gridworld environments. Further results are given in Appendix D.3.

We sample a number of trajectories from ψ, and take the average return to estimate V^{ψ}. Although we are evaluating the algorithms in an *offline planning* setting, it still indicates how the algorithms perform in an *online* setting where we interleave planning in simulation with letting the agent act.

5.1.1 Domains

To validate our approach, we use the Frozen Lake environment [5], and the Sailing Problem [28], commonly used to evaluate tree search algorithms [28, 22, 33, 13]. We chose these environments to compare our algorithms in a domain with a sparse and dense reward respectively.

The *(Deterministic) Frozen Lake* is a grid world environment with one goal state. The agent can move in any cardinal direction at each time step, and walking into a wall leaves the agent in the same location. Trap states exist where the agent falls into a hole and the trial ends. If the agent arrives at the goal state after t timesteps, then a reward of 0.99^t is received.

The *Sailing Problem* is a grid world environment with one goal state, at the opposite corner to the starting location of the agent. The agent has 8 different actions to travel each of the 8 adjacent states. In each state, the wind is blowing in a given direction and will stochastically change after every transition. The agent cannot sail directly into the wind. The cost of each action depends on the *tack*, the angle between the direction of the agent's travel and the wind.

5.1.2 Results

We used an 8x12 Frozen Lake environment and a 6x6 Sailing Problem for evaluation, more environment details are given in Appendix D.1. Parameters were selected using a hyper-parameter search (Appendix D.3). Each algorithm is run 25 times on each environment and evaluated every 250 trials using 250 trajectories. A horizon of 100 was used for Frozen Lake and 50 for the Sailing Problem.

In Frozen Lake (Figure 3a), entropy proved to be a useful exploration bonus for the *sparse reward*. Values in UCT and BTS remain at zero until a trial successfully reaches the goal. However, entropy guides agents to avoid trap states, where the entropy is zero. DENTS was able to perform similarly to MENTS, and BTS was able to improve its policy over time more than UCT.

In the Sailing Problem (Figure 3b) UCT performs well due to the dense reward. BTS and DENTS also manage to keep up with UCT. MENTS and TENTS appear to be slightly hindered by entropy in this environment. The relative entropy encourages RENTS to pick the same actions over time, so it tends to pick a direction and stick with it regardless of cost.

Finally, BTS and DENTS were able to perform well in both domains with a sparse and dense reward structure, whereas the existing methods performed better on one than the other, hence making BTS and DENTS good candidates for a general purpose MCTS algorithm.

5.2 Go

For a more challenging domain we ran a round-robin tournament using the game of Go, which has widely motivated the development of MCTS methods [14, 30, 32]. In each match, each algorithm played 50 games as black and 50 as white. *Area scoring* is used to score the games, with a *komi* of 7.5. We used an openly available value network $\tilde{V}$ and policy network $\tilde{\pi}$ from KataGo [36]. Our baseline was the PUCT algorithm [2], as described in Alpha Go Zero [32] using prioritised UCB [29] to utilise the policy neural network. Each algorithm was limited to 5 seconds of compute time per move, allowed to use 32 search threads per move, and had access to 80 Intel Xeon E5-2698V4 CPUs clocked at 2.2GHz, and a single Nvidia V100 GPU on a shared compute cluster.

To use Boltzmann search in Go, we adapted the algorithms to account for an opponent that wishes to minimise the value of a two-player game. This is achieved by appropriately negating values used in the search policy and backups, which is described precisely in Appendix C.2.

Additionally, we found that adapting the algorithms to use average returns (recall Equation (8)) outperformed using Bellman backups for Go (Appendix D.5.1). The Bellman backups were sensitive to and propogated noise from the neural network evaluations. We use the prefix 'AR' to denote the algorithms using average returns, such as AR-DENTS. Full details for these algorithms are given in Appendix B.

5.2.1 Using neural networks with Boltzmann search

This section describes how to use value and policy networks in BTS. Adapting MENTS and DENTS are similar (Appendix C.2). Values can be initialised with the neural networks as $\hat{Q}(s,a) \leftarrow \log \tilde{\pi}(a|s) + B$ and $\hat{V}(s) \leftarrow \tilde{V}(s)$, where B is a constant (adapted from Xiao et al. [37]). With such an initialisation, the initial BTS policy is $\rho_{\text{BTS}}(a|s) \propto \tilde{\pi}(a|s)^{1/\alpha}$. For these experiments we set a value of $B = \frac{-1}{|\mathcal{A}|}\sum_{a\in\mathcal{A}} \log \tilde{\pi}(a|s)$. Additionally, the stochastic search policy naturally lends itself to mixing in a prior policy, so we can replace BTS search policy π_{BTS} (Equation (15)) with $\pi_{\text{BTS,mix}}$:

$$\pi_{\text{BTS,mix}}(a|s) = \lambda_{\tilde{\pi}}\tilde{\pi}(a|s) + (1-\lambda_{\tilde{\pi}})\pi_{\text{BTS}}(a|s) \tag{26}$$

$$= \lambda_{\tilde{\pi}}\tilde{\pi}(a|s) + (1-\lambda_{\tilde{\pi}})(1-\lambda_s)\rho_{\text{BTS}}(a|s) + \frac{(1-\lambda_{\tilde{\pi}})\lambda_s}{|\mathcal{A}|}, \tag{27}$$

where $\lambda_{\tilde{\pi}} = \min(1, \epsilon_{\tilde{\pi}}/\log(e+N(s)))$, and $\epsilon_{\tilde{\pi}} \in (0,\infty)$ controls the weighting for the prior policy.

5.2.2 Results

Results of the round-robin are summarised in Table 1, and we discuss how parameters were selected in Appendix D.5.1. BTS was able to run the most trials per move and beat all of the other algorithms other than DENTS which it drew. We used the optimisations outlined in Appendix C.1 which allowed the Boltzmann search algorithms to run significantly more trials per move than PUCT. BTS and DENTS were able to beat PUCT with results of 57-43 and 58-42 respectively. Using entropy did not seem to have much benefit in these experiments, as can be witnessed by MENTS only beating TENTS, and DENTS drawing 50-50 with BTS. This is likely because the additional exploration provided by entropy is vastly outweighed by utilising the information contained in the neural networks $\tilde{V}$ and $\tilde{\pi}$. Interestingly RENTS had the best performance out of the prior works, losing 43-57 to PUCT, and the use of relative entropy appears to take advantage of a heuristic for Go that the RAVE [15] algorithm used: the value of a move is typically unaffected by other moves on the board.

To validate the strength of our PUCT agent, we also compared it directly with KataGo [36], limiting each algorithm to 1600 trials per move. Our PUCT agent won 61-39 in 9x9 Go, and lost 35-65 in 19x19 Go, suggesting that our PUCT agent is strong enough to provide a meaningful comparison for our other general purpose algorithms. Finally, note that we did not fine-tune the neural networks, so the Boltzmann search algorithms directly used the networks that were trained for use in PUCT.

6 Related work

UCT [22, 23] is a widely used variant of MCTS. Polynomial UCT [2], replaces the logarithmic term in UCB with a polynomial one (such as a square root), which has been further popularised by its use

Black \White	PUCT	AR-M	AR-R	AR-T	AR-B	AR-D	Trials/move
PUCT	-	33-17	27-23	42-8	17-33	15-35	1054
AR-MENTS	12-48	-	13-37	38-12	10-40	12-38	4851
AR-RENTS	20-30	24-26	-	39-11	18-32	14-36	3672
AR-TENTS	8-42	11-39	9-41	-	6-44	10-40	5206
AR-BTS	25-25	35-15	31-19	34-16	-	15-35	5375
AR-DENTS	23-27	36-14	29-21	36-14	15-35	-	4677

Table 1: Results for the Go round-robin tournament. The first column gives the agent playing as black. The final column gives the average trials run per move across the entire round-robin. In the top row, we abbreviate the algorithm names for space.

in Alpha Go and Alpha Zero [30, 32, 31]. Coquelin and Munos introduce the Flat-UCB and BAST algorithms to adapt UCT for the D-chain problem [9]. However, we consider an alternative approach for search in MCTS rather than adapting UCB.

Maximum entropy policy optimization methods are well-known in the reinforcement learning literature [16, 17, 38]. MENTS [37] is the first method to combine the principle of maximum entropy and MCTS. Kozakowski et al. [25] extend MENTS to arrive at Adaptive Entropy Tree Search (ANTS), adapting parameters throughout the search to match a prescribed entropy value. Dam et al. [10] also extend MENTS using *Relative* and *Tsallis entropy* to arrive at the RENTS and TENTS algorithms. Our work is closely related to MENTS, however, we focus on reward maximisation and consider how entropy can be used in MCTS without altering our planning objective.

Bubeck et al. [7, 8] introduce simple regret in the context of multi-armed bandit problems (MABs). They alternate between pulling arms for exploration and outputting a *recommendation*. They show for MABs that a uniform exploration produces an exponential bound on the simple regret of recommendations. We use simple regret, but in the context of sequential decision-making, to analyse the convergence of MCTS algorithms.

Tolpin and Shimony [33] extend simple regret to MDP settings, showing an $O(\exp(-\sqrt{n}))$ bound on simple regret after n trials by adapting UCT. The subsequent work of Hay et al. [19] extends [33] to consider a *metalevel decision problem*, incorporating computation costs into the objective. Pepels et al. [27] introduce a Hybrid MCTS (H-MCTS) motivated by the notion of simple regret. H-MCTS uses a mixture of Sequential Halving [20], and UCT. Feldman et al. [13, 11, 12] introduce the Best Recommendation with Uniform Exploration (BRUE) algorithm. BRUE splits trials up to explicitly focus on exploration and value estimation one at a time. BRUE achieves an exponential bound $O(\exp(-n))$ on simple regret after n trials [13]. Prior work that considers simple regret in MCTS has focused on adaptations to UCT, whereas this work focuses on algorithms that sample actions from Boltzmann distributions, rather than using UCB for action seclection.

7 Conclusion

We considered the recently introduced MENTS algorithm, compared and contrasted it to UCT, and discussed the limitations of both. We introduced two new algorithms, BTS and DENTS, that are consistent, converge to the optimal standard policy, while preserving the benefits that come with using a stochastic Boltzmann search policy. Finally, we compared our algorithms in gridworld environments and Go, demonstrating the performance benefits of utilising the Alias method, that entropy can be a useful exploration bonus with sparse rewards, and more generally, demonstrating the advantage of prioritising exploration in planning, by using Boltzmann search policies.

An interesting area of future work may include investigating good heuristics for setting parameters in BTS and DENTS. We noticed that the best value for the search temperature tended to be the same order of magnitude as the optimal value at the root node $V^*(s_0)$, which suggests a heuristic similar to [21] may be reasonable.

References

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

[2] David Auger, Adrien Couetoux, and Olivier Teytaud. Continuous upper confidence trees with polynomial exploration–consistency. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 194–209. Springer, 2013.

[3] Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, 6(5):679–684, 1957.

[4] Blai Bonet and Hector Geffner. Labeled rtdp: Improving the convergence of real-time dynamic programming. In *ICAPS*, volume 3, pages 12–21, 2003.

[5] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[6] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[7] Sébastien Bubeck, Rémi Munos, and Gilles Stoltz. Pure exploration in multi-armed bandits problems. In *International conference on Algorithmic learning theory*, pages 23–37. Springer, 2009.

[8] Sébastien Bubeck, Rémi Munos, and Gilles Stoltz. Pure exploration in finitely-armed and continuous-armed bandits. *Theoretical Computer Science*, 412(19):1832–1852, 2011.

[9] Pierre-Arnaud Coquelin and Rémi Munos. Bandit algorithms for tree search. In *Uncertainty in Artificial Intelligence*, 2007.

[10] Tuan Q Dam, Carlo D'Eramo, Jan Peters, and Joni Pajarinen. Convex regularization in monte-carlo tree search. In *International Conference on Machine Learning*, pages 2365–2375. PMLR, 2021.

[11] Zohar Feldman and Carmel Domshlak. Monte-carlo planning: Theoretically fast convergence meets practical efficiency. *arXiv preprint arXiv:1309.6828*, 2013.

[12] Zohar Feldman and Carmel Domshlak. On mabs and separation of concerns in monte-carlo planning for mdps. In *Twenty-Fourth International Conference on Automated Planning and Scheduling*, 2014.

[13] Zohar Feldman and Carmel Domshlak. Simple regret optimization in online planning for markov decision processes. *Journal of Artificial Intelligence Research*, 51:165–205, 2014.

[14] Sylvain Gelly and David Silver. Combining online and offline knowledge in uct. In *Proceedings of the 24th international conference on Machine learning*, pages 273–280, 2007.

[15] Sylvain Gelly and David Silver. Monte-carlo tree search and rapid action value estimation in computer go. *Artificial Intelligence*, 175(11):1856–1875, 2011.

[16] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning*, pages 1352–1361. PMLR, 2017.

[17] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[18] Eric A Hansen and Shlomo Zilberstein. Lao*: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, 129(1-2):35–62, 2001.

[19] Nicholas Hay, Stuart Russell, David Tolpin, and Solomon Eyal Shimony. Selecting computations: Theory and applications. *arXiv preprint arXiv:1408.2048*, 2014.

[20] Zohar Karnin, Tomer Koren, and Oren Somekh. Almost optimal exploration in multi-armed bandits. In *International Conference on Machine Learning*, pages 1238–1246. PMLR, 2013.

[21] Thomas Keller and Patrick Eyerich. Prost: Probabilistic planning based on uct. In *Twenty-Second International Conference on Automated Planning and Scheduling*, 2012.

[22] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

[23] Levente Kocsis, Csaba Szepesvári, and Jan Willemson. Improved monte-carlo search. *Univ. Tartu, Estonia, Tech. Rep*, 1, 2006.

[24] Andrey Kolobov. *Planning with Markov Decision Processes: An AI Perspective*, volume 6. Morgan & Claypool Publishers, 2012.

[25] Piotr Kozakowski, Mikołaj Pacek, and Piotr Miłoś. Planning and learning using adaptive entropy tree search. *arXiv preprint arXiv:2102.06808*, 2021.

[26] Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

[27] Tom Pepels, Tristan Cazenave, Mark HM Winands, and Marc Lanctot. Minimizing simple and cumulative regret in monte-carlo tree search. In *Workshop on Computer Games*, pages 1–15. Springer, 2014.

[28] Laurent Péret and Frédérick Garcia. On-line search for solving markov decision processes via heuristic sampling. *learning*, 16:2, 2004.

[29] Christopher D Rosin. Multi-armed bandits with episode context. *Annals of Mathematics and Artificial Intelligence*, 61(3):203–230, 2011.

[30] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[31] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[32] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[33] David Tolpin and Solomon Shimony. Mcts based on simple regret. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, pages 570–576, 2012.

[34] Michael D Vose. A linear algorithm for generating random numbers with a given distribution. *IEEE Transactions on software engineering*, 17(9):972–975, 1991.

[35] Alastair J Walker. New fast method for generating discrete random numbers with arbitrary frequency distributions. *Electronics Letters*, 8(10):127–128, 1974.

[36] David J Wu. Accelerating self-play learning in go. *arXiv preprint arXiv:1902.10565*, 2019.

[37] Chenjun Xiao, Ruitong Huang, Jincheng Mei, Dale Schuurmans, and Martin Müller. Maximum entropy monte-carlo planning. *Advances in Neural Information Processing Systems*, 32, 2019.

[38] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. 2008.

Appendices

These appendices are structured as follows:

- Appendix B gives details on how to adapt the BTS, DENTS, MENTS, RENTS and TENTS algorithms to use average returns backups, rather than dynamic programming backups, to arrive at the AR-BTS, AR-DENTS, AR-MENTS, AR-RENTS and AR-TENTS algorithms respectively.
- Appendix C covers algorithm details that we could not fit into the main paper. Specifically:
  - in Appendix C.1 we discuss the computational complexity of BTS, DENTS and MENTS in more detail, including what can be achieved by using the alias method [35, 34];
  - in Appendix C.2 we give the specific details of how we adapted BTS, DENTS, MENTS, RENTS and TENTS for two-player games and to utilise neural networks;
  - and in Appendix C.3 we summarise the differences between the MCTS algorithms considered in this work in a table.
- Appendix D generally covers additional details about experiments and results. Specifically:
  - in Appendix D.1 we give more specific details about the gridworld envrionments used in our experiments;
  - in Appendix D.2 we have a more detailed discussion about parameter sensitivity, with additional results and discussion using the D-chain environments in Appendix D.2.1 and in the Frozen Lake environment in Appendix D.2.2;
  - in Appendix D.3 we give details for the hyperparameter search used to select parameters in the gridworld domains;
  - in Appendix D.4 we show empirically that DENTS can follow a search policy very similar to the MENTS search policy;
  - and in Appendix D.5 we give some additional details about our Go experiments, including how we selected hyperparameters.
- Appendix E constitutes the theoretical section of this work, beginning by revisiting the MCTS *stochastic process*, introducing notation needed to reason about convergence properties and then providing proofs for the results quoted in the main paper and Appendix B. Appendix E has its own introduction/contents to give an overview of how the theory is built up.

A Code

The code for this paper can be found at `https://github.com/MWPainter/thts-plus-plus/tree/xpr_go`.

B Using average returns with Boltzmann search

In this section we consider how to adapt each of the algorithms considered in this paper to use average returns. We start by describing AR-BTS and AR-DENTS and then subsequently how AR-DENTS can be adapted to give AR-MENTS, AR-RENTS and AR-TENTS. We also give an example of why the adaptations are necessary.

B.1 AR-BTS

We can replace the Bellman values used in BTS with average returns similar to UCT, resulting in the subsequent AR-BTS algorithm. Recall that $\bar{Q}(s,a)$, the average return at (s,a), is defined by:

$$\bar{Q}(s_t,a_t) \leftarrow \bar{Q}(s_t,a_t) + \frac{\bar{R}(s_t,a_t) - \bar{Q}(s_t,a_t)}{N(s_t,a_t)+1}, \tag{28}$$

where $\bar{R}(s_t,a_t) = \sum_{i=t}^{H} R(s_i,a_i)$. An additional issue that needs consideration when using average returns is that the resulting empirical values will not converge to the optimal values when using a stochastic policy, as we argue below. Hence, in AR-BTS we also decay the search temperature, such that the Boltzmann policy tends towards a greedy policy. So, let α be a non-negative function with $\alpha(m) \to 0$ as $m \to \infty$. The search policy used for AR-BTS is then:

$$\pi_{\text{AR-BTS}}(a|s) = (1-\lambda_s)\rho_{\text{AR-BTS}}(a|s) + \frac{\lambda_s}{|\mathcal{A}|}, \tag{29}$$

$$\rho_{\text{AR-BTS}}(a|s) \propto \exp\left(\frac{1}{\alpha(N(s))}\left(\bar{Q}(s,a)\right)\right). \tag{30}$$

The recommendation policy for AR-BTS then uses the $\bar{Q}$ average return values:

$$\psi_{\text{AR-BTS}}(s) = \arg\max_{a\in\mathcal{A}} \bar{Q}(s,a). \tag{31}$$

To see why the decaying search temperature is necessary, consider the MDP in Figure 4, with a temperature of $\alpha(m)=1$. Consider the average return $\bar{V}(2)$ from state 2 that has been visited $N(2)$ times. The probability of selecting action a_1 from state 2 will be $1/(1+e^2)$, and the probability of selecting action a_2 from state 2 will be $e^2/(1+e^2)$. So as $N(2)\to\infty$ we expect the average return to converge to slightly less than two: $\bar{V}(2) \to 2e^2/(1+e^2) < 2$. Taking advantage of this, we can show that for any fixed temperature $\alpha(m) = \alpha_{\text{fix}}$, there is an MDP that AR-BTS would not be consistent. We summarise this result in Proposition B.1. Thus, necessitating the use of the decaying search temperature.

Proposition B.1. *For any $\alpha_{\text{fix}} > 0$, there is an MDP $\mathcal{M}$ such that AR-BTS with $\alpha(m) = \alpha_{\text{fix}}$ is not consistent: $\mathbb{E}[\text{reg}(s_0, \psi_{\text{AR-BTS}})] \not\to 0$ as $n \to \infty$.*

Proof outline. We give a proof outline in Section E.9, using an MDP similar to the one shown in Figure 4. □

Additionally, provided the search temperature decays to zero, AR-BTS will be consistent (Theorem B.2).

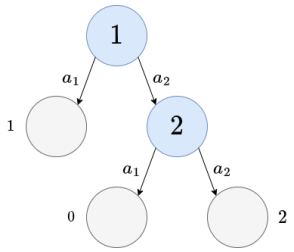


Figure 4: An example MDP to demonstrate the necessity for a decaying search temperature when using average returns.

Theorem B.2. *For any MDP $\mathcal{M}$, if $\alpha(m) \to 0$ as $m \to \infty$ then $\mathbb{E}[\text{reg}(s_0, \psi_{\text{AR-BTS}})] \to 0$ as $n \to \infty$, where n is the number of trials.*

Proof outline. We give a proof outline in Section E.9. □

B.2 AR-DENTS

Using average returns in DENTS is similar to BTS. To compute the average returns we use Equation (28) again. To compute the entropy values $\mathcal{H}_V$ and $\mathcal{H}_Q$ to use in AR-DENTS, we use the same entropy backups as in DENTS:

$$\mathcal{H}_V(s_t) \leftarrow \mathcal{H}(\pi_{\text{AR-DENTS}}(\cdot|s_t)) + \sum_{a \in \mathcal{A}} \pi_{\text{AR-DENTS}}(a_t|s_t)\mathcal{H}_Q(s_t, a_t), \tag{32}$$

$$\mathcal{H}_Q(s_t, a_t) \leftarrow \sum_{s' \in \text{Succ}(s_t, a_t)} \frac{N(s')}{N(s_t, a_t)} \mathcal{H}_V(s'). \tag{33}$$

The search policy in AR-DENTS is the same as in DENTS, but with the Bellman value $\hat{Q}$ replaced by the average return value $\bar{Q}$:

$$\pi_{\text{AR-DENTS}}(a|s) = (1 - \lambda_s)\rho_{\text{AR-DENTS}}(a|s) + \frac{\lambda_s}{|\mathcal{A}|}, \tag{34}$$

$$\rho_{\text{AR-DENTS}}(a|s) \propto \exp\left(\frac{1}{\alpha(N(s))}\left(\bar{Q}(s, a) + \beta(N(s))\mathcal{H}_Q(s_t, a_t)\right)\right). \tag{35}$$

The recommendation policy for AR-DENTS also use the $\bar{Q}$ average return values:

$$\psi_{\text{AR-DENTS}}(s) = \arg\max_{a \in \mathcal{A}} \bar{Q}(s, a).$$

Similarly to the AR-BTS result, we show that if $\alpha(m) \to 0$ and $\beta(m) \to 0$, then AR-DENTS is consistent. Again, because we are considering the AR versions of our algorithms from a practical viewpoint rather than a theoretical one, we only show consistency without any specific (simple) regret bounds.

Theorem B.3. *For any MDP $\mathcal{M}$, if $\alpha(m) \to 0$ and $\beta(m) \to 0$ as $m \to \infty$ then $\mathbb{E}[\text{reg}(s_0, \psi_{\text{AR-DENTS}})] \to 0$ as $n \to \infty$, where n is the number of trials.*

Proof outline. We give a proof outline in Section E.9. □

B.3 AR-MENTS, AR-RENTS and AR-TENTS

MENTS uses *soft values*, $\hat{Q}_{\text{sft}}$, which are not obvious how to replace with average returns. So to produce the AR variants of MENTS, RENTS and TENTS we use AR-DENTS as a starting point.

AR-MENTS. For AR-MENTS we use AR-DENTS, but set $\beta(m) = \alpha(m) = \alpha_{\text{fix}}$. This algorithm resembles MENTS, as the weighting used for entropy in soft values is the same as the Boltzmann policy search temperature.

AR-RENTS. To arrive at AR-RENTS, we replace any use of $\hat{Q}_{\text{sft}}(s, a)$ with $\bar{Q}(s, a) + \beta(m)\mathcal{H}_Q(s, a)$. So we use Equations (28), (32) and (33) for backups, but replace the Shannon entropy function $\mathcal{H}$, with a relative entropy function $\mathcal{H}_{\text{relative}}$ in Equation (32). The relative entropy function $\mathcal{H}_{\text{relative}}$ uses the *Kullback-Leibler divergence* between the search policy and the search policy of the parent decision node. The search policy used is the same as in RENTS, with the aforementioned substitution for soft values. See [10] for full details on computing relative entropy and the search policy used in RENTS.

AR-TENTS. Similarly, for AR-TENTS, we replace any use of $\hat{Q}^m_{\text{sft}}(s, a)$ with $\bar{Q}^m(s, a) + \beta(m)\mathcal{H}_Q(s, a)$, and use Equations (28), (32) and (33) for backups. This time, we replace the Shannon entropy function $\mathcal{H}$, with a Tsallis entropy function $\mathcal{H}_{\text{Tsallis}}$ in Equation (32). Again, we use the same search policy used in TENTS, with the substitution for soft values. See [10] for how Tsallis entropy is computed and the corresponding search policy for TENTS.

C Additional algorithm discussion

C.1 The alias method, implementation details and optimisations

The Alias method [35, 34] can be used to sample from fixed categorical distribution in $O(1)$ time. If the distribution consists of m categories, then a preprocessing step of $O(m)$ is needed to construct the table, details on how to construct the table can be found in [34]. We provide a small example of an alias table in Figure 5.

Any MCTS algorithm that samples actions from a stochastic distribution can make use of the Alias method, for example all of the Boltzmann search algorithms considered in this work. For a node with $A = |\mathcal{A}|$ actions/children, an MCTS algorithm that samples actions from a distribution would typically on each trial: construct the distribution in $O(A)$ time, and then sample from the distribution using a single uniform random number in $O(A)$ time. However, we can instead construct a distribution every A times we visit a search node, which gives an amortised complexity of $O(1)$, and then sampling from an alias table only takes $O(1)$ time.

It is worth noting that when we use the Alias method, we are not sampling from the most up to date distributions possible, so we are making a trade off between being able to sample actions more efficiently and using the most up to date distributions. Moreover, when we use this method, we are making use of the stochastic distribution that will still allow a variety of actions to be sampled on different trials with a fixed distribution. In contrast, UCT uses a deterministic distribution (recall Equation (7)) which selects the action with maximum UCB value and different actions are selected on different trials by changing the distribution (i.e. a new action is selected typically when the confidence term becomes large enough, which requires the UCB values to be computed each time).

We now consider the runtime complexity of running n trials for different algorithms. We summarise these complexities as part of Table 2 and we assume that successor states can be sampled in $O(1)$ time, noting that if we are given a tabular successor state distribution then we can use the Alias method again to sample from it in $O(1)$ time.

C.1.1 BTS backup complexity

Firstly, recall the backups used by BTS for a trajectory $\tau = (s_0, a_0, ..., a_{h-1}, s_h)$:

$$\hat{Q}(s_t, a_t) \leftarrow R(s_t, a_t) + \sum_{s' \in \text{Succ}(s_t, a_t)} \left(\frac{N(s')}{N(s_t, a_t)} \hat{V}(s') \right), \tag{36}$$

$$\hat{V}(s_t) \leftarrow \max_{a \in \mathcal{A}} \hat{Q}(s_t, a). \tag{37}$$

In Equation (36) observe that only the $\hat{V}(s_t)$ term in the sum will change, that is, for each $s' \in \{s'' \in \text{Succ}(s_t, a_t) | s'' \neq s_{t+1}\}$ that the value of $\hat{V}(s')$ will be the same. Hence, we can implement the update in $O(1)$ as:

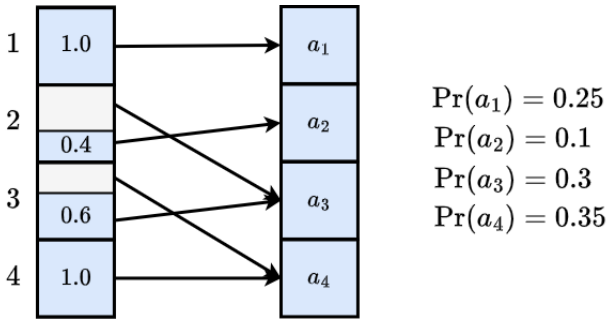


Figure 5: An example of an alias table for a categorical distribution over four actions. To sample from the table we can draw a random index from $I \sim U(\{1, 2, 3, 4\})$, sample a uniformly random number from $x \sim U(0, 1)$ and then follow the pointer depending on if $x >$ thresholds$[I]$ or not. For example, sampling either $(I, x) = (2, 0.5)$ or $(I, x) = (3, 0.1)$ would lead to the action a_3.

$$\hat{Q}(s_t, a_t) \leftarrow \begin{cases} R(s_t, a_t) + \hat{V}(s') & \text{if } N(s_t, a_t) = 1, \\ \hat{Q}(s_t, a_t) - \frac{N(s_{t+1})-1}{N(s_t,a_t)-1}\hat{V}^{\text{old}}(s_{t+1}) + \frac{N(s_{t+1})}{N(s_t,a_t)}\hat{V}(s_{t+1}) & \text{if } N(s_t, a_t) > 1, \end{cases} \tag{38}$$

where $\hat{V}^{\text{old}}(s_{t+1})$ is the previous value of $\hat{V}(s_{t+1})$. To more efficiently implement Equation (37) we can use a *max heap*, which will take $O(\log(A))$ to update the value of $\hat{Q}(s_t, a_t)$ in the heap, and $O(1)$ to read out the maximum value.

Hence, an optimised version of the BTS backups will take $O(\log(A))$. So when running BTS using the Alias method, we need a total of $O(\log(A))$ time to perform the sampling and backups of each node visited, and additionally we will initialise a new Alias table on every trial for the new node added. Hence the runtime complexity of running n trials of BTS with the Alias method will take $O(n(H\log(A) + A))$.

C.1.2 DENTS backup complexity

When using the Alias method with DENTS, we also need to consider the entropy backups too:

$$\mathcal{H}_V(s_t) \leftarrow \mathcal{H}(\pi_{\text{DENTS}}(\cdot|s_t)) + \sum_{a\in\mathcal{A}} \pi_{\text{DENTS}}(a|s_t)\mathcal{H}_Q(s_t, a), \tag{39}$$

$$\mathcal{H}_Q(s_t, a_t) \leftarrow \sum_{s'\in\text{Succ}(s_t,a_t)} \frac{N(s')}{N(s_t, a_t)}\mathcal{H}_V(s'). \tag{40}$$

The backup for $\mathcal{H}_Q$ can be performed in $O(1)$ time, similarly to Equation (38):

$$\mathcal{H}_Q(s_t, a_t) \leftarrow \begin{cases} \mathcal{H}_V(s_{t+1}) & \text{if } N(s_t, a_t) = 1, \\ \mathcal{H}_Q(s_t, a_t) - \frac{N(s_{t+1})-1}{N(s_t,a_t)-1}\mathcal{H}_V^{\text{old}}(s_{t+1})\hat{V}^+(s_{t+1})\frac{N(s_{t+1})}{N(s_t,a_t)}\mathcal{H}_V(s_{t+1}) & \text{if } N(s_t, a_t) > 1, \end{cases} \tag{41}$$

where $\mathcal{H}_V^{\text{old}}(s_{t+1})$ is the previous value of $\mathcal{H}_V(s_{t+1})$.

The backup for $\mathcal{H}_V$ can also be implemented in amortised $O(1)$ time, by noting that we are only updating π_{DENTS} every A visits. Let $\pi_{\text{DENTS}}^{\text{old}}$ be the policy on the previous trial. Then, we have to perform the full $O(A)$ backup when the policy is updated every A visits, but otherwise we can get away with an $O(1)$ backup:

$$\mathcal{H}_V(s_t) \leftarrow \begin{cases} \mathcal{H}(\pi_{\text{DENTS}}(\cdot|s_t)) + \sum_{a\in\mathcal{A}} \pi_{\text{DENTS}}(a|s_t)\mathcal{H}_Q(s_t, a) & \text{if } \pi_{\text{DENTS}} \neq \pi_{\text{DENTS}}^{\text{old}}, \\ \mathcal{H}_V(s_t) + \pi_{\text{DENTS}}(a|s_t)\left(\mathcal{H}_Q(s_t, a_t) - \mathcal{H}_Q^{\text{old}}(s_t, a_t)\right) & \text{if } \pi_{\text{DENTS}} = \pi_{\text{DENTS}}^{\text{old}}, \end{cases} \tag{42}$$

where $\mathcal{H}_Q^{\text{old}}(s_t, a_t)$ is the previous value of $\mathcal{H}_Q(s_t, a_t)$.

As the entropy backups used in DENTS can be implemented in amortised $O(1)$ time, it follows that the complexity of running n trials of DENTS with the Alias method will also take $O(n(H\log(A) + A))$.

C.1.3 MENTS backup complexity

Recall the backups used for MENTS:

$$\hat{Q}_{\text{sft}}(s_t, a_t) \leftarrow R(s_t, a_t) + \sum_{s'\in\text{Succ}(s,a)} \left(\frac{N(s')}{N(s_t, a_t)}\hat{V}_{\text{sft}}(s')\right), \tag{43}$$

$$\hat{V}_{\text{sft}}(s_t) \leftarrow \alpha \log \sum_{a\in\mathcal{A}} \exp\left(\frac{1}{\alpha}\hat{Q}_{\text{sft}}(s_t, a)\right). \tag{44}$$

The backup for $\hat{Q}_{\text{sft}}(s_t, a_t)$ can be implemented in $O(1)$ similarly to Equation (38). However, considering how to implement Equation (44) is more complex. Firstly, we point out the implementing

the equation exactly as written is infact numerically unstable, and in practise we need to make use of the equation:

$$\alpha \log \sum_{a \in \mathcal{A}} \exp\left(\frac{1}{\alpha}\hat{Q}_{\text{sft}}(s_t, a)\right) = \alpha \log \sum_{a \in \mathcal{A}} \exp\left(\frac{1}{\alpha}\left(\hat{Q}_{\text{sft}}(s_t, a) - C\right)\right) + C, \tag{45}$$

where C is an arbitrary constant. The value of C needs to be chosen carefully to avoid numerical underflow or overflow, and is typically set to $C = \max_a \hat{Q}_{\text{sft}}(s_t, a)$. To efficiently compute this backup, we can keep two auxilary variables:

$$M(s_t) = \max_a \hat{Q}_{\text{sft}}(s_t, a) \tag{46}$$

$$E(s_t) = \sum_{a \in \mathcal{A}} \exp\left(\frac{1}{\alpha}\left(\hat{Q}_{\text{sft}}(s_t, a) - M(s_t)\right)\right). \tag{47}$$

Now we can perform the backup for $\hat{Q}_{\text{sft}}(s_t, a_t)$ as follows:

$$M(s_t) \leftarrow \max_a \hat{Q}_{\text{sft}}(s_t, a) \tag{48}$$

$$E(s_t) \leftarrow \begin{cases} \exp(0) & \text{if} N(s_t) = 1 \\ \left(E(s_t) - \exp\left(\frac{1}{\alpha}\left(\hat{Q}^{\text{old}}_{\text{sft}}(s_t, a_t) - M^{\text{old}}(s_t)\right)\right)\right)\exp\left(\frac{1}{\alpha}(M^{\text{old}}(s_t) - M(s_t))\right) & \\ \quad + \exp\left(\frac{1}{\alpha}\left(\hat{Q}_{\text{sft}}(s_t, a_t) - M(s_t)\right)\right) & \text{if} N(s_t) > 1. \end{cases} \tag{49}$$

$$\hat{V}_{\text{sft}}(s_t) \leftarrow \alpha \log(E(s_t)) + M(s_t), \tag{50}$$

where $M^{\text{old}}(s_t), \hat{Q}^{\text{old}}_{\text{sft}}(s_t, a_t)$ are the previous values of $M(s_t), \hat{Q}_{\text{sft}}(s_t, a_t)$ respectively. Similarly to the BTS backups, the requirement of computing a $\max$ operation means that these backups can be computed in $O(\log(A))$ time, so the complexity of running n trials of MENTS with the Alias method takes $O(n(H \log(A) + A))$ time.

It may be possible to implement MENTS faster with a runtime of $O(n(H + A))$ if we could set $M(s_t)$ to some constant value, however this will likely depend on the MDP and the scale of the rewards. Additionally, we found even running this version of MENTS to be quite unstable, and had to resort to using $O(A)$ backups for MENTS in our experiments.

C.1.4 RENTS and TENTS

Both RENTS and TENTS can utilise the Alias method too. We did not consider how to optimise the backups for these algorithms.

C.1.5 Average return complexities

For the AR versions of the algorithms, we note that the backup from Equation (28) can be implemented in $O(1)$ time. Following similar reasoning to the previous algorithms, this means that running n trials with the Alias method for one of the average return algorithms will take $O(n(H + A))$ time.

C.1.6 Decaying the search temperature

Although theoretically using a fixed search temperature α is sufficient for BTS and DENTS to converge, in practise algorithms may perform better with a decaying search temperature $\alpha(m)$ as described in AR-BTS and AR-DENTS. Note that the proofs of convergence for AR-BTS and AR-DENTS also hold for BTS with a decaying temperature. A decaying search temperature would allow BTS and DENTS to focus on deeper parts of the search tree over time.

C.2 Adapting Boltzmann search algorithms for two-player games and neural nets

Now we will detail the adaptions required for each of BTS, DENTS, and MENTS to be used for games. For completeness, we reiterate the adaptions for BTS.

To run our algorithms on games with two players we need to account for an *opponent* that is trying to minimise the value of the game. The agent playing as black (or the *player*) in Go will act on odd timesteps, and the opponent playing as white will act on even timesteps. Fairly informally, this means that the maximum entropy objective needs to be updated:

$$V^{\pi}_{\text{sft}}(s,t) = \mathbb{E}_{\pi}\left[\sum_{i=t}^{H} R\left(s_i,a_i\right) + (-1)^{t-1}\alpha\mathcal{H}\left(\pi\left(\cdot|s_i\right)\right)\Big|s_t = s\right]. \tag{51}$$

This means that each agent (the player and the opponent) will be simultaneously trying to maximise their own entropy, whilst minimising the others.

BTS Values can be initialised with the neural networks as $Q^{\text{init}}(s,a) = \log\tilde{\pi}(a|s) + B$ and $V^{\text{init}}(s) = \tilde{V}(s)$, where B is a constant (adapted from Xiao et al. [37]). With such an initialisation, the initial BTS policy is $\rho_{\text{BTS}}(a|s) \propto \tilde{\pi}(a|s)^{1/\alpha}$.

The stochastic search policy naturally lends itself to mixing in a prior policy, so we can replace BTS search policy π^{n}_{BTS} (Equation (15)) with $\pi_{\text{BTS,mix}}$:

$$\pi_{\text{BTS,mix}}(a|s) = \lambda_{\tilde{\pi}}\tilde{\pi}(a|s) + (1-\lambda_{\tilde{\pi}})\pi_{\text{BTS}}(a|s) \tag{52}$$

$$= \lambda_{\tilde{\pi}}\tilde{\pi}(a|s) + (1-\lambda_{\tilde{\pi}})(1-\lambda_s)\rho_{\text{BTS}}(a|s) + \frac{(1-\lambda_{\tilde{\pi}})\lambda_s}{|\mathcal{A}|}, \tag{53}$$

where $\lambda_{\tilde{\pi}} = \min(1, \epsilon_{\tilde{\pi}}/\log(e+N(s)))$, and $\epsilon_{\tilde{\pi}} \in (0,\infty)$ controls the weighting for the prior policy. To run BTS on a two-player game we need to account the opponent trying to minimise the value of the game. In BTS we can negate the values used in the search policy, and replace the max operation with a min in Bellman backups. That is, we replace equations (16) and (18) with:

$$\rho_{\text{BTS}}(a|s) \propto \exp\left(\frac{-1}{\alpha}\hat{Q}(s,a)\right), \tag{54}$$

$$\hat{V}(s_t) = \min_{a\in\mathcal{A}}\hat{Q}(s_t,a). \tag{55}$$

Finally, when making a recommendation as the opponent, we use the replace the recommendation policy with:

$$\psi_{\text{BTS}}(s) = \arg\min_{a\in\mathcal{A}}\hat{Q}(s,a). \tag{56}$$

MENTS For MENTS we can use the same initialisations as in BTS, that is $Q^{\text{init}}_{\text{sft}}(s,a) = \log\tilde{\pi}(a|s) + B$ and $V^{\text{init}}(s) = \tilde{V}(s)$, with the same value for B. The prior policy $\tilde{\pi}$ is mixed into the MENTS search policy in the same way as for BTS (Equation 53).

At opponent nodes, we can replace any use of the temperature α by $-\alpha$, which effectively turns the softmax into a softmin and gives the highest density to the lowest values in the search policy. So at opponent nodes, we replace Equations (10) and (12) by:

$$\rho_{\text{sft}}(a|s) = \exp\left(\frac{-1}{\alpha}\left(\hat{Q}_{\text{sft}}(s,a) - \hat{V}_{\text{sft}}(s)\right)\right), \tag{57}$$

$$\hat{V}_{\text{sft}}(s_t) = -\alpha\log\sum_{a\in\mathcal{A}}\exp\left(\frac{-1}{\alpha}\hat{Q}_{\text{sft}}(s_t,a)\right). \tag{58}$$

Finally, to make a recommendation as the opponent, we replace the recommendation policy with:

$$\psi_{\text{MENTS}}(s) = \arg\min_{a\in\mathcal{A}}\hat{Q}_{\text{sft}}(s,a). \tag{59}$$

DENTS For DENTS we again use the same initialisations as BTS, setting $Q^{\text{init}}(s,a) = \log\tilde{\pi}(a|s) + B$ and $V^{\text{init}}(s) = \tilde{V}(s)$, using the same value for B. And again, the prior policy $\tilde{\pi}$ is mixed into the DENTS search policy in the same way as for BTS (Equation 53).

	UCT	MENTS	BTS	DENTS
Is consistent for any setting of hyperparameters (Simple regret tends to 0 as $n \to \infty$)	✓	X	✓	✓
Utilises entropy for exploration (E.g. Helpful for sparser rewards)	X	✓	X	✓
Samples actions stochastically (from a Boltzmann distribution)	X	✓	✓	✓
Optimises for cumulative regret (penalises suboptimal actions during planning)	✓	X	X	X
Optimises for simple regret (does not penalise exploration during planning)	X	X	✓	✓
Complexity to run n trials	$O(nHA)$			
Complexity to run n trials using the Alias method (with backups implemented as efficiently as possible)	N/A	$O(n(H\log(A)+A))$		

Table 2: Summary of complexities for different algorithms considered in this work, considering average return variants and alias table optimisations.

At opponent nodes, we update the search policy and backups by replacing Equations (21) and (22) by the following:

$$\rho_{\text{DENTS}}(a|s) \propto \exp\left(\frac{-1}{\alpha}\left(\hat{Q}(s,a) + \beta(N(s))\mathcal{H}_Q(s_t,a_t)\right)\right), \tag{60}$$

$$\mathcal{H}_V(s_t) = -\mathcal{H}(\pi_{\text{DENTS}}(\cdot|s_t)) + \sum_{a\in\mathcal{A}} \pi_{\text{DENTS}}(a_t|s_t)\mathcal{H}_Q(s_t,a_t), \tag{61}$$

and we use replacement for Bellman backups as BTS does in Equation (55). Finally, the recommendation policy for an opponent is:

$$\psi_{\text{DENTS}}(s) = \arg\min_{a\in\mathcal{A}} \hat{Q}(s,a). \tag{62}$$

C.3 Comparison of MCTS algorithms

In Table 2 we summarise the properties of UCT, MENTS, BTS and DENTS so that they can be easily compared.

We also summarise here the benefits that using a stochastic (Boltzmann) policy for action selection can provide:

- Using a stochastic policy allows the Alias method (as described in Section C.1) to be used, which can significantly increase the number of trials that can be run in a fixed time period;
- Using a stochastic policy naturally encourages exploration as it will still always have some probability of sampling each action;
- And the entropy can be computed of a stochastic distribution, which can then be used as an exploration bonus.

The benefits of additional exploration from entropy and the stochastic distribution include: helping to find delayed/sparse rewards in the environment; confirming that bad actions are in fact bad; and, greater exploration leads to less contention between threads in a multi-threaded implementation (discussed in Section C.3.2).

C.3.1 When to use each MCTS algorithm

Although the best way to know for sure which MCTS algorithm will be the best for a given problem is to directly try it and compare performance, there are some properties of problems that give a good indication of which algorithm would be a good fit.

Two properties of environments that would make them more amenable to using entropy for exploration are large delayed rewards, and, containing trap states (i.e. states where the agent cannot move to

another state for the rest of the trial). The Frozen Lake environment (Section 5.1.1) demonstrates both of these properties, there is a large reward that the agent only gets when it reaches the goal, and there are holes that the agent falls into. Hence, if a problem contains large delayed rewards or trap states, then DENTS would likely be the best fit.

When the reward signal is dense or engineered so that optimal policy can be found without needing additional exploration, then using entropy for additional exploration may cause unnecessary additional exploration, in which case using UCT or BTS would be more suitable.

Finally, we note that there may be more properties of problems that we have not considered in this work that make them ameniable to different algorithms. For example, as mentioned in Section 3 that sometimes it may be desirable for an agent to follow a maximum entropy policy to explore and learn in an unknown environment, in which case MENTS, RENTS or TENTS may be more preferrable.

C.3.2 Discussion on multi-threading in MCTS

In this section we give a brief discussion about why Boltzmann search policies naturally utilise multi-threading more than UCT. Consider a two-armed bandit, with actions a_1 and a_2, and corresponding rewards of 0 and 1 respectively. Intuitively, an emphasis on exploration will lead to less contention between threads as they will explore different parts of the search tree.

When running UCB on this multi-armed bandit, with a bias of 2.0, it will pull a_1 a total of 20 times in the first 1000 pulls, and then it will only pull a_1 a total of 9 times in the next 4000 pulls, as it begins to exploit. If we had multiple threads trying to pull arms simultaneously, but they could not pull arm a_2 simultaneously, then it would lead to lots of contention between the threads.

In stark contrast, when using a Boltzmann policy, with a temperature of 1.0 on this multi-armed bandit, the expected number of pulls for a_1 will be 268 pull for every 1000 total pulls. In a multi-threaded environment, having around a quarter of the threads pulling arm a_1 rather than all threads pulling arm a_2 naturally leads to less contention and waiting.

Although this is an unrealistic example, it highlights the problem at hand: the exploitation of UCB leads to contention between threads in a multi-threaded setting. Moreover, this can be witnessed in Section D.5, Table **??** where UCT is able to run fewer trials in the earlier moves of a Go game.

Aditionally, it is worth observing that both UCT and BTS have parameters that control the amount of exploration (the bias c and search temperature α for UCT and BTS respectively). However, because UCT is designed with cumulative regret in mind, it will always towards picking the same action on successive trials, as can be seen in our simple example.

D Additional results and discussion

D.1 Gridworld environment details

For space, the specific maps for the gridworlds are omitted from the results section in the main paper. In the gridworld maps, `S` denotes the starting location of the agent, `F` denotes spaces the agent can move to, `H` denote holes that end the agents trial and `G` is the goal location.

In Figure 6a we give an 8x8 Frozen Lake environment that is used in Section D.2 to demonstrate how the different algorithms perform with a variety of temperatures. Figure 6b gives the 8x12 Frozen Lake Environment that is used for hyperparameter selection in Section D.3. And in Figure 6c we give the 8x12 Frozen Lake Environment that is used to test the algorithms in Section 5. Each of these maps was randomly generated, with each location having a probability of $1/5$ of being a hole, and the maps were checked to have a viable path from the starting location to the goal location.

Aditionally, we give the map used in the 6x6 Sailing Problem, and the wind transition probabilities in Figure 7. In the Sailing domain, actions and wind directions can take values in $\{0, 1, ..., 7\}$, with a value of 0 representing North/up, 2 representing East/right, 4 representing South/down and 6 representing West/right. The remaining numbers represent the inter-cardinal directions. In Section D.3 the wind direction was set to North (or 0) in the initial state, and for testing in Section 5, the initial wind direction was set to South-East (or 3).

D.2 Futher discussion on parameter sensitivity

In this section we provide a more detailed discussion on sensitivity to the temperature parameter in MENTS [37], RENTS and TENTS [10]. We provide more thorough results on D-chain environments, with each algorithm, and varying both the temperature parameter and the ϵ exploration parameter. Additionally, we provide results using the 8x8 Frozen Lake environment (Figure 6a) using a variety of temperatures to demonstrate how each algorithm performs with different temperatures in that domain.

D.2.1 Parameter sensitivity in D-Chain

We run each algorithm with a variety of α temperatures, and the ϵ exploration parameter on the 10-chain environments (Figure 8). Additionally, we ran UCT with a variety of bias parameters. Figures 9, 10, 11, 12, 13 and 14 give results for the 10-chain environment, with algorithms UCT, MENTS, RENTS, TENTS, BTS and DENTS respectively. Figures 15, 16, 17, 18, 19 and 20 give results for the modified 10-chain environment, with algorithms UCT, MENTS, RENTS, TENTS, BTS and DENTS respectively.

As expected with UCT, regardless of how the bias parameter is set, in both the 10-chain ($D = 10$, $R_f = 1.0$) and modified 10-chain ($D = 10$, $R_f = 0.5$) environments, it only achieves a value of 0.9. See Figures 9 and 15 for plots.

As discussed in Section 3, for higher temperatures in MENTS it will find the reward of R_f in both the 10-chain and modified 10-chain environments. At a temperature of $\alpha = 0.15$ MENTS is able to find the reward of $R_f = 1$ on the 10-chain (Figure 10), but will still recommend a policy that gives the

```
SFFFFFHF
FFFFFFFF
FHFHFFFF
FFFFFFHH
FFFHFFFF
FHHHFFFF
FFFFFHFF
FFFFFFFG
```

(a) 8x8 Frozen Lake.

```
SFHFFFHFFFFF
FFFFFFFHFFFF
HFFFFFHFFFFF
FHFFHFFFFFFF
HHFFFFFFFFFF
FHFFFFHFFFFF
FHFFFHHFHFFF
FFFFFFFFFHHG
```

(b) 8x12 Frozen Lake.

```
SFHFFFFFFFHF
FFFFFFFFFFFF
FHFFFFHFFFFF
FFFHFFFFFFHF
FFFFFFFFFFFF
FFFFHFFFHFFF
FFHFFFFFFFFH
FFFFFFFFFFFG
```

(c) 8x12 Test Frozen Lake.

Figure 6: Maps used for experiments using the Frozen Lake environment in Sections 5, D.2 and D.3. `S` is the starting location for the agent, `F` represents floor that the agent can move too, `H` are holes that end the agents trial and `G` is the goal location.

```
FFFFFG
FFFFFF
FFFFFF
FFFFFF
FFFFFF
SFFFFF
```

(a) 6x6 Sailing Problem map.

$$\begin{pmatrix} 0.4 & 0.3 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.3 \\ 0.4 & 0.3 & 0.3 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.4 & 0.3 & 0.3 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.4 & 0.3 & 0.3 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.4 & 0.2 & 0.4 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.3 & 0.3 & 0.4 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.3 & 0.3 & 0.4 \\ 0.4 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.3 & 0.3 \end{pmatrix}$$

(b) Wind transition probabilities.

Figure 7: The map used for the 6x6 Sailing Problem and the wind transition probabilities. For the wind transition probabilities, the $(i, j)th$ element of the matrix denotes the probability that the wind changes from direction i to direction j, where 0 denotes North/up, 1 denotes North-East/up-right, and so on.

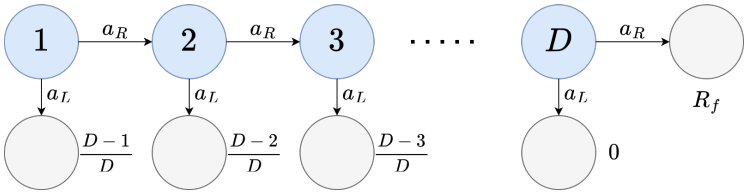


Figure 8: An illustration of the *(modified) D-chain problem*, where 1 is the starting state, transitions are deterministic and values next to states represent rewards for arriving in that state.

reward of $R_f = 0.5$ on the modified 10-chain (Figure 16). At a temperature of $\alpha = 0.1$ MENTS will struggle to find the reward of $R_f = 1$ in the 10-chain, without the help of the exploration parameter, but this is the first temperature we tried that was able to recommend the optimal policy in the modified 10-chain (Figure 16). For low temperatures, such as $\alpha = 0.01$, MENTS was able to find the optimal policy, but in the case of the 10-chain with $R_f = 1$ it can only do so with the help of a higher exploration parameter.

When we ran TENTS on the (modified) 10-chain, we see results that parallel MENTS, see Figures 12 and 18. Interestingly, RENTS was only able to find the reward of $R_f = 1$ on the 10-chain environment if we used a low temperature, $\alpha = 0.01$ and a high exploration parameter, $\epsilon = 10$. Otherwise, RENTS tended to behave similarly to UCT on these environments, see Figures 11 and 17.

In contrast, BTS was able to find the reward of $R_f = 1.0$ in the 10-chain when a high search temperature or high exploration parameter was used (Figure 13). And, in the modified 10-chain, BTS always achieves a reward of 0.9 regardless of how the parameters are set (Figure 19). DENTS performance on the 10-chain (Figure 14) and modified 10-chain (Figure 20) was similar to BTS, but tended to find the reward of $R_f = 1$ in the 10-chain marginally faster. For the decay function β in DENTS, we always set $\beta(m) = \alpha / \log(e + m)$ for these experiments.

To demonstrate that the ϵ exploration parameter is insufficient to make up for a low temperature, we also consider the 20-chain ($D = 20$, $R_f = 1$) and modified 20-chain ($D = 20$, $R_f = 0.5$) problems. We don't give plots for all algorithms on both of the 20-chain environments like we do for 10-chain environments, but opt for the plots that demonstrate something interesting.

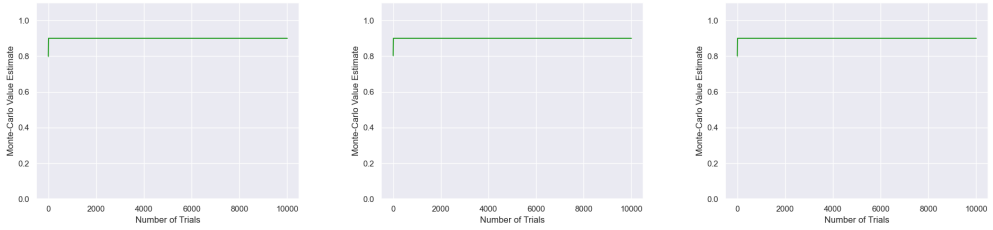
In Figure 21 we see MENTS on the 20-chain is able to find the reward of $R_f = 1$ for higher temperatures. However, this time, the exploration parameter does not make much of an impact when using lower temperatures. Moreover, a large exploration parameter appears to negatively impact MENTS ability to find $R_f = 1$. This makes sense considering that a uniformly random policy will find the reward at the end of the chain once every 2^{10} trials in the 10-chain, but only once every 2^{20} in the 20-chain. Again, on the modified 20-chain, MENTS is only able to recommend the optimal policy for low temperatures (see Figure 22).

When we ran BTS on the 20-chain, it was unsuccessful at finding the final reward of $R_f = 1$, which makes sense as it is not using entropy for exploration, and it is unlikely to follow a random policy to the

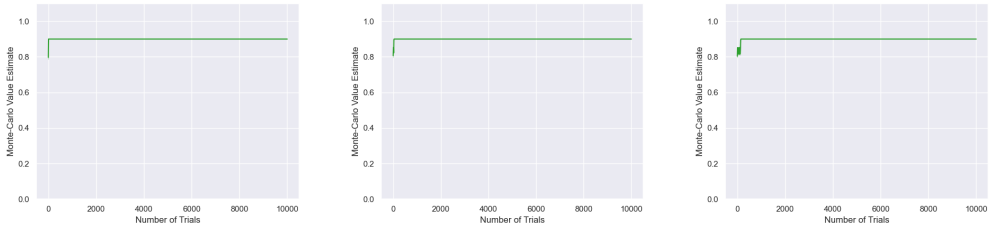
end of the chain (Figure 23). For DENTS, we again used a decay function of $\beta(m) = \alpha/\log(e+m)$ for simplicity, and unfortunately it was only able to make slow progress towards finding the final reward of $R_f = 1$ for high temperatures. However, if we independently set the values of α and

However, DENTS on the 20-chain begins to make slow progress towards finding the final reward of $R_f = 1$, but requires a higher temperature to be used, as we decay the weighting of entropy over time (Figure 24). Again we used a decay function of $\beta(m) = \alpha/\log(e+m)$ here for simplicity, and if we properly select them DENTS is more than capable of solving the 20-chain. For example we show that using DENTS with $\alpha = 0.5$, $\beta(m) = 10/\log(e+m)$ and $\epsilon = 0.01$ in Figure 26, where α is set low enough that there is still a high probability of following the chain to the end, β is set to be large initially to encourage exploring with the entropy reward and ϵ is set low to avoid random exploration ending trials before reaching the end of the chain. If we were to run DENTS and BTS on the modified 20-chain they would recommend the optimal policy giving a value of 0.95 for all of the parameters we searched over (not shown).

Finally, in Figure 25 we also consider running DENTS, but instead setting $\beta(m) = \alpha$ to replicate MENTS. The main difference between DENTS in this case and MENTS is the recommendation policy, where DENTS uses the Bellman values for recommendations, rather than soft values. So even in cases where the MENTS search is more desirable, we can replicate it with DENTS while providing recommendations for the standard objective. Moreover, running DENTS with $\beta(m) = \alpha$ on the modified 20-chain would always yield the optimal value of 0.95 because of the use of Bellman values for recommendations (not shown).

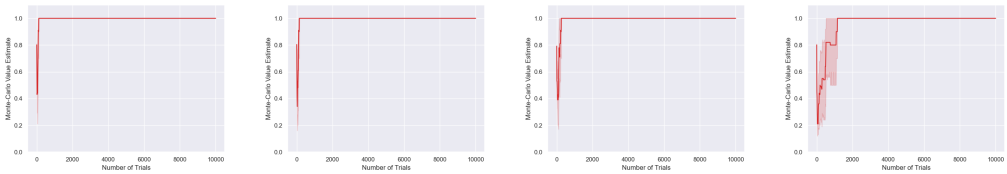


Bias set using [21]. Bias $= 0.1$. Bias $= 1$.

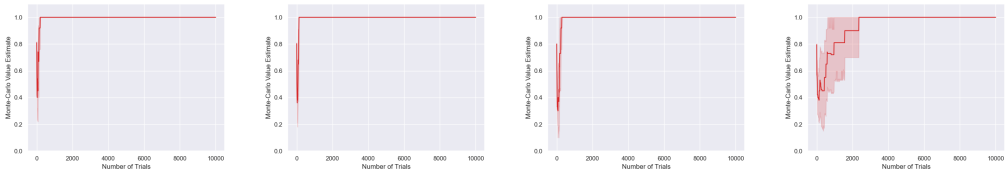


Bias $= 2$. Bias $= 10$. Bias $= 100$.

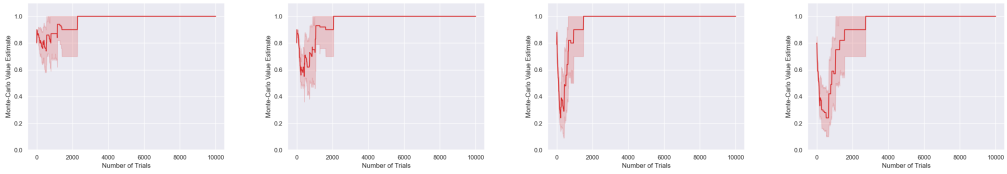
Figure 9: Results for UCT on the 10-chain ($D = 10$, $R_f = 1.0$), for varying bias parameters.



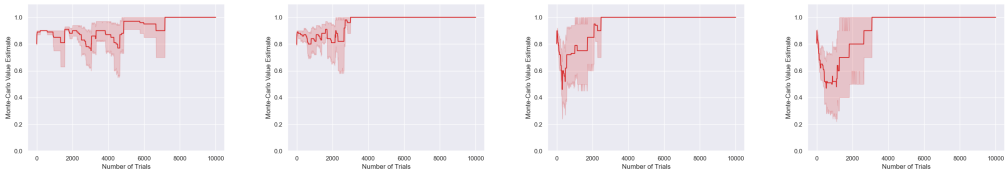
$\alpha = 1, \epsilon = 0.01$ $\alpha = 1, \epsilon = 0.1$ $\alpha = 1, \epsilon = 1$ $\alpha = 1, \epsilon = 10$



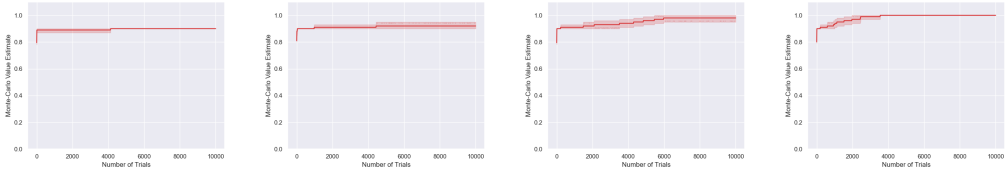
$\alpha = 0.5, \epsilon = 0.01$ $\alpha = 0.5, \epsilon = 0.1$ $\alpha = 0.5, \epsilon = 1$ $\alpha = 0.5, \epsilon = 10$



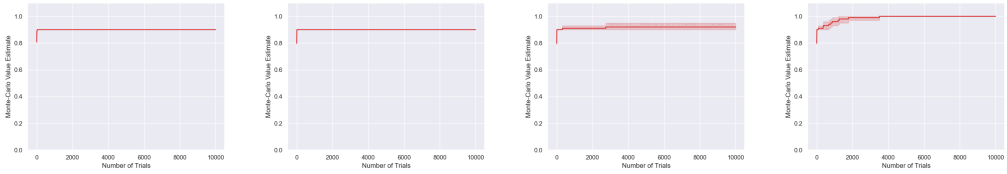
$\alpha = 0.2, \epsilon = 0.01$ $\alpha = 0.2, \epsilon = 0.1$ $\alpha = 0.2, \epsilon = 1$ $\alpha = 0.2, \epsilon = 10$



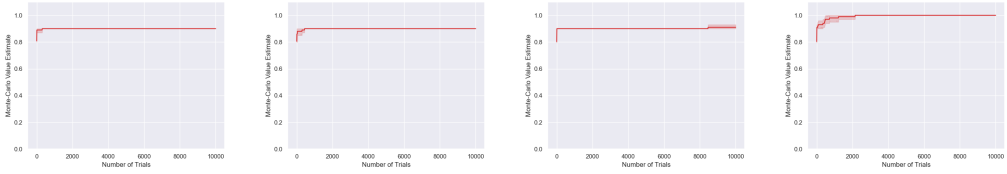
$\alpha = 0.15, \epsilon = 0.01$ $\alpha = 0.15, \epsilon = 0.1$ $\alpha = 0.15, \epsilon = 1$ $\alpha = 0.15, \epsilon = 10$



$\alpha = 0.1, \epsilon = 0.01$ $\alpha = 0.1, \epsilon = 0.1$ $\alpha = 0.1, \epsilon = 1$ $\alpha = 0.1, \epsilon = 10$

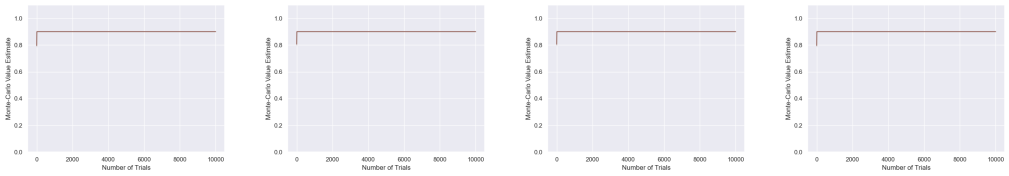


$\alpha = 0.05, \epsilon = 0.01$ $\alpha = 0.05, \epsilon = 0.1$ $\alpha = 0.05, \epsilon = 1$ $\alpha = 0.05, \epsilon = 10$

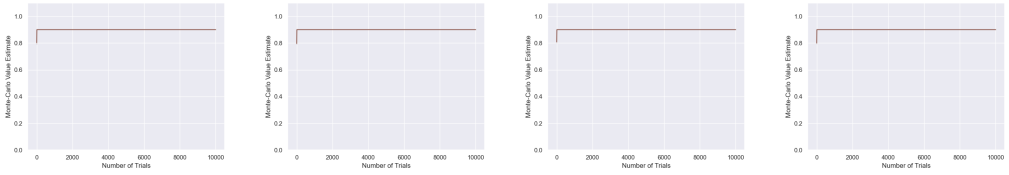


$\alpha = 0.01, \epsilon = 0.01$ $\alpha = 0.01, \epsilon = 0.1$ $\alpha = 0.01, \epsilon = 1$ $\alpha = 0.01, \epsilon = 10$

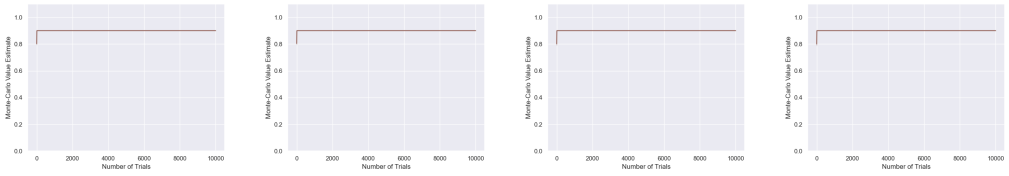
Figure 10: Results for MENTS on the 10-chain ($D = 10$, $R_f = 1.0$), for varying temperatures and exploration parameters.



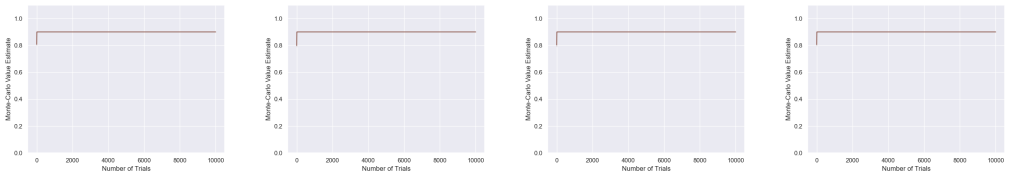
$\alpha = 1000, \epsilon = 0.01$ $\alpha = 1000, \epsilon = 0.1$ $\alpha = 1000, \epsilon = 1$ $\alpha = 1000, \epsilon = 10$



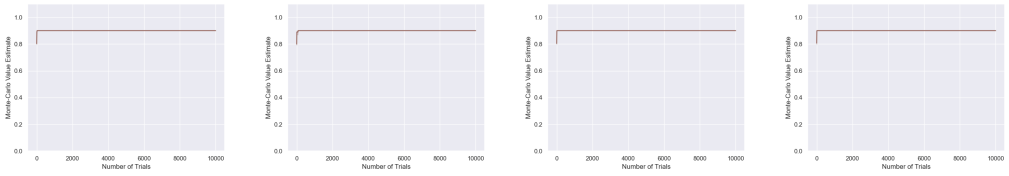
$\alpha = 100, \epsilon = 0.01$ $\alpha = 100, \epsilon = 0.1$ $\alpha = 100, \epsilon = 1$ $\alpha = 100, \epsilon = 10$



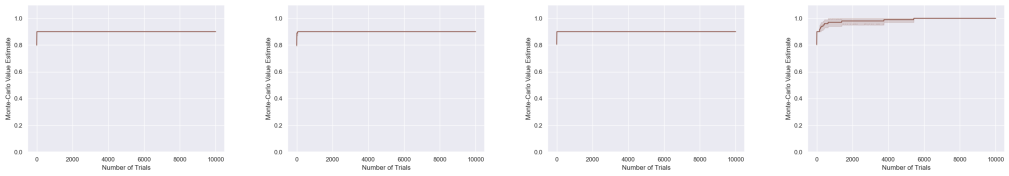
$\alpha = 10, \epsilon = 0.01$ $\alpha = 10, \epsilon = 0.1$ $\alpha = 10, \epsilon = 1$ $\alpha = 10, \epsilon = 10$



$\alpha = 1, \epsilon = 0.01$ $\alpha = 1, \epsilon = 0.1$ $\alpha = 1, \epsilon = 1$ $\alpha = 1, \epsilon = 10$

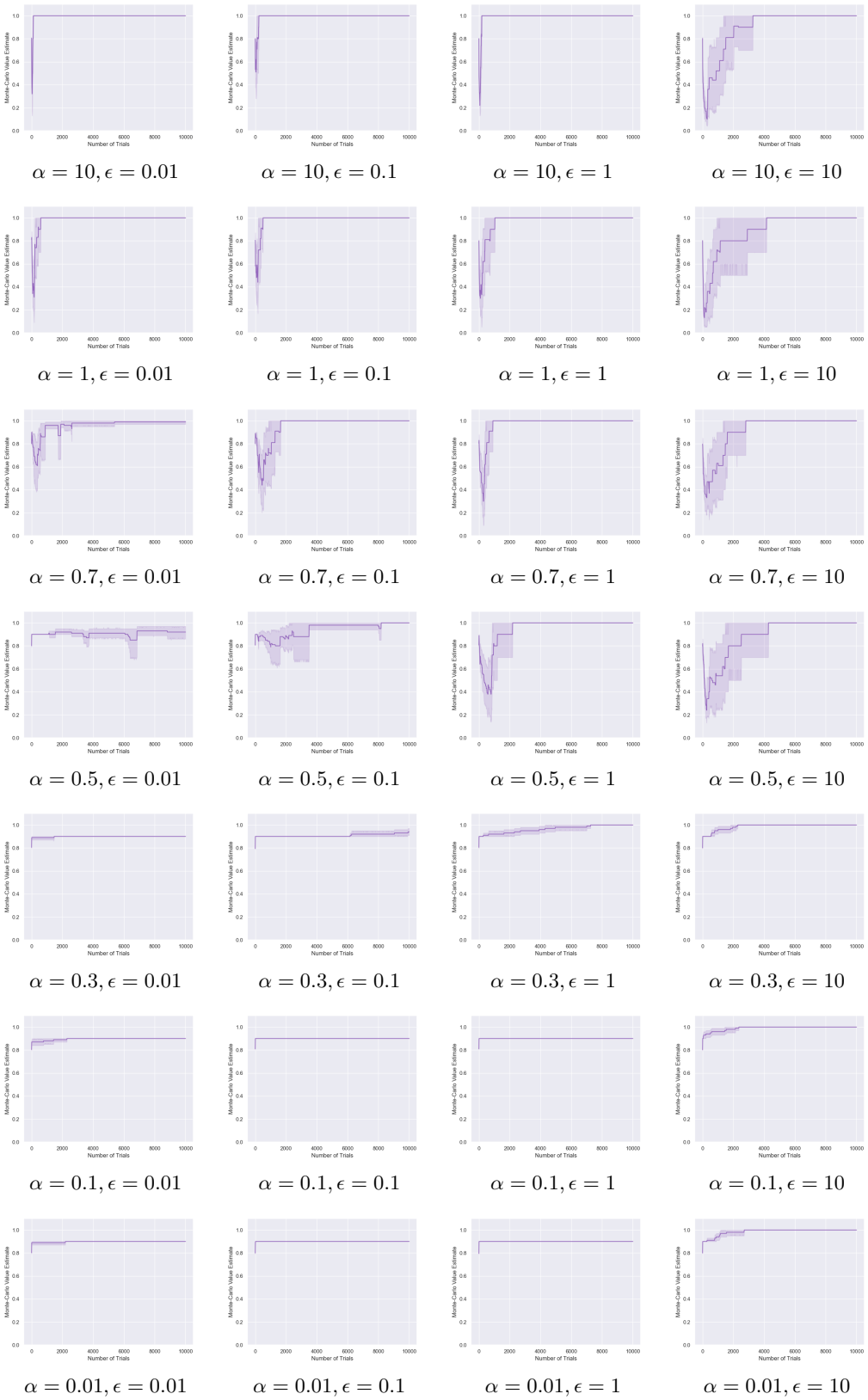


$\alpha = 0.1, \epsilon = 0.01$ $\alpha = 0.1, \epsilon = 0.1$ $\alpha = 0.1, \epsilon = 1$ $\alpha = 0.01, \epsilon = 10$



$\alpha = 0.01, \epsilon = 0.01$ $\alpha = 0.01, \epsilon = 0.1$ $\alpha = 0.01, \epsilon = 1$ $\alpha = 0.01, \epsilon = 10$

Figure 11: Results for RENTS on the 10-chain ($D = 10$, $R_f = 1.0$), for varying temperatures and exploration parameters.



$\alpha = 10, \epsilon = 0.01$ $\alpha = 10, \epsilon = 0.1$ $\alpha = 10, \epsilon = 1$ $\alpha = 10, \epsilon = 10$

$\alpha = 1, \epsilon = 0.01$ $\alpha = 1, \epsilon = 0.1$ $\alpha = 1, \epsilon = 1$ $\alpha = 1, \epsilon = 10$

$\alpha = 0.7, \epsilon = 0.01$ $\alpha = 0.7, \epsilon = 0.1$ $\alpha = 0.7, \epsilon = 1$ $\alpha = 0.7, \epsilon = 10$

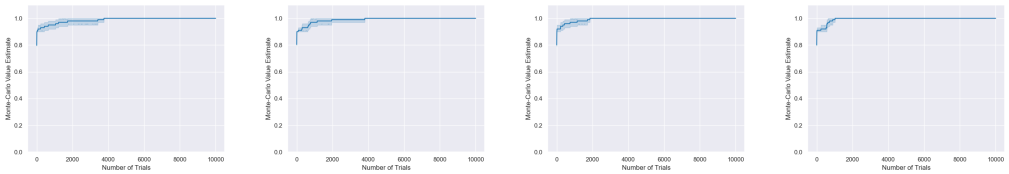
$\alpha = 0.5, \epsilon = 0.01$ $\alpha = 0.5, \epsilon = 0.1$ $\alpha = 0.5, \epsilon = 1$ $\alpha = 0.5, \epsilon = 10$

$\alpha = 0.3, \epsilon = 0.01$ $\alpha = 0.3, \epsilon = 0.1$ $\alpha = 0.3, \epsilon = 1$ $\alpha = 0.3, \epsilon = 10$

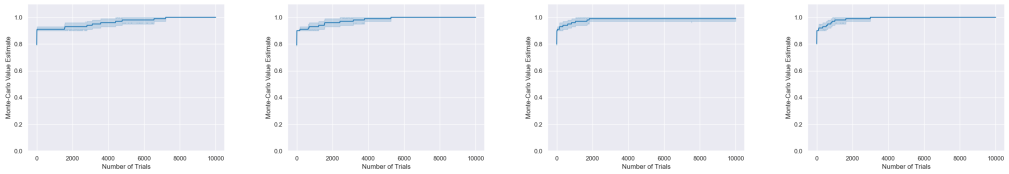
$\alpha = 0.1, \epsilon = 0.01$ $\alpha = 0.1, \epsilon = 0.1$ $\alpha = 0.1, \epsilon = 1$ $\alpha = 0.1, \epsilon = 10$

$\alpha = 0.01, \epsilon = 0.01$ $\alpha = 0.01, \epsilon = 0.1$ $\alpha = 0.01, \epsilon = 1$ $\alpha = 0.01, \epsilon = 10$

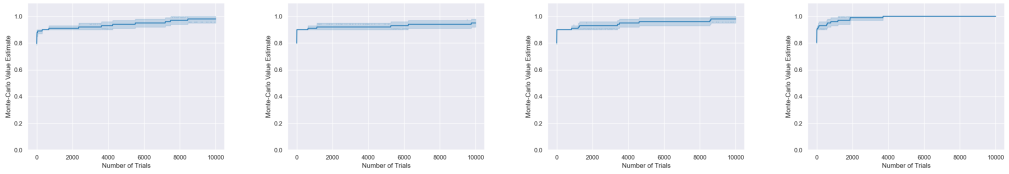
Figure 12: Results for TENTS on the 10-chain ($D = 10$, $R_f = 1.0$), for varying temperatures and exploration parameters.



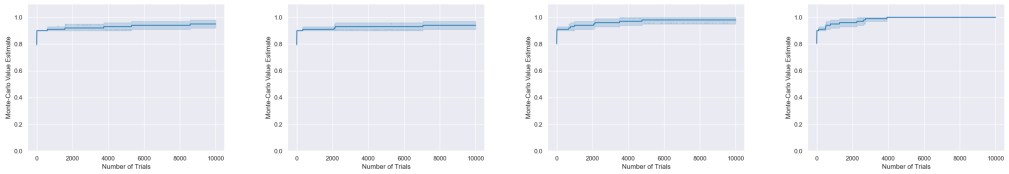
$\alpha = 1, \epsilon = 0.01$ $\alpha = 1, \epsilon = 0.1$ $\alpha = 1, \epsilon = 1$ $\alpha = 1, \epsilon = 10$



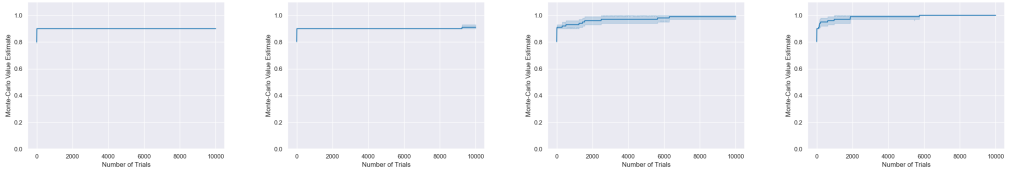
$\alpha = 0.5, \epsilon = 0.01$ $\alpha = 0.5, \epsilon = 0.1$ $\alpha = 0.5, \epsilon = 1$ $\alpha = 0.5, \epsilon = 10$



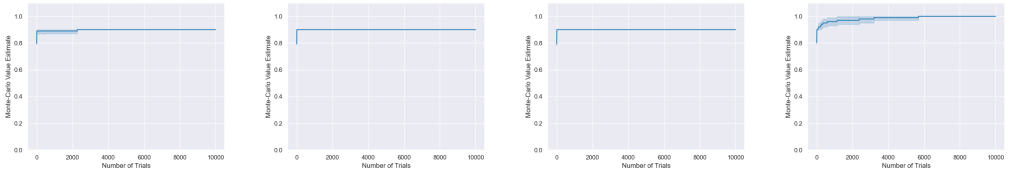
$\alpha = 0.2, \epsilon = 0.01$ $\alpha = 0.2, \epsilon = 0.1$ $\alpha = 0.2, \epsilon = 1$ $\alpha = 0.2, \epsilon = 10$



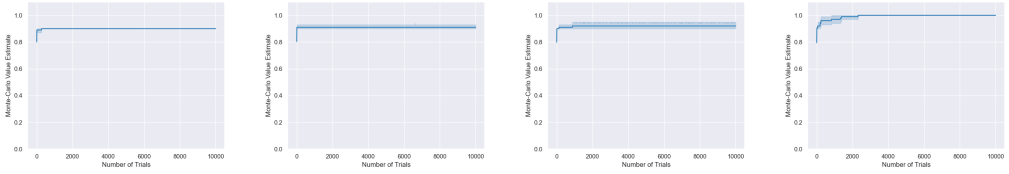
$\alpha = 0.15, \epsilon = 0.01$ $\alpha = 0.15, \epsilon = 0.1$ $\alpha = 0.15, \epsilon = 1$ $\alpha = 0.15, \epsilon = 10$



$\alpha = 0.1, \epsilon = 0.01$ $\alpha = 0.1, \epsilon = 0.1$ $\alpha = 0.1, \epsilon = 1$ $\alpha = 0.1, \epsilon = 10$

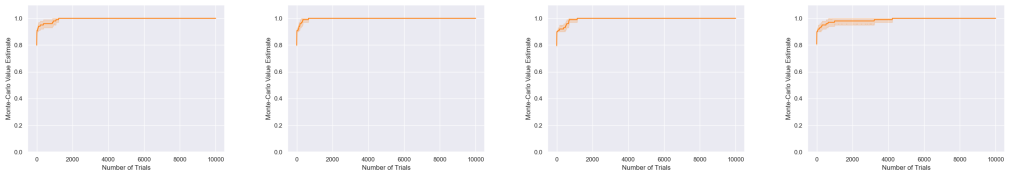


$\alpha = 0.05, \epsilon = 0.01$ $\alpha = 0.05, \epsilon = 0.1$ $\alpha = 0.05, \epsilon = 1$ $\alpha = 0.05, \epsilon = 10$

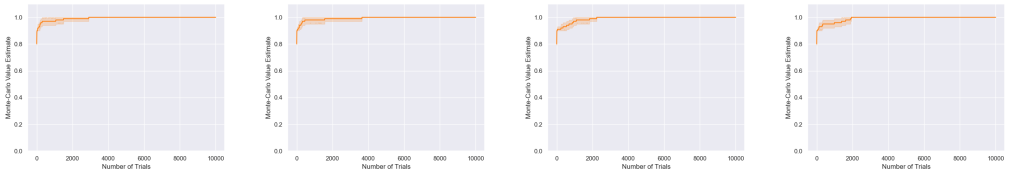


$\alpha = 0.01, \epsilon = 0.01$ $\alpha = 0.01, \epsilon = 0.1$ $\alpha = 0.01, \epsilon = 1$ $\alpha = 0.01, \epsilon = 10$

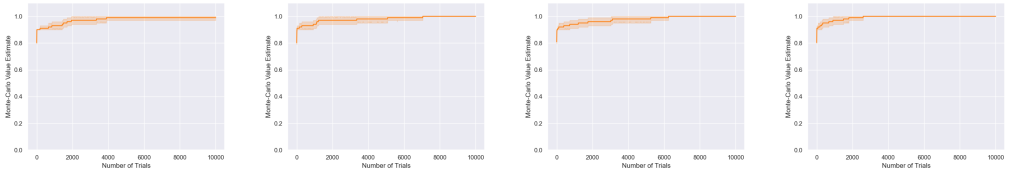
Figure 13: Results for BTS on the 10-chain ($D = 10$, $R_f = 1.0$), for varying temperatures and exploration parameters.



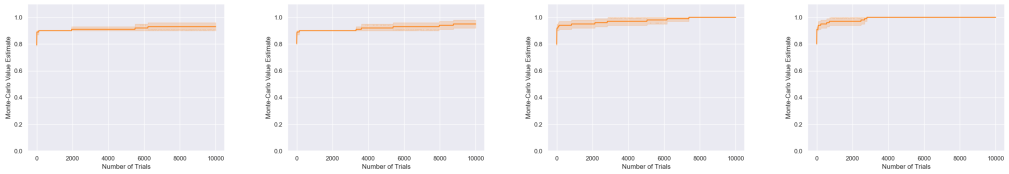
$\alpha = 1, \epsilon = 0.01$ $\alpha = 1, \epsilon = 0.1$ $\alpha = 1, \epsilon = 1$ $\alpha = 1, \epsilon = 10$



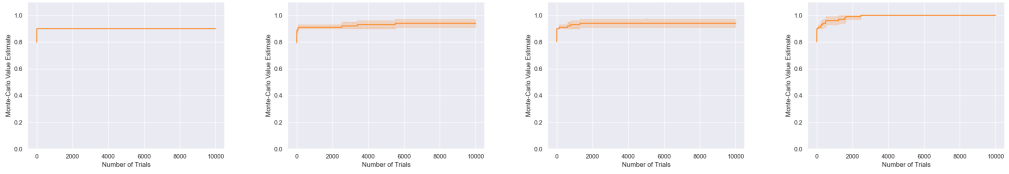
$\alpha = 0.5, \epsilon = 0.01$ $\alpha = 0.5, \epsilon = 0.1$ $\alpha = 0.5, \epsilon = 1$ $\alpha = 0.5, \epsilon = 10$



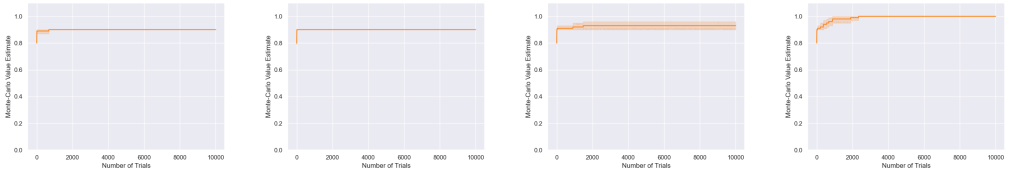
$\alpha = 0.2, \epsilon = 0.01$ $\alpha = 0.2, \epsilon = 0.1$ $\alpha = 0.2, \epsilon = 1$ $\alpha = 0.2, \epsilon = 10$



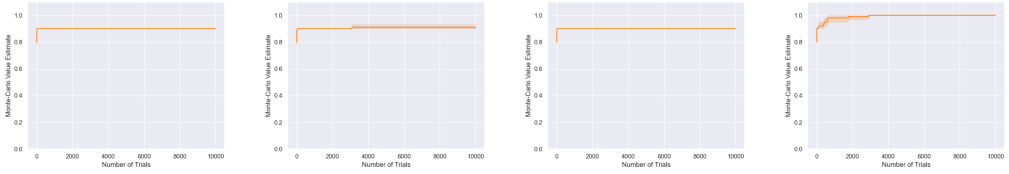
$\alpha = 0.15, \epsilon = 0.01$ $\alpha = 0.15, \epsilon = 0.1$ $\alpha = 0.15, \epsilon = 1$ $\alpha = 0.15, \epsilon = 10$



$\alpha = 0.1, \epsilon = 0.01$ $\alpha = 0.1, \epsilon = 0.1$ $\alpha = 0.1, \epsilon = 1$ $\alpha = 0.1, \epsilon = 10$

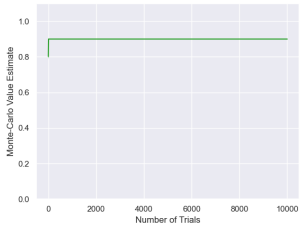


$\alpha = 0.05, \epsilon = 0.01$ $\alpha = 0.05, \epsilon = 0.1$ $\alpha = 0.05, \epsilon = 1$ $\alpha = 0.05, \epsilon = 10$

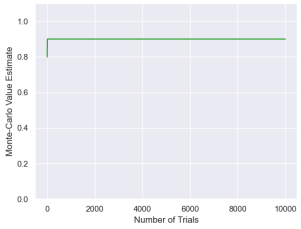


$\alpha = 0.01, \epsilon = 0.01$ $\alpha = 0.01, \epsilon = 0.1$ $\alpha = 0.01, \epsilon = 1$ $\alpha = 0.01, \epsilon = 10$

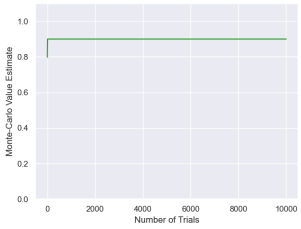
Figure 14: Results for DENTS on the 10-chain ($D = 10$, $R_f = 1.0$), for varying temperatures and exploration parameters. The decay function was set to $\beta(m) = \alpha / \log(e + m)$.



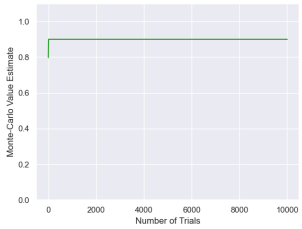
Bias set using [21].



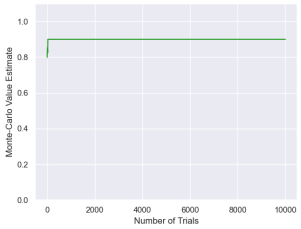
Bias $= 0.1$.



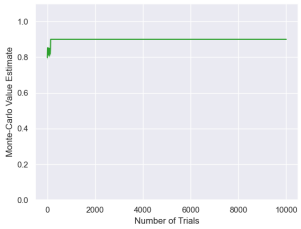
Bias $= 1$.



Bias $= 2$.

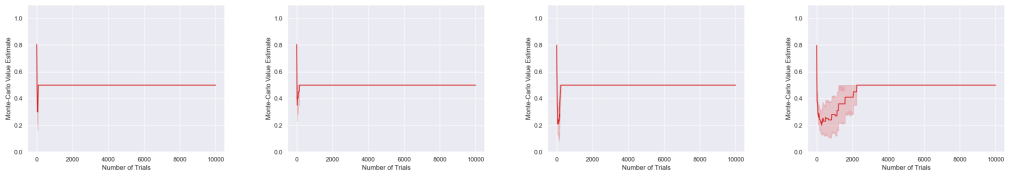


Bias $= 10$.

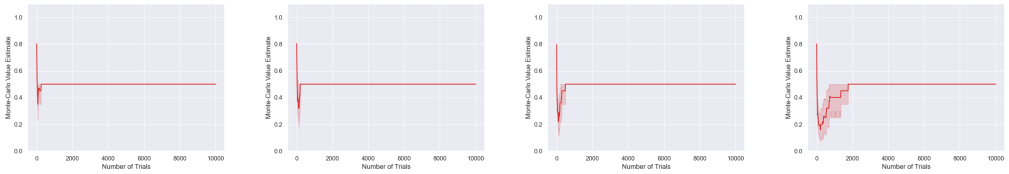


Bias $= 100$.

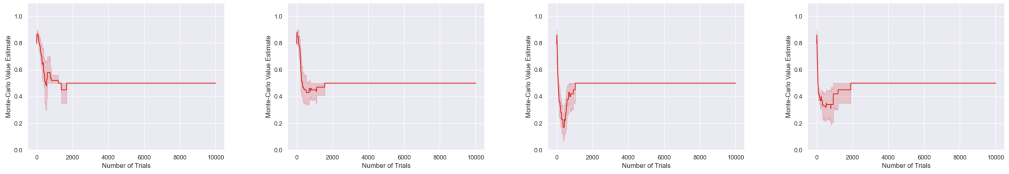
Figure 15: Results for UCT on the modified 10-chain ($D = 10$, $R_f = 0.5$), for varying bias parameters.



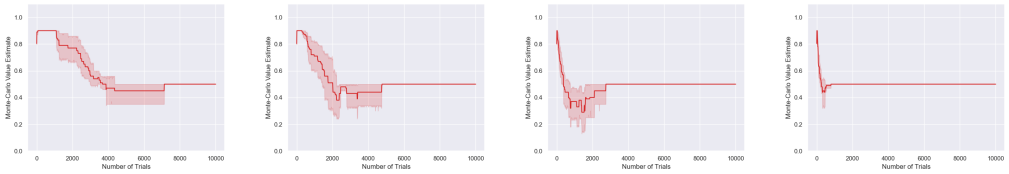
$\alpha = 1, \epsilon = 0.01$ $\alpha = 1, \epsilon = 0.1$ $\alpha = 1, \epsilon = 1$ $\alpha = 1, \epsilon = 10$



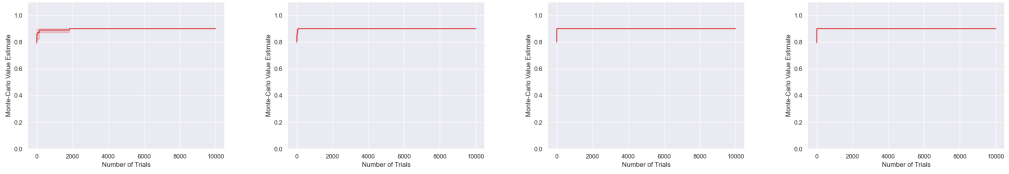
$\alpha = 0.5, \epsilon = 0.01$ $\alpha = 0.5, \epsilon = 0.1$ $\alpha = 0.5, \epsilon = 1$ $\alpha = 0.5, \epsilon = 10$



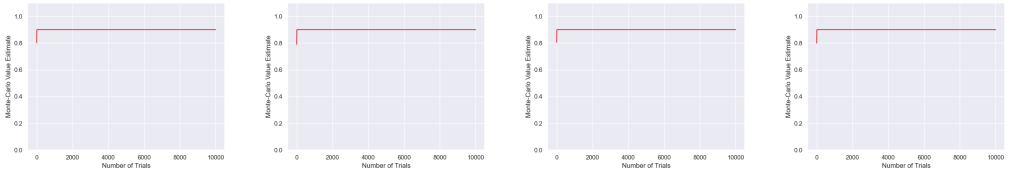
$\alpha = 0.2, \epsilon = 0.01$ $\alpha = 0.2, \epsilon = 0.1$ $\alpha = 0.2, \epsilon = 1$ $\alpha = 0.2, \epsilon = 10$



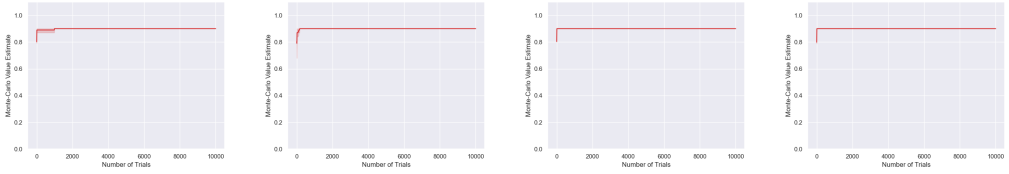
$\alpha = 0.15, \epsilon = 0.01$ $\alpha = 0.15, \epsilon = 0.1$ $\alpha = 0.15, \epsilon = 1$ $\alpha = 0.15, \epsilon = 10$



$\alpha = 0.1, \epsilon = 0.01$ $\alpha = 0.1, \epsilon = 0.1$ $\alpha = 0.1, \epsilon = 1$ $\alpha = 0.1, \epsilon = 10$

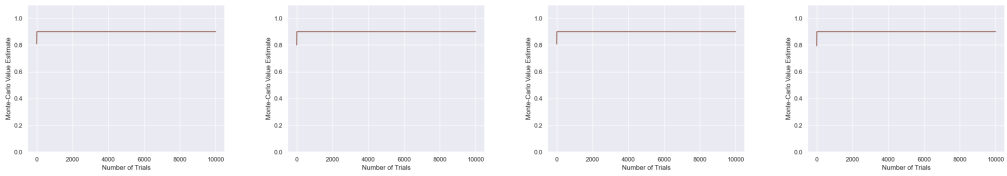


$\alpha = 0.05, \epsilon = 0.01$ $\alpha = 0.05, \epsilon = 0.1$ $\alpha = 0.05, \epsilon = 1$ $\alpha = 0.05, \epsilon = 10$

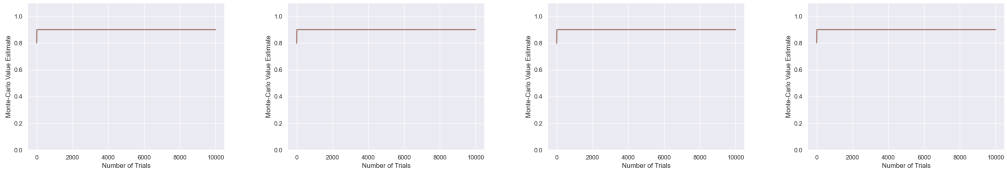


$\alpha = 0.01, \epsilon = 0.01$ $\alpha = 0.01, \epsilon = 0.1$ $\alpha = 0.01, \epsilon = 1$ $\alpha = 0.01, \epsilon = 10$

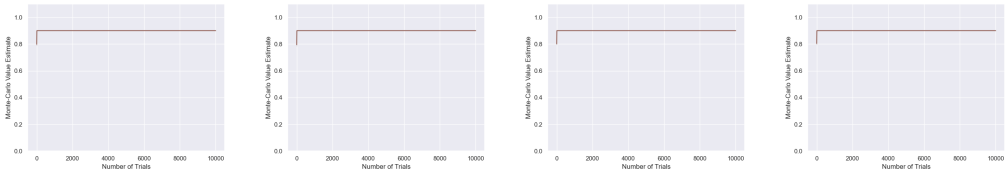
Figure 16: Results for MENTS on the modified 10-chain ($D = 10$, $R_f = 0.5$), for varying temperatures and exploration parameters.



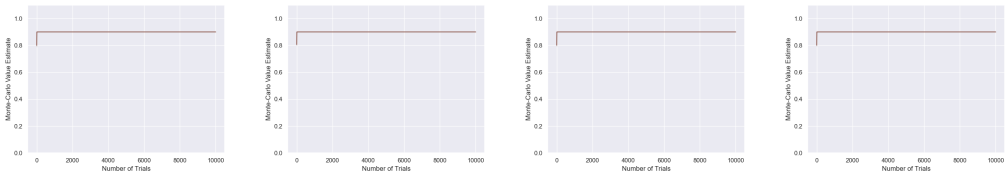
$\alpha = 1000, \epsilon = 0.01$ $\alpha = 1000, \epsilon = 0.1$ $\alpha = 1000, \epsilon = 1$ $\alpha = 1000, \epsilon = 10$



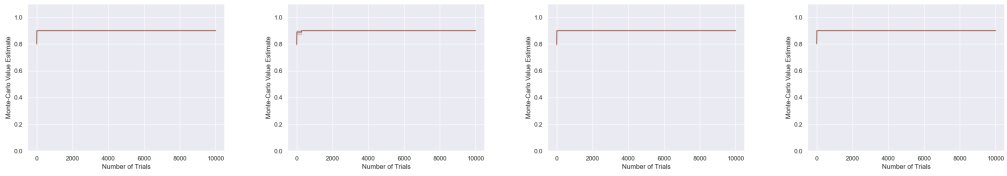
$\alpha = 100, \epsilon = 0.01$ $\alpha = 100, \epsilon = 0.1$ $\alpha = 100, \epsilon = 1$ $\alpha = 100, \epsilon = 10$



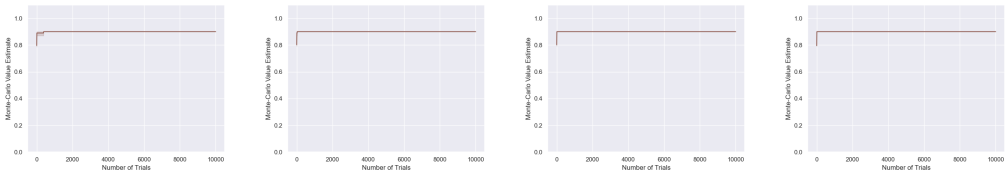
$\alpha = 10, \epsilon = 0.01$ $\alpha = 10, \epsilon = 0.1$ $\alpha = 10, \epsilon = 1$ $\alpha = 10, \epsilon = 10$



$\alpha = 1, \epsilon = 0.01$ $\alpha = 1, \epsilon = 0.1$ $\alpha = 1, \epsilon = 1$ $\alpha = 1, \epsilon = 10$

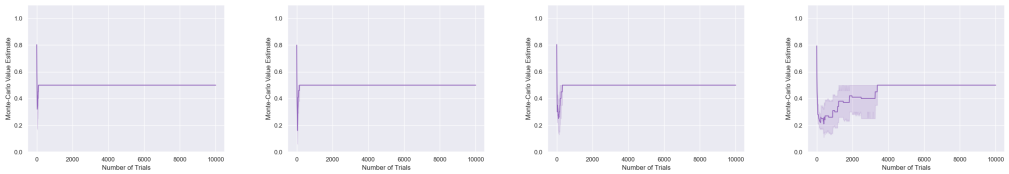


$\alpha = 0.1, \epsilon = 0.01$ $\alpha = 0.1, \epsilon = 0.1$ $\alpha = 0.1, \epsilon = 1$ $\alpha = 0.01, \epsilon = 10$

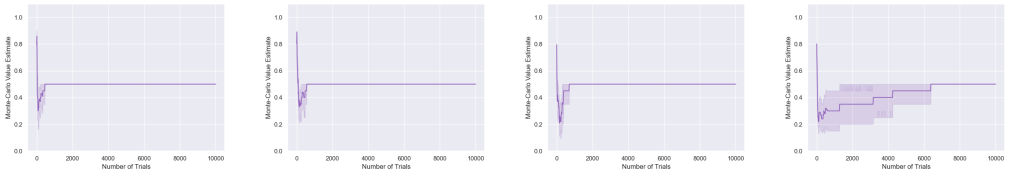


$\alpha = 0.01, \epsilon = 0.01$ $\alpha = 0.01, \epsilon = 0.1$ $\alpha = 0.01, \epsilon = 1$ $\alpha = 0.01, \epsilon = 10$

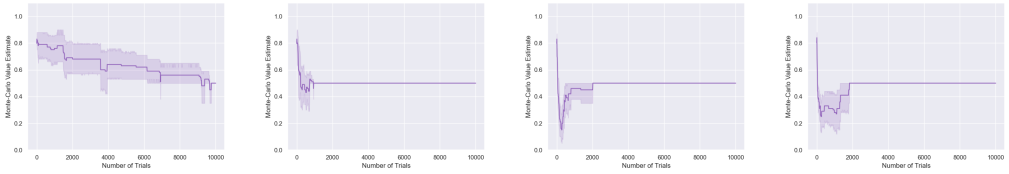
Figure 17: Results for RENTS on the modified 10-chain ($D = 10$, $R_f = 0.5$), for varying temperatures and exploration parameters.



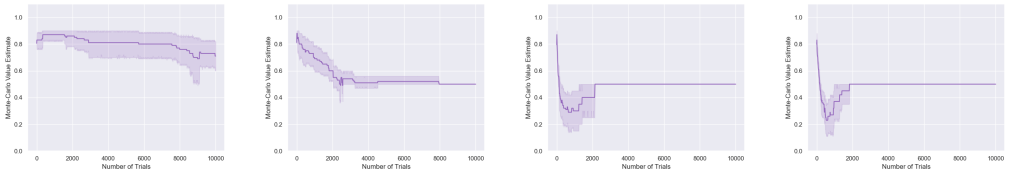
$\alpha = 10, \epsilon = 0.01$ $\alpha = 10, \epsilon = 0.1$ $\alpha = 10, \epsilon = 1$ $\alpha = 10, \epsilon = 10$



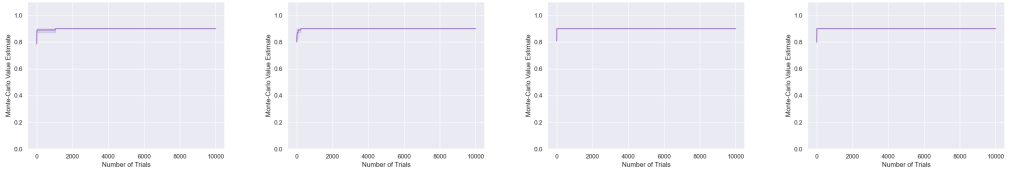
$\alpha = 1, \epsilon = 0.01$ $\alpha = 1, \epsilon = 0.1$ $\alpha = 1, \epsilon = 1$ $\alpha = 1, \epsilon = 10$



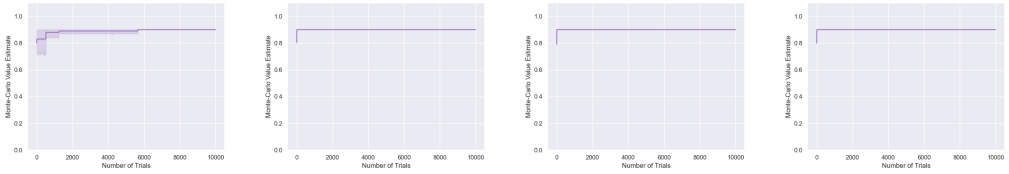
$\alpha = 0.7, \epsilon = 0.01$ $\alpha = 0.7, \epsilon = 0.1$ $\alpha = 0.7, \epsilon = 1$ $\alpha = 0.7, \epsilon = 10$



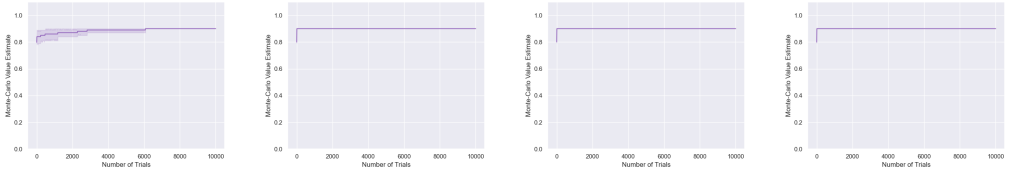
$\alpha = 0.5, \epsilon = 0.01$ $\alpha = 0.5, \epsilon = 0.1$ $\alpha = 0.5, \epsilon = 1$ $\alpha = 0.5, \epsilon = 10$



$\alpha = 0.3, \epsilon = 0.01$ $\alpha = 0.3, \epsilon = 0.1$ $\alpha = 0.3, \epsilon = 1$ $\alpha = 0.3, \epsilon = 10$

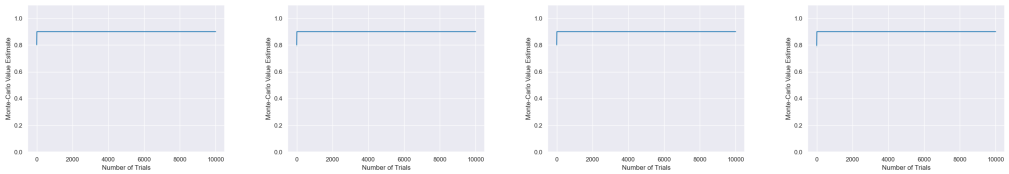


$\alpha = 0.1, \epsilon = 0.01$ $\alpha = 0.1, \epsilon = 0.1$ $\alpha = 0.1, \epsilon = 1$ $\alpha = 0.1, \epsilon = 10$

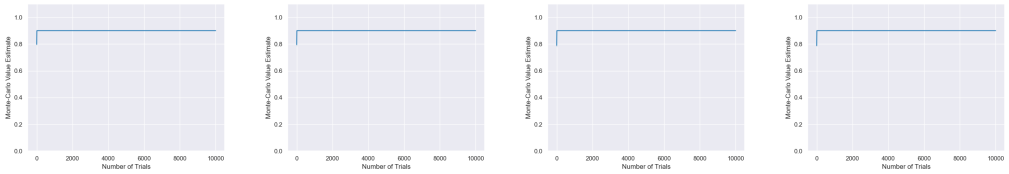


$\alpha = 0.01, \epsilon = 0.01$ $\alpha = 0.01, \epsilon = 0.1$ $\alpha = 0.01, \epsilon = 1$ $\alpha = 0.01, \epsilon = 10$

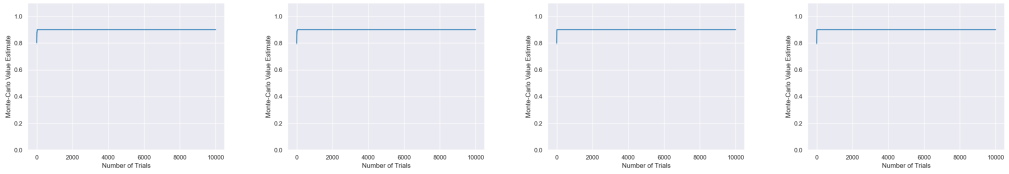
Figure 18: Results for TENTS on the modified 10-chain ($D = 10$, $R_f = 0.5$), for varying temperatures and exploration parameters.



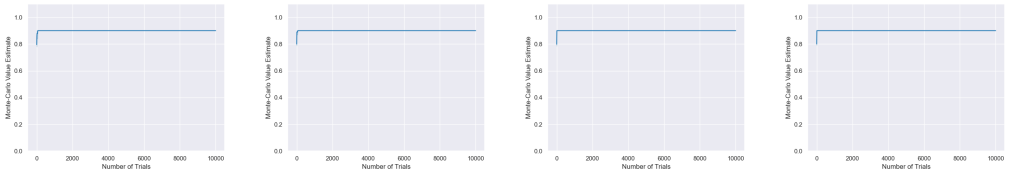
$\alpha = 1, \epsilon = 0.01$ $\alpha = 1, \epsilon = 0.1$ $\alpha = 1, \epsilon = 1$ $\alpha = 1, \epsilon = 10$



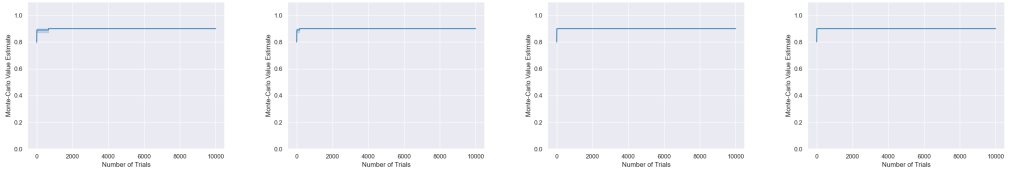
$\alpha = 0.5, \epsilon = 0.01$ $\alpha = 0.5, \epsilon = 0.1$ $\alpha = 0.5, \epsilon = 1$ $\alpha = 0.5, \epsilon = 10$



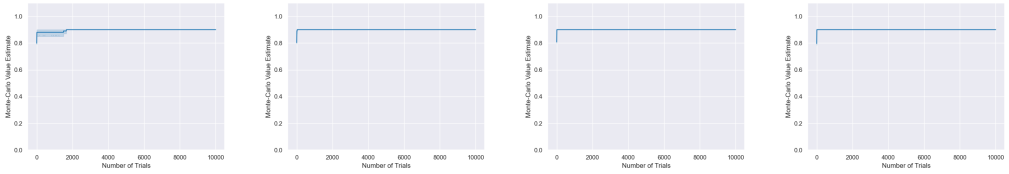
$\alpha = 0.2, \epsilon = 0.01$ $\alpha = 0.2, \epsilon = 0.1$ $\alpha = 0.2, \epsilon = 1$ $\alpha = 0.2, \epsilon = 10$



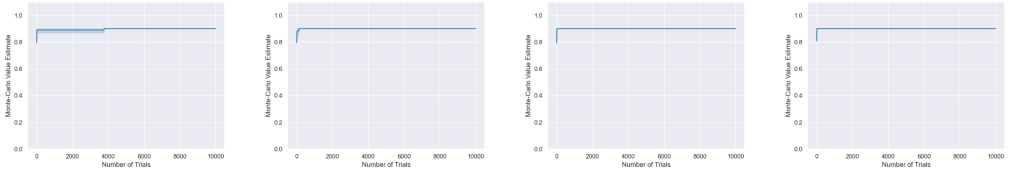
$\alpha = 0.15, \epsilon = 0.01$ $\alpha = 0.15, \epsilon = 0.1$ $\alpha = 0.15, \epsilon = 1$ $\alpha = 0.15, \epsilon = 10$



$\alpha = 0.1, \epsilon = 0.01$ $\alpha = 0.1, \epsilon = 0.1$ $\alpha = 0.1, \epsilon = 1$ $\alpha = 0.1, \epsilon = 10$

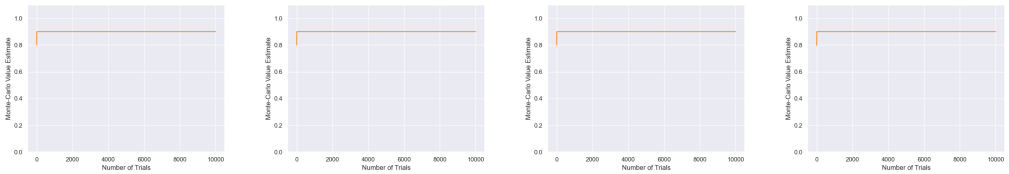


$\alpha = 0.05, \epsilon = 0.01$ $\alpha = 0.05, \epsilon = 0.1$ $\alpha = 0.05, \epsilon = 1$ $\alpha = 0.05, \epsilon = 10$

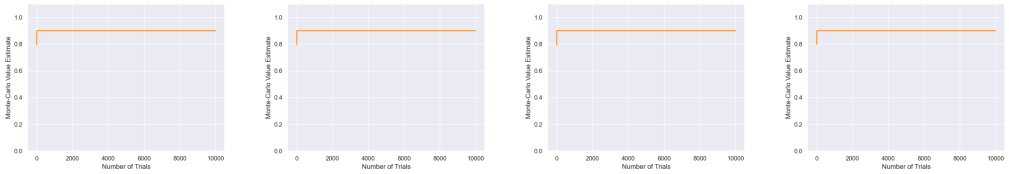


$\alpha = 0.01, \epsilon = 0.01$ $\alpha = 0.01, \epsilon = 0.1$ $\alpha = 0.01, \epsilon = 1$ $\alpha = 0.01, \epsilon = 10$

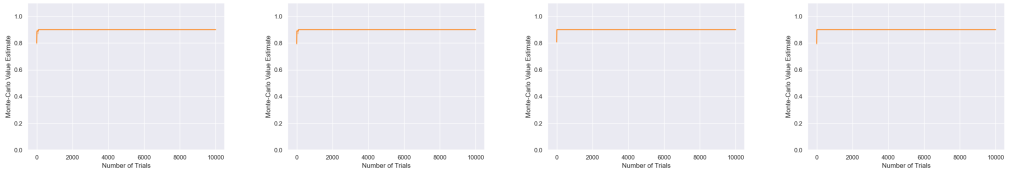
Figure 19: Results for BTS on the modified 10-chain ($D = 10$, $R_f = 0.5$), for varying temperatures and exploration parameters.



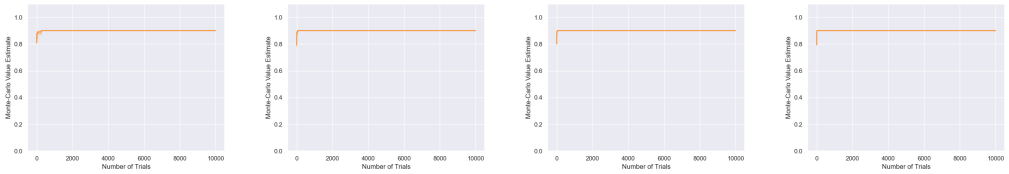
$\alpha = 1, \epsilon = 0.01$ $\alpha = 1, \epsilon = 0.1$ $\alpha = 1, \epsilon = 1$ $\alpha = 1, \epsilon = 10$



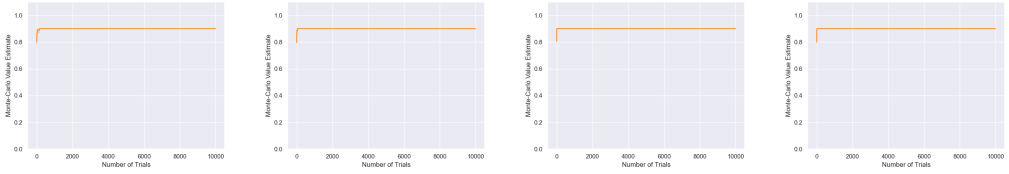
$\alpha = 0.5, \epsilon = 0.01$ $\alpha = 0.5, \epsilon = 0.1$ $\alpha = 0.5, \epsilon = 1$ $\alpha = 0.5, \epsilon = 10$



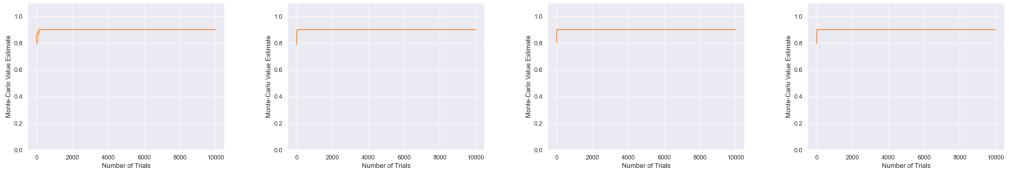
$\alpha = 0.2, \epsilon = 0.01$ $\alpha = 0.2, \epsilon = 0.1$ $\alpha = 0.2, \epsilon = 1$ $\alpha = 0.2, \epsilon = 10$



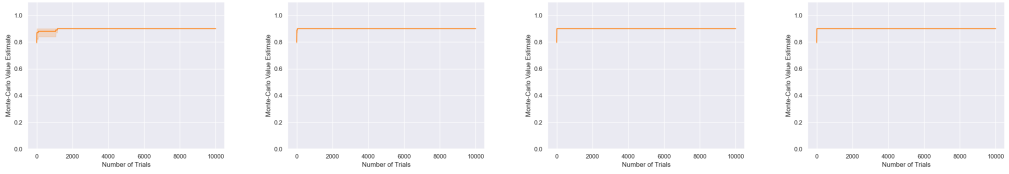
$\alpha = 0.15, \epsilon = 0.01$ $\alpha = 0.15, \epsilon = 0.1$ $\alpha = 0.15, \epsilon = 1$ $\alpha = 0.15, \epsilon = 10$



$\alpha = 0.1, \epsilon = 0.01$ $\alpha = 0.1, \epsilon = 0.1$ $\alpha = 0.1, \epsilon = 1$ $\alpha = 0.1, \epsilon = 10$



$\alpha = 0.05, \epsilon = 0.01$ $\alpha = 0.05, \epsilon = 0.1$ $\alpha = 0.05, \epsilon = 1$ $\alpha = 0.05, \epsilon = 10$



$\alpha = 0.01, \epsilon = 0.01$ $\alpha = 0.01, \epsilon = 0.1$ $\alpha = 0.01, \epsilon = 1$ $\alpha = 0.01, \epsilon = 10$

Figure 20: Results for DENTS on the modified 10-chain ($D = 10$, $R_f = 0.5$), for varying temperatures and exploration parameters. The decay function was set to $\beta(m) = \alpha / \log(e + m)$.

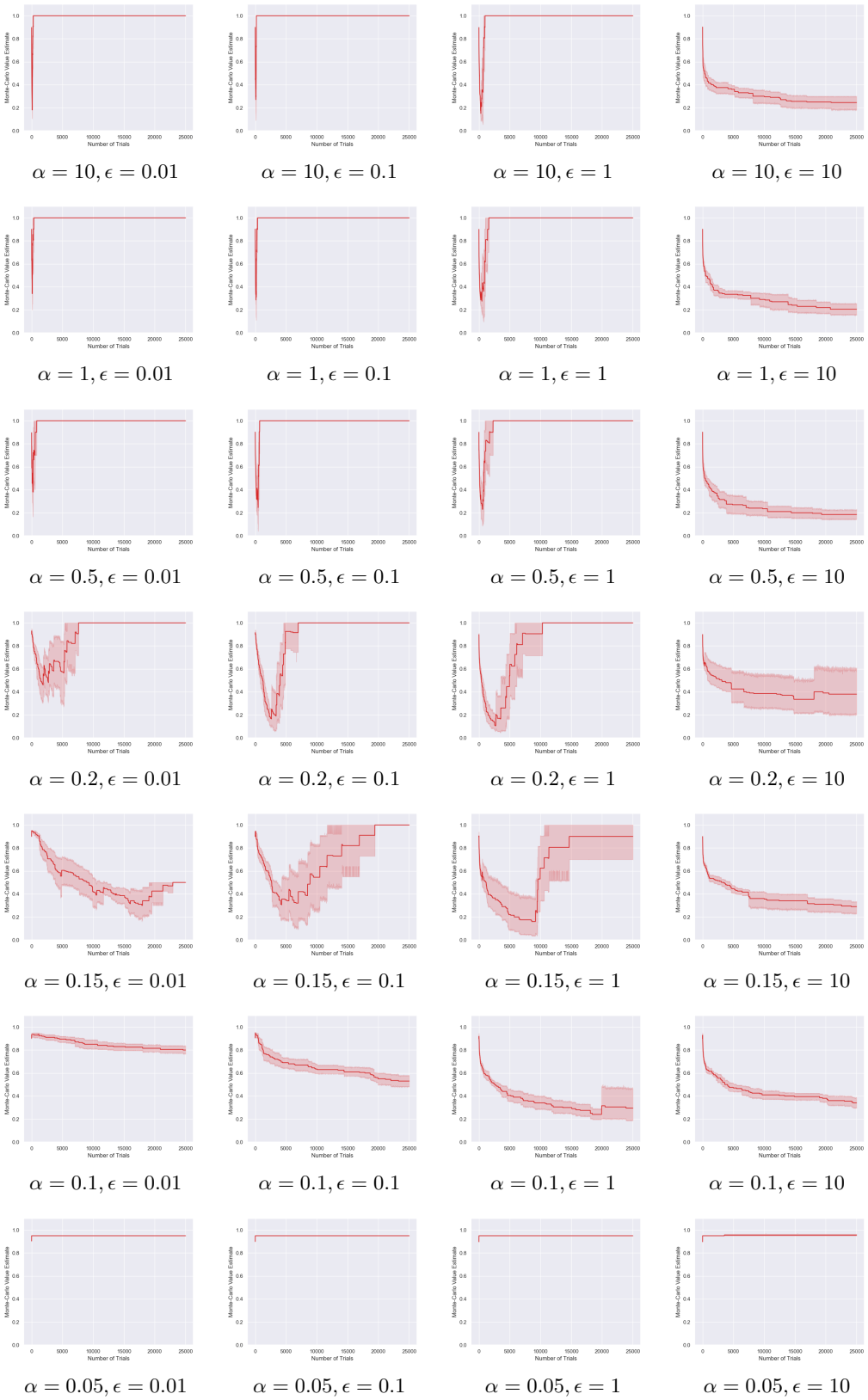


Figure 21: Results for MENTS on the 20-chain ($D = 20$, $R_f = 1.0$), for varying temperatures and exploration parameters.

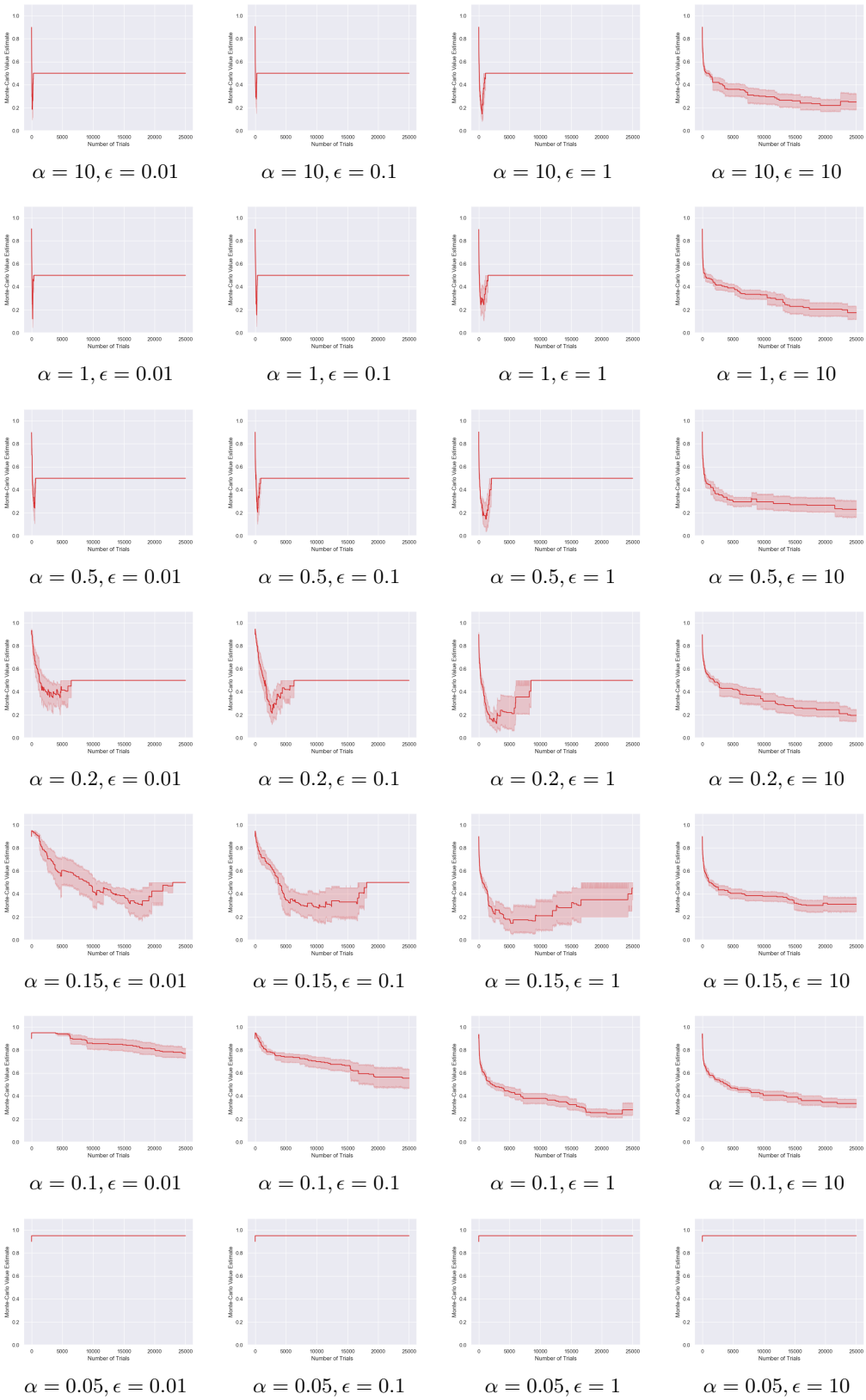
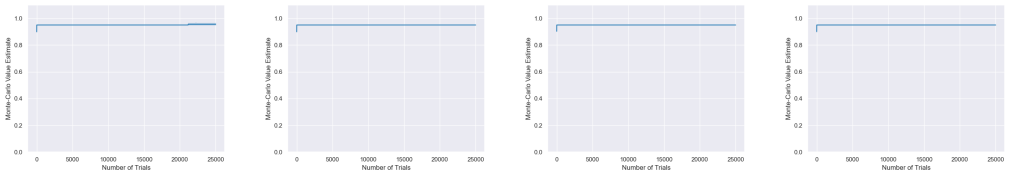
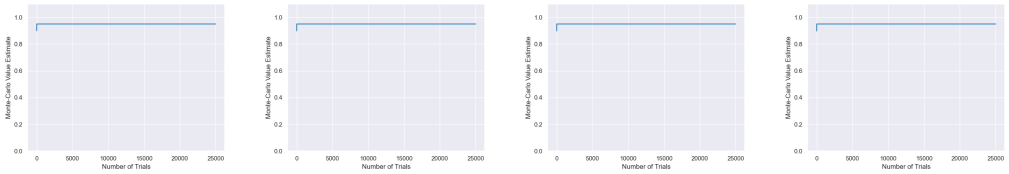


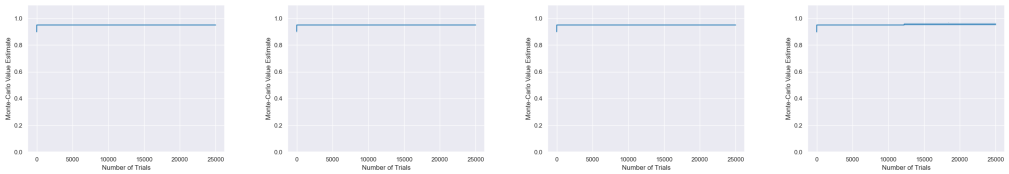
Figure 22: Results for MENTS on the modified 20-chain ($D = 20$, $R_f = 0.5$), for varying temperatures and exploration parameters.



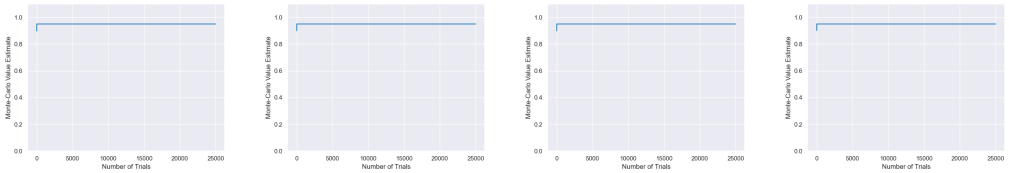
$\alpha = 10, \epsilon = 0.01$ $\alpha = 10, \epsilon = 0.1$ $\alpha = 10, \epsilon = 1$ $\alpha = 10, \epsilon = 10$



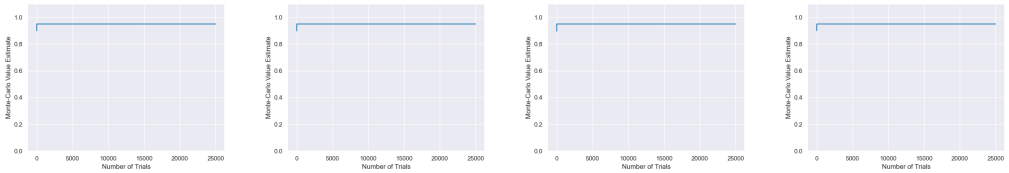
$\alpha = 1, \epsilon = 0.01$ $\alpha = 1, \epsilon = 0.1$ $\alpha = 1, \epsilon = 1$ $\alpha = 1, \epsilon = 10$



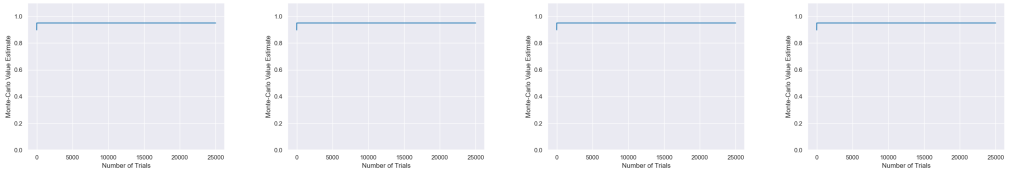
$\alpha = 0.5, \epsilon = 0.01$ $\alpha = 0.5, \epsilon = 0.1$ $\alpha = 0.5, \epsilon = 1$ $\alpha = 0.5, \epsilon = 10$



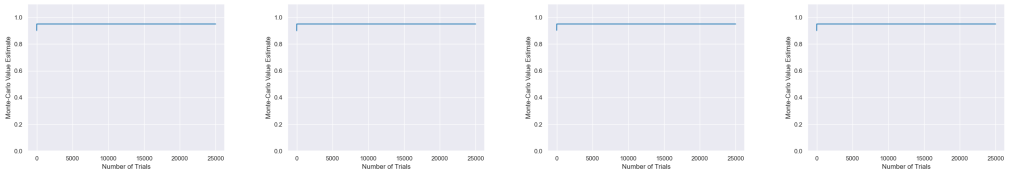
$\alpha = 0.2, \epsilon = 0.01$ $\alpha = 0.2, \epsilon = 0.1$ $\alpha = 0.2, \epsilon = 1$ $\alpha = 0.2, \epsilon = 10$



$\alpha = 0.15, \epsilon = 0.01$ $\alpha = 0.15, \epsilon = 0.1$ $\alpha = 0.15, \epsilon = 1$ $\alpha = 0.15, \epsilon = 10$

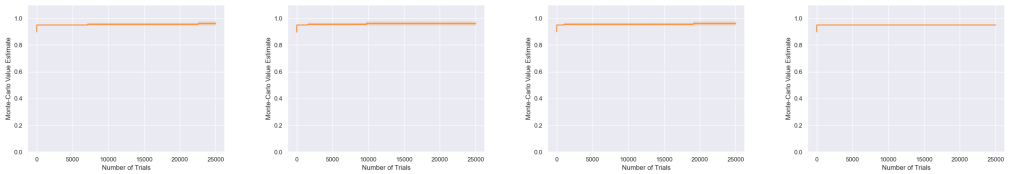


$\alpha = 0.1, \epsilon = 0.01$ $\alpha = 0.1, \epsilon = 0.1$ $\alpha = 0.1, \epsilon = 1$ $\alpha = 0.1, \epsilon = 10$

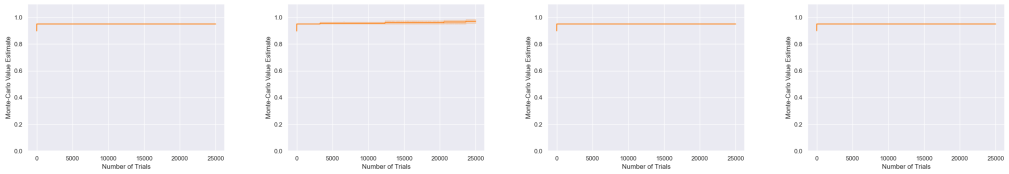


$\alpha = 0.05, \epsilon = 0.01$ $\alpha = 0.05, \epsilon = 0.1$ $\alpha = 0.05, \epsilon = 1$ $\alpha = 0.05, \epsilon = 10$

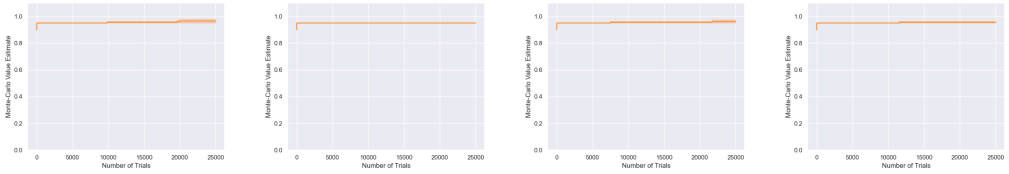
Figure 23: Results for BTS on the 20-chain ($D = 20$, $R_f = 1.0$), for varying temperatures and exploration parameters.



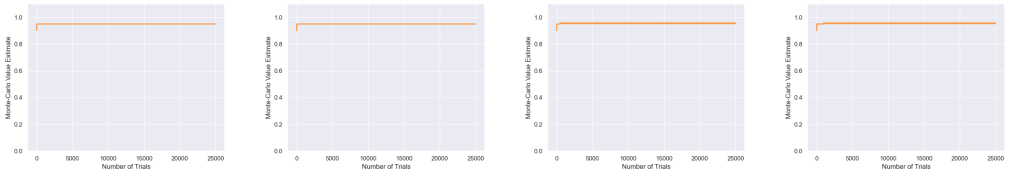
$\alpha = 10, \epsilon = 0.01$ $\alpha = 10, \epsilon = 0.1$ $\alpha = 10, \epsilon = 1$ $\alpha = 10, \epsilon = 10$



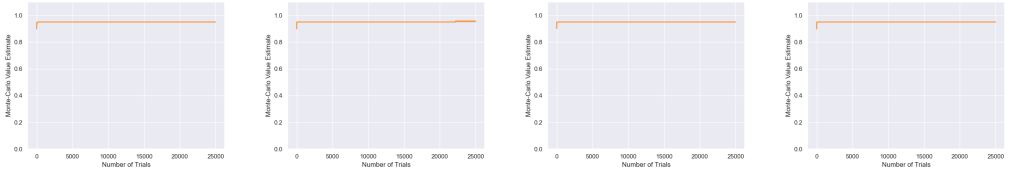
$\alpha = 1, \epsilon = 0.01$ $\alpha = 1, \epsilon = 0.1$ $\alpha = 1, \epsilon = 1$ $\alpha = 1, \epsilon = 10$



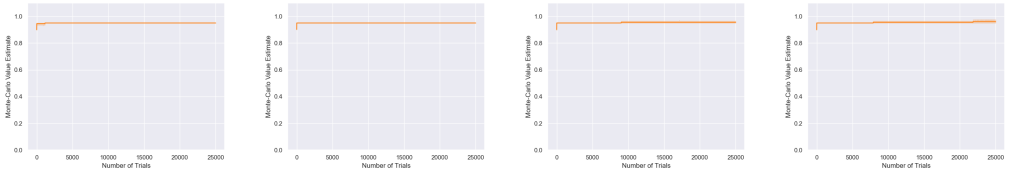
$\alpha = 0.5, \epsilon = 0.01$ $\alpha = 0.5, \epsilon = 0.1$ $\alpha = 0.5, \epsilon = 1$ $\alpha = 0.5, \epsilon = 10$



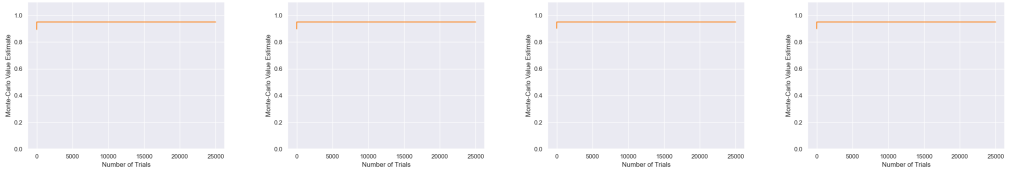
$\alpha = 0.2, \epsilon = 0.01$ $\alpha = 0.2, \epsilon = 0.1$ $\alpha = 0.2, \epsilon = 1$ $\alpha = 0.2, \epsilon = 10$



$\alpha = 0.15, \epsilon = 0.01$ $\alpha = 0.15, \epsilon = 0.1$ $\alpha = 0.15, \epsilon = 1$ $\alpha = 0.15, \epsilon = 10$

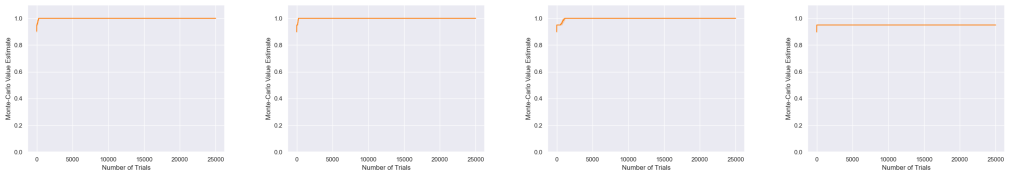


$\alpha = 0.1, \epsilon = 0.01$ $\alpha = 0.1, \epsilon = 0.1$ $\alpha = 0.1, \epsilon = 1$ $\alpha = 0.1, \epsilon = 10$

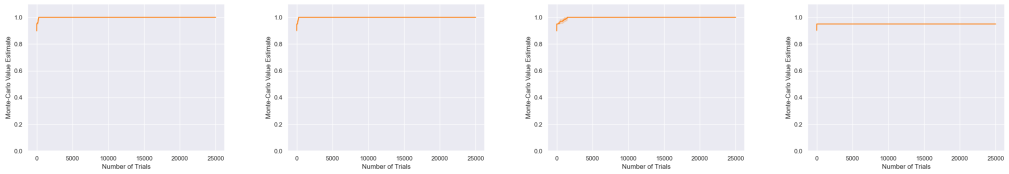


$\alpha = 0.05, \epsilon = 0.01$ $\alpha = 0.05, \epsilon = 0.1$ $\alpha = 0.05, \epsilon = 1$ $\alpha = 0.05, \epsilon = 10$

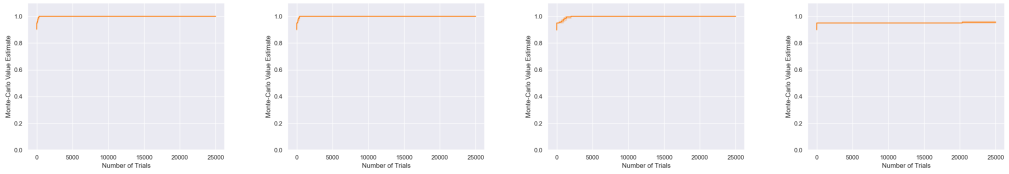
Figure 24: Results for DENTS on the 20-chain ($D = 20$, $R_f = 1.0$), for varying temperatures and exploration parameters. The decay function was set to $\beta(m) = \alpha / \log(e + m)$.



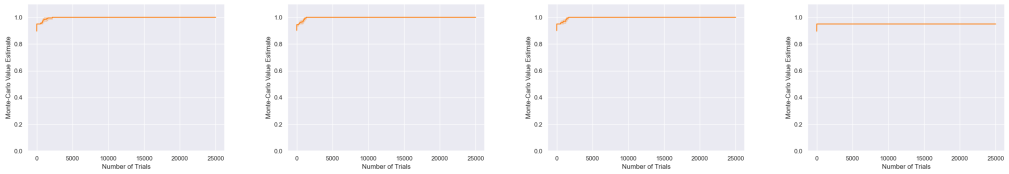
$\alpha = 10, \epsilon = 0.01$ $\alpha = 10, \epsilon = 0.1$ $\alpha = 10, \epsilon = 1$ $\alpha = 10, \epsilon = 10$



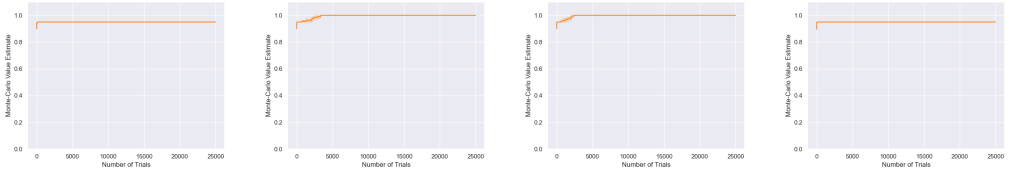
$\alpha = 1, \epsilon = 0.01$ $\alpha = 1, \epsilon = 0.1$ $\alpha = 1, \epsilon = 1$ $\alpha = 1, \epsilon = 10$



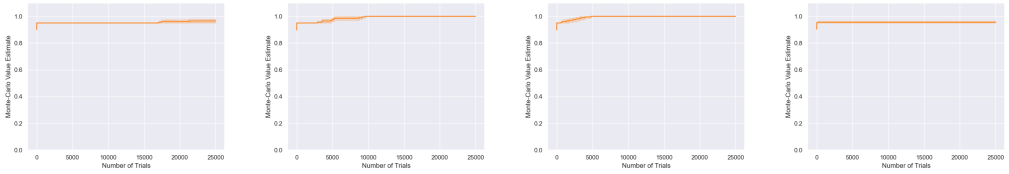
$\alpha = 0.5, \epsilon = 0.01$ $\alpha = 0.5, \epsilon = 0.1$ $\alpha = 0.5, \epsilon = 1$ $\alpha = 0.5, \epsilon = 10$



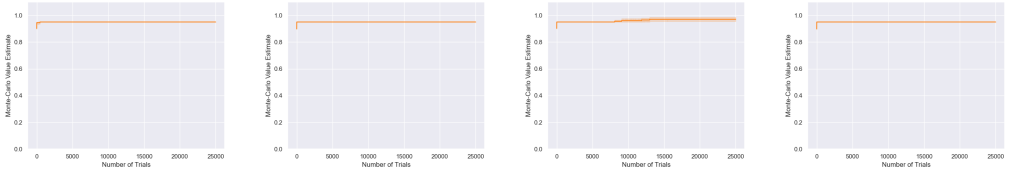
$\alpha = 0.2, \epsilon = 0.01$ $\alpha = 0.2, \epsilon = 0.1$ $\alpha = 0.2, \epsilon = 1$ $\alpha = 0.2, \epsilon = 10$



$\alpha = 0.15, \epsilon = 0.01$ $\alpha = 0.15, \epsilon = 0.1$ $\alpha = 0.15, \epsilon = 1$ $\alpha = 0.15, \epsilon = 10$



$\alpha = 0.1, \epsilon = 0.01$ $\alpha = 0.1, \epsilon = 0.1$ $\alpha = 0.1, \epsilon = 1$ $\alpha = 0.1, \epsilon = 10$



$\alpha = 0.05, \epsilon = 0.01$ $\alpha = 0.05, \epsilon = 0.1$ $\alpha = 0.05, \epsilon = 1$ $\alpha = 0.05, \epsilon = 10$

Figure 25: Results for DENTS on the 20-chain ($D = 20$, $R_f = 1.0$). The decay function was set to $\beta(m) = \alpha$, so that DENTS mimics MENTS search policy.

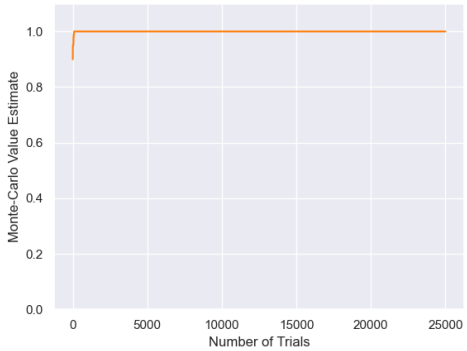


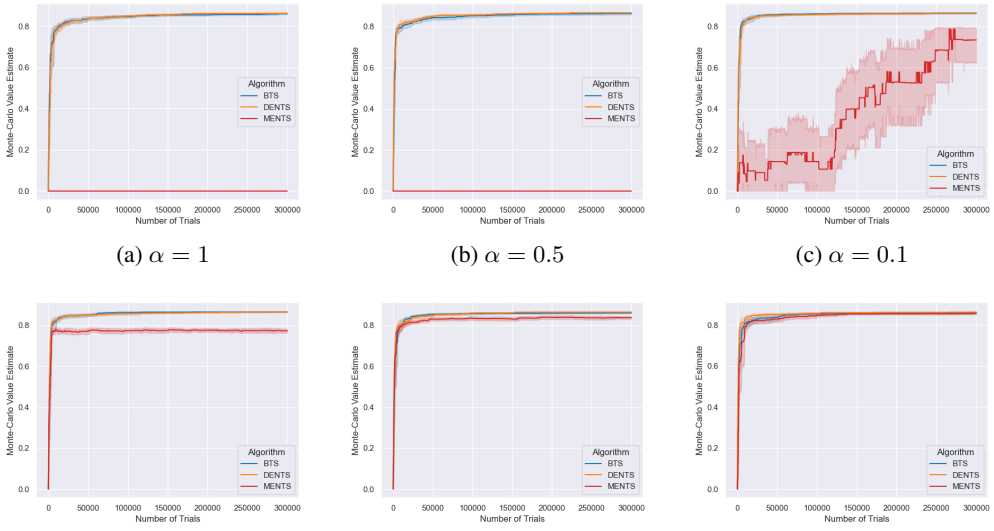
Figure 26: Results for DENTS on the 20-chain ($D = 20$, $R_f = 1.0$), with $\alpha = 0.5$, $\beta(m) = 10/\log(e + m)$ and $\epsilon = 0.01$.

D.2.2 Parameter sensitivity in Frozen Lake

We also ran MENTS, RENTS, TENTS, BTS and DENTS with a variety of temperatures on the 8x8 Frozen Lake environment given in Figure 6a. Again, we set $\beta(m) = \alpha/\log(e+m)$ for the decay function in DENTS, and we used an exploration parameter of $\epsilon = 1$ for all of the algorithms.

In Figures 27 and 29 we can see that MENTS and TENTS take the scenic route to the goal state for medium temperatures, where the reward for reaching the goal is still significant, but they can obtain more entropy reward by wondering around the gridworld for a while first. For higher temperatures they completely ignore the goal state, opting to rather maximise policy entropy. Interestingly, RENTS in Figure 28 fared better than MENTS and TENTS and never really ignored the goal state at the temperatures that we considered.

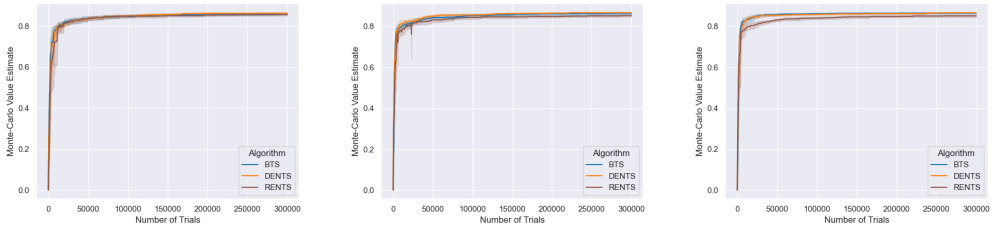
In contrast, both BTS and DENTS were agnostic to the temperature parameter in this environment (with $\epsilon = 1$) and were always able to find the goal state. We include the plots for BTS and DENTS in all of Figures 27, 29 and 28 for reference and as a comparison for MENTS, RENTS and TENTS.



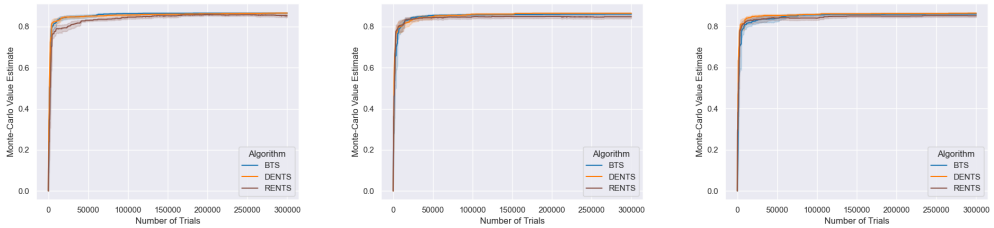
(a) $\alpha = 1$ (b) $\alpha = 0.5$ (c) $\alpha = 0.1$

(d) $\alpha = 0.05$ (e) $\alpha = 0.01$ (f) $\alpha = 0.005$

Figure 27: MENTS with a variety of temperatures on an 8x8 Frozen Lake environment. BTS and DENTS are included for reference.

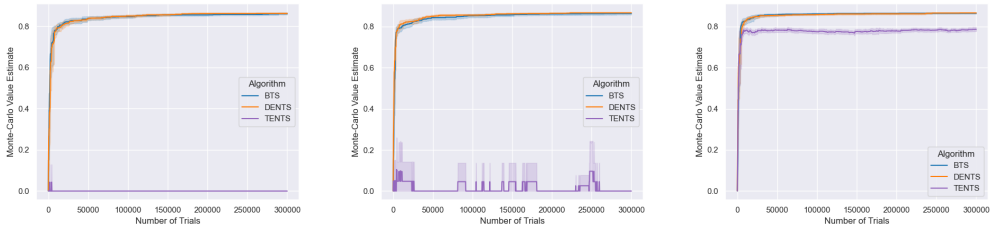


(a) $\alpha = 1$ (b) $\alpha = 0.5$ (c) $\alpha = 0.1$

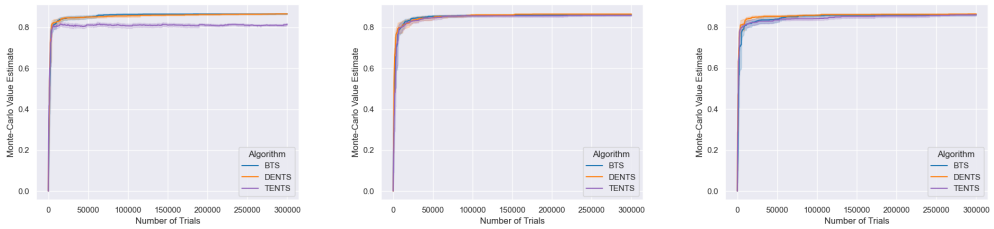


(d) $\alpha = 0.05$ (e) $\alpha = 0.01$ (f) $\alpha = 0.005$

Figure 28: RENTS with a variety of temperatures on an 8x8 Frozen Lake environment. BTS and DENTS are included for reference.



(a) $\alpha = 1$ (b) $\alpha = 0.5$ (c) $\alpha = 0.1$



(d) $\alpha = 0.05$ (e) $\alpha = 0.01$ (f) $\alpha = 0.005$

Figure 29: TENTS with a variety of temperatures on an 8x8 Frozen Lake environment. BTS and DENTS are included for reference.

D.3 Grid world hyper-parameter search and additional results

To select hyper-parameters for the experiments detailed in Section 5.1, we performed a hyper-parameter search. The search was run on the 8x12 Frozen Lake environment from Figure 6b, and the results in Section 5.1 were run on the 8x12 Frozen Lake envrionment from Figure 6c. For the Sailing problem, we performed the search using an initial wind direction of North, and the result in Section 5 used an initial wind direction of South-East.

To avoid the search space from becoming too large, we set some parameters manually. A good rule of thumb for initial values is to assure that $Q^{\text{init}}_{\text{sft}}(s,a) < Q^{*}_{\text{sft}}(s,a)$ and $Q^{\text{init}}(s,a) < Q^{*}(s,a)$. Explicitly this means that an initial value of zero is *not* a good choice for the Sailing problem, as rewards are negative (i.e. it has costs). In the Sailing environment, we actually set the initial values to -200, so that they were equal to the lowest possible return from a trial (the trial length was set to 50, and an agent can incur a cost of at most -4 per timestep). To simplify the search space, we initially set the decay function in DENTS to $\beta(m) = \alpha/\log(e+m)$ and tune it after.

For the remaining parameters, we considered all combinations of the following values:

- *UCT Bias*: [21], 100.0, 10.0, 1.0, 0.1;
- *MENTS exploration coefficient*: 2.0, 1.0, 0.3, 0.1, 0.03, 0.01;
- *Temperature*: 100.0, 10.0, 1.0, 0.1, 0.01, 0.001;
- *HMCTS UCT budget*: 100000, 30000, 10000, 3000, 1000, 300, 100, 30, 10.

Where a UCT bias of [21] refers to the adaptive bias introduced by Keller and Eyerich [21], and 'temperature' refers to the relevant temperature for the algorithm (i.e. search temperature in BTS and DENTS, and the temperature for Shannon/Relative/Tsallis entropy in MENTS/RENTS/DENTS). After that search, for DENTS we considered the decay functions of the form $\beta(m) = \beta_{\text{init}}/\log(e+m)$ and considered the following values:

- *DENTS initial entropy temperature* (β_{init}): 100.0, 10.0, 1.0, 0.1.

The final set of hyperparameters is given in Tables 3 and 4, which were used in the gridworld experiments in Section 5.1. Not included in the tables: [21] was selected for the UCT bias in both Frozen Lake and the Sailing problem.

D.4 DENTS with a constant β

To empirically demonstrate that DENTS search policy can mimic the search policy of MENTS, we ran DENTS with $\alpha = 1.0, \beta(m) = \alpha$ on the 10-chain environment, and compared it to MENTS with $\alpha = 1.0$. We also ran DENTS with $\alpha = 0.001, \beta(m) = \alpha$ in the Frozen Lake environment, and compared it to MENTS with $\alpha = 0.001$ which is what was selected in the hpyerparameter search (Appendix D.3). Results are given in Figure 30. Note that in the 10-chain, only MENTS has a dip in performance initially, which is due to the two algorithms using different recommendation policies.

D.5 Additional Go details, results and discussion

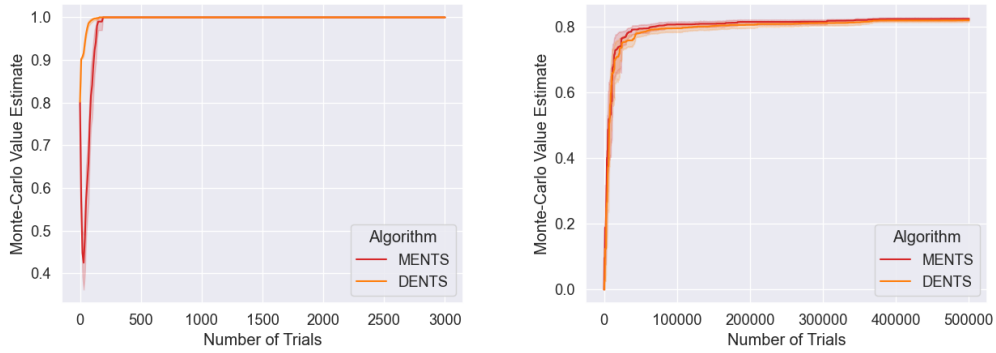
Recall from Appendix C.2 that we initialised values with the neural networks as $Q^{\text{init}}(s,a) = \log\tilde{\pi}(a|s) + B$ and $V^{\text{init}}(s) = \tilde{V}(s)$, where B is a constant (adapted from Xiao et al. [37]). For these

Algorithm	Exploration Parameter (ϵ)	Temperature (α)	Initial Values (Q^{init}, $Q^{\text{init}}_{\text{sft}}$)
MENTS	1.0	0.001	0
RENTS	2.0	0.001	0
TENTS	1.0	0.001	0
BTS	2.0	0.1	0
DENTS	1.0	0.1	0

Table 3: Final hyperparameters used for Frozen Lake in Section 5.1. Not included in the table: [21] was selected for the bias in UCT, [21] was selected for the bias and 3000 for the UCT budget in HMCTS and $\beta_{\text{init}} = 1.0$ was selected as the initial entropy temperature for DENTS.

Algorithm	Exploration Parameter (ϵ)	Temperature (α)	Initial Values (Q^{init}, $Q^{\text{init}}_{\text{sft}}$)
MENTS	1.0	10.0	-200
RENTS	1.0	10.0	-200
TENTS	2.0	0.1	-200
BTS	1.0	10.0	-200
DENTS	1.0	10.0	-200

Table 4: Final hyperparameters used for the Sailing Problem in Section 5.1. Not included in the table: [21] was selected for the bias in UCT, [21] was selected for the bias and 30 for the UCT budget in HMCTS and $\beta_{\text{init}} = 10.0$ was selected as the initial entropy temperature for DENTS.



(a) 10-chain. (b) Frozen Lake.

Figure 30: Comparing DENTS with MENTS, by setting $\beta_{\text{DENTS}}(m) = \alpha_{\text{MENTS}}, \alpha_{\text{DENTS}} = \alpha_{\text{MENTS}}$, where α_{MENTS} is the temperature used for MENTS, and $\alpha_{\text{DENTS}}, \beta_{\text{DENTS}}$ are the temperatures used by DENTS.

experiments we set a value of $B = \frac{-1}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} \log \tilde{\pi}(a|s)$. Initialising the values in such a way tends to lead to the values of $Q^{\text{init}}(s, a)$ being in the range $[-20, 20]$, to account for this, we scaled the results of the game to 100 and -100, which means that any parameters selected are about 100 times larger than they would be if we had not have used this scaling.

In these experiments we used a recommendation policy that recommends the action that was sampled the most, i.e. $\psi(s) = \max_a N(s, a)$, as this tends to be more robust to any noise from neural network outputs.

To select each parameter we ran a round robin tournament where we varied the value of one parameter. The agent that won the most games was used to select the parameter moving forward, and in the case of a tie we used the agent which won the most games. If the winning agent had the largest or smallest value, then we ran another tournament adjusting the values accordingly.

We tuned all of these algorithms using the game of 9x9 Go with a komi of 6.5, and giving each algorithm 2.5 seconds per move. For our final results we used the same parameters on 19x19 Go with a komi of 7.5, giving each algorithm 5.0 seconds per move.

D.5.1 Parameter selection for Go and supplimentary results

In this section we work through the process used to select parameters for the Go round robin tournament used in Section 5.2. Predominantly parameters were chosen by playing out games of Go between agents. The parameters used for PUCT were copied from Kata Go and Alpha Go Zero [36, 32].

Initially we set the values of $\epsilon, \epsilon_{\tilde{\lambda}}$ to $0.03, 1.0$. In Tables 5 and 6 we give results of tuning the search temperature for BTS and AR-BTS. For AR-BTS we tried different values of α_{init} in $\alpha(m) = \alpha_{\text{init}}/\sqrt{m}$.

Black \White	3	1	0.3	0.1	0.03
3	-	3-12	4-11	3-12	1-14
1	14-1	-	8-7	9-6	7-8
0.3	14-1	8-7	-	7-8	11-4
0.1	15-0	11-4	9-6	-	9-6
0.03	14-1	8-7	4-11	8-7	-

Table 5: Results for round robin to select the temperature parameter α for BTS. The value of 0.1 won all four of its matches so was selected.

Black \White	3	1	0.3	0.1	0.03
3	-	7-8	9-6	12-3	12-3
1	15-0	-	12-3	15-0	15-0
0.3	13-2	11-4	-	14-1	15-0
0.1	14-1	10-5	11-4	-	15-0
0.03	13-2	5-10	8-7	13-2	-

Table 6: Results for round robin to select the temperature parameter α_{init} for AR-BTS. The value of 1.0 won all four of its matches so was selected.

We then tuned the weighting of the prior policy $\epsilon_{\tilde{\lambda}}$. In Tables 7 and 8 we give results of tuning the prior policy weight for BTS and AR-BTS.

Following that, we then tuned MENTS exploration parameter ϵ. In Tables 9 and 10 we give results of tuning the exploration parameter for BTS and AR-BTS. Although we selected the lowest value we tried here, we note that with such a value that a random action would have been sampled very few times, so the result of the hyperparameter selection was essentially that ϵ should be as low as possible.

Then we tuned the (fixed and constant) search temperatures for MENTS, AR-MENTS, RENTS, AR-RENTS, TENTS and AR-TENTS in tables 11, 12,13, 14,15 and 16.

Finally, we considered entropy temperatures of the form $\beta(m) = \beta_{\text{init}}/\sqrt{m}$ for DENTS and AR-DENTS, and tuned the value of β_{init}, in Tables 17 and 18 for DENTS and AR-DENTS respectively.

After tuning all of the algorithms, we compared each algorithm to their AR version in table **??**, and the AR versions universally outperformed their counterparts. We used all of the selected parameters in 19x19 Go to run our final experiments given in Table 1.

Finally, we also ran AR-BTS against Kata Go directly limiting each algorithm to 1600 trials. KataGo beat AR-BTS by 31-19, confirming that the aditional exploration is outweighed by the information contained in the neural networks, and in Go the Boltzmann search algorithms gain their advantage via the Alias method and being able to run more trials quickly.

Black \White	10	5	3	2	1
10	-	5-10	3-12	1-14	3-12
5	14-1	-	12-3	11-4	11-4
3	15-0	15-0	-	12-3	12-3
2	15-0	12-3	12-3	-	10-5
1	15-0	14-1	12-3	9-6	-

Table 7: Results for round robin to select the weighting of the prior policy $\epsilon_{\tilde{\lambda}}$ for BTS. The value of 2.0 won the most matches so was selected.

Black \White	3	2	1	0.75	0.5
3	-	14-1	9-6	13-2	14-1
2	12-3	-	12-3	15-0	10-5
1	12-3	13-2	-	11-4	13-2
0.75	11-4	10-5	12-3	-	14-1
0.5	11-4	13-2	9-6	7-8	-

Table 8: Results for round robin to select the weighting of the prior policy $\epsilon_{\tilde{\lambda}}$ for AR-BTS. The value of 1.0 won the most matches so was selected.

Black \White	0.1	0.03	0.01	0.003	0.001
0.1	-	12-3	10-5	8-7	7-8
0.03	9-6	-	8-7	13-2	12-3
0.01	11-4	9-6	-	7-8	12-3
0.003	13-2	9-6	11-4	-	11-4
0.001	12-3	11-4	8-7	9-6	-

Table 9: Results for round robin to select the exploration parameter ϵ for BTS. The value of 0.003 won the most matches so was selected.

Black \White	0.1	0.03	0.01	0.003	0.001
0.1	-	12-3	14-1	12-3	13-2
0.03	15-0	-	11-4	14-1	14-1
0.01	15-0	11-4	-	13-2	13-2
0.003	15-0	14-0	14-1	-	13-2
0.001	14-1	14-1	14-1	15-0	-

Table 10: Results for round robin to select the exploration parameter ϵ for BTS. The value of 0.001 won the most matches so was selected.

Black \White	0.3	0.1	0.03	0.01	0.003
Black \White	3	1	0.3	0.1	0.03
3	-	9-6	9-6	11-4	10-5
1	11-4	-	11-4	9-6	10-5
0.3	9-6	7-8	-	11-4	6-9
0.1	4-11	4-11	0-15	-	4-11
0.03	2-13	2-13	0-15	0-15	-

Table 11: Results for round robin to select the temperature parameter α for MENTS. The value of 1.0 won the most matches so was selected.

Black \White	3	1	0.3	0.1	0.03
3	-	2-13	1-14	3-12	5-10
1	14-1	-	11-4	13-2	15-0
0.3	15-0	12-3	-	12-3	14-1
0.1	15-0	9-6	9-6	-	15-0
0.03	15-0	8-7	9-6	14-1	-

Table 12: Results for round robin to select the temperature parameter α for AR-MENTS. The value of 0.3 won the most matches so was selected.

Black \White	3	1	0.3	0.1	0.03
3	-	7-8	9-6	10-5	9-6
1	13-2	-	12-3	11-4	12-3
0.3	10-5	11-4	-	10-5	5-10
0.1	3-12	2-13	1-14	-	3-12
0.03	3-12	0-15	0-15	0-15	-

Table 13: Results for round robin to select the temperature parameter α for RENTS. The value of 1.0 won the most matches so was selected.

Black \White	3	1	0.3	0.1	0.03
3	-	4-11	2-13	9-6	3-12
1	15-0	-	12-3	13-2	13-2
0.3	15-0	13-2	-	14-1	15-0
0.1	15-0	10-5	13-2	-	15-0
0.03	13-2	13-2	12-3	14-1	-

Table 14: Results for round robin to select the temperature parameter α for AR-RENTS. The value of 0.3 won the most matches so was selected.

Black \White	300	100	30	10	3
300	-	3-12	5-10	7-8	9-6
100	6-9	-	9-6	12-3	12-3
30	8-7	7-8	-	9-6	11-4
10	0-15	2-13	2-13	-	6-9
3	1-14	1-14	2-13	5-10	-

Table 15: Results for round robin to select the temperature parameter α for TENTS. The value of 1.0 won the most matches so was selected.

Black \White	30	10	3	1	0.3
30	-	14-1	15-0	14-1	14-1
10	15-0	-	13-2	15-0	15-0
3	15-0	14-1	-	14-1	15-0
1	15-0	15-0	14-1	-	15-0
0.3	15-0	15-0	14-1	15-0	-

Table 16: Results for round robin to select the temperature parameter α for AR-TENTS. The value of 3.0 won the most matches so was selected.

Black \White	0.1	0.03	0.01	0.003	0.001
0.1	-	10-5	12-3	8-7	11-4
0.1	13-2	-	11-4	13-2	10-5
0.1	12-3	11-4	-	10-5	11-4
0.1	7-8	13-2	13-2	-	10-5
0.1	13-2	13-2	11-4	11-4	-

Table 17: Results for round robin to select the initial entropy temperature β_{init} for BTS. The value of 0.3 won the most matches so was selected.

Black \White	3	1	0.3	0.1	0.03
3	-	11-4	12-3	12-3	14-1
1	13-2	-	11-4	13-2	13-2
0.3	14-1	13-2	-	13-2	14-1
0.1	12-3	12-3	12-3	-	11-4
0.03	12-3	13-2	13-2	14-1	-

Table 18: Results for round robin to select the initial entropy temperature β_{init} for BTS. The value of 0.3 won the most matches so was selected.

Black \White	MENTS	AR-MENTS	BTS	AR-BTS	DENTS	AR-DENTS
MENTS	-	0-25				
AR-MENTS	25-0	-				
BTS			-	16-9		
AR-BTS			22-3	-		
DENTS					-	21-4
AR-DENTS					22-3	-
Black \White	RENTS	AR-RENTS	TENTS	AR-TENTS		
RENTS	-	2-23				
AR-RENTS	24-1	-				
TENTS			-	8-17		
AR-TENTS			23-2	-		

Table 19: Results for the matches of each algorithm against its AR version.

E Proofs

This proof section is structured as follows:

1. First, in Section E.1 we revisit MCTS as a stochastic process, defining some additional notation that was not useful in the main body of the paper, but will be for the following proofs;

2. Second, in Section E.2 we introduce preliminary results, that will be useful building blocks for proofs in later Theorems;

3. Third, in Section E.3 we show some general results about soft values that will also be useful later;

4. Fourth, in Section E.4 simple regret is then revisited, and we show that any bounds on the simple regret of a policy are equivalent to showing bounds on the simple regret of an action;

5. Fifth, in Section E.5 we show in a general way, that if a value function admits a concentration inequality, then the corresponding Q-value function admits a similar concentration inequality;

6. Sixth, in Section E.6 we show concentration inequalities for MENTS about the optimal soft values, and give bounds on the simple regret of MENTS, provided the temperature parameter is sufficiently small;

7. Seventh, in Sections E.7 and E.8 we also provide concentration inequalities around the optimal standard values for BTS and DENTS, and give simple regret bounds, irrespective of the temperature paramters;

8. Finally, in Section E.9 we consider results that are relavant for the algorithms using average returns from Section B.

E.1 Revisiting the MCTS stochastic process

In this subsection we recall the relevant details of the *MCTS stochastic process*, and introduce additional notation that will be useful in the proofs that were not necessary in the main paper. We will also recall the details for the UCT, MENTS, BTS and DENTS processes. We will use the phrase *any Boltzmann MCTS process* to refer to any one of the MENTS, BTS and DENTS processes. In particular, we index the (Q-)values with the number visits to the relevant node.

An MCTS process begins with a search tree $\mathcal{T}^0 = \{s_0\}$ that contains only the root node/initial state. Let $\tau^n = (s_0^n, a_0^n, ..., s_{h-1}^n, a_{h-1}^n, s_h^n)$ be a random variable for the nth trajectory or trial, where $s_0^n, ..., s_{h-1}^n \in \mathcal{T}^{n-1}$ and $s_h^n \notin \mathcal{T}^{n-1}$ (or $h = H$), and where actions are selected or sampled using π^n the search policy for the nth trial. A new node is added to the search tree $\mathcal{T}^n = \{s_h^n\} \cup \mathcal{T}^{n-1}$. (Q-)Value estimates are kept at each node in the tree, and new values are initialised using the functions $V^{\text{init}}(s)$ and $Q^{\text{init}}(s, a)$, which are typically implemented as a constant value, using a *rollout policy* or using a *neural network*. Finally, the (Q-)value estimates are updated in a backup phase.

In the following, it will be useful to write $N(s)$ the number of times state s was visited, and $N(s, a)$ the number of times action a was selected from state s as a sum of indicator random variables. Let $T(s_t)$ (and $T(s_t, a_t)$) be the set of trajectory indices that s_t was visited on (and action a_t selected), that is:

$$T(s_t) = \{i | s_t^i = s_t\} \tag{63}$$

$$T(s_t, a_t) = \{i | s_t^i = s_t, a_t^i = a_t\}. \tag{64}$$

This allows the counts $N(s_t)$, $N(s_t, a_t)$ and $N(s_{t+1})$ (with $s_{t+1} \in \text{Succ}(s_t, a_t)$) to be written as sums of indicator random variables:

$$N(s_t) = \sum_{i=1}^{n} \mathbb{1}[s_t^i = s_t] = |T(s_t)|, \tag{65}$$

$$N(s_t, a_t) = \sum_{i=1}^{n} \mathbb{1}[s_t^i = s_t, a_t^i = a_t] = |T(s_t, a_t)|, \tag{66}$$

$$N(s_t, a_t) = \sum_{i \in T(s_t)} \mathbb{1}[a_t^i = a_t], \tag{67}$$

$$N(s_{t+1}) = \sum_{i \in T(s_t, a_t)} \mathbb{1}[s_{t+1}^i = s_{t+1}]. \tag{68}$$

Additionally, we make the assumption that for any two states $s, s' \in \mathcal{S}$ that $s = s'$ if and only if the trajectories leading to them are identical. This assumption is purely to simplify notation, so that nodes in the tree have a one-to-one correspondence with states (or state-action pairs).

The UCT process. The UCT search policy can be defined as:

$$\pi^n_{\text{UCT}}(s_t) = \max_{a \in \mathcal{A}} \text{UCB}^n(s_t, a), \tag{69}$$

$$\text{UCB}^n(s_t, a) = \begin{cases} \infty & \text{if } N(s_t, a) = 0 \\ \bar{Q}^{N(s_t,a)}(s_t, a) + c\sqrt{\frac{\log N(s_t)}{N(s_t,a)}} & \text{if } N(s_t, a) > 0 \end{cases} \tag{70}$$

where, after n trials, $\bar{Q}^{N(s,a)}(s, a)$ is the empirical estimate of the value at node (s, a), where action a has been selected $N(s, a)$ from state s. The backup consists of updating empirical estimates for $t = h - 1, ..., 0$:

$$\bar{V}^{N(s_t)+1}(s_t) = \bar{V}^{N(s_t)}(s_t) + \frac{\bar{R}(t) - \bar{V}^{N(s_t)}(s_t)}{N(s_t) + 1}, \tag{71}$$

$$\bar{Q}^{N(s_t,a_t)+1}(s_t, a_t) = \bar{Q}^{N(s_t,a_t)}(s_t, a_t) + \frac{\bar{R}(t) - \bar{Q}^{N(s_t,a_t)}(s_t, a_t)}{N(s_t, a_t) + 1}, \tag{72}$$

where $\bar{R}(t) = V^{\text{init}}(s_h) + \sum_{i=t}^{h-1} R(s_i, a_i)$, and values are initialised as $\bar{V}^1(s) = V^{\text{init}}(s)$ and $\bar{Q}^0(s, a) = 0$.

The MENTS process. The policy for the MENTS process at state s on the nth trial is:

$$\pi^n_{\text{MENTS}}(a|s) = (1 - \lambda_s) \exp\left(\left(\hat{Q}^{N(s,a)}_{\text{sft}}(s, a) - \hat{V}^{N(s)}_{\text{sft}}(s)\right)/\alpha\right) + \frac{\lambda_s}{|\mathcal{A}|}, \tag{73}$$

where $\lambda_s = \min(1, \epsilon/\log(e + N(s)))$, $\epsilon \in (0, \infty)$. The estimated soft values are computed using the backups for $t = h - 1, ..., 0$:

$$\hat{V}^{N(s_t)}_{\text{sft}}(s_t) = \alpha \log \sum_{a \in \mathcal{A}} \exp\left(\hat{Q}^{N(s_t,a)}_{\text{sft}}(s_t, a)/\alpha\right), \tag{74}$$

$$\hat{Q}^{N(s_t,a_t)}_{\text{sft}}(s_t, a_t) = R(s_t, a_t) + \sum_{s' \in \text{Succ}(s,a)} \left(\frac{N(s')}{N(s_t, a_t)} \hat{V}^{N(s')}_{\text{sft}}(s')\right). \tag{75}$$

The soft (Q-)values are initialised as $\hat{V}^1_{\text{sft}}(s) = V^{\text{init}}(s)$ and $\hat{Q}^0_{\text{sft}}(s, a) = Q^{\text{init}}_{\text{sft}}(s)$ (typically $Q^{\text{init}}_{\text{sft}}(s) = 0$).

The DENTS process. Let $\beta : \mathbb{R} \to [0, \infty)$ be a bounded function. The policy for the DENTS process at state s on the nth trial is:

$$\pi^n_{\text{DENTS}}(a|s) = (1 - \lambda_s)\rho^n(a|s) + \frac{\lambda_s}{|\mathcal{A}|}, \tag{76}$$

where $\lambda_s = \min(1, \epsilon/\log(e + N(s)))$, $\epsilon \in (0, \infty)$ and where ρ is given by:

$$\rho^n_{\text{DENTS}}(a|s) \propto \exp\left(\frac{1}{\alpha}\left(\hat{Q}^{N(s,a)}(s,a) + \beta(N(s))\mathcal{H}_Q^{N(s,a)}(s,a)\right)\right). \tag{77}$$

The Bellman values and entropy values are computed using the backups for $t = h-1, ..., 0$:

$$\hat{Q}^{N(s_t,a_t)}(s_t,a_t) = R(s_t,a_t) + \sum_{s' \in \text{Succ}(s_t,a_t)} \left(\frac{N(s')}{N(s_t,a_t)}\hat{V}^{N(s')}(s')\right), \tag{78}$$

$$\hat{V}^{N(s_t)}(s_t) = \max_{a \in \mathcal{A}} \hat{Q}^{N(s_t,a)}(s_t,a), \tag{79}$$

$$\mathcal{H}_Q^{N(s_t,a_t)}(s_t,a_t) = \sum_{s' \in \text{Succ}(s_t,a_t)} \frac{N(s')}{N(s_t,a_t)}\mathcal{H}_V^{N(s')}(s'), \tag{80}$$

$$\mathcal{H}_V^{N(s_t)}(s_t) = \mathcal{H}(\pi^n_{\text{DENTS}}(\cdot|s_t)) + \sum_{a \in \mathcal{A}} \pi^n_{\text{DENTS}}(a_t|s_t)\mathcal{H}_Q^{N(s_t,a_t)}(s_t,a_t), \tag{81}$$

The (Q-)values are initialised as $\hat{V}^1(s) = V^{\text{init}}(s)$ and $\hat{Q}^0(s,a) = Q^{\text{init}}_{\text{sft}}(s)$ (typically $Q^{\text{init}}_{\text{sft}}(s) = 0$). The entropy values are initialised as $\mathcal{H}^0_Q(s,a) = 0$ and $\mathcal{H}^1_V(s) = \mathcal{H}(\pi^n_{\text{DENTS}}(\cdot|s_t))$, where the node for s is created on the nth trial.

Because we need to reason about π^n_{DENTS}, and subsequently ρ^n_{DENTS}, we give the exact form of ρ^n_{DENTS}:

$$\rho^n_{\text{DENTS}}(a|s) = \frac{\exp\left(\frac{1}{\alpha}\left(\hat{Q}^{N(s,a)}(s,a) + \beta(N(s))\mathcal{H}_Q^{N(s,a)}(s,a)\right)\right)}{\sum_{a' \in \mathcal{A}} \exp\left(\frac{1}{\alpha}\left(\hat{Q}^{N(s,a')}(s,a') + \beta(N(s))\mathcal{H}_Q^{N(s,a')}(s,a')\right)\right)}. \tag{82}$$

Let $V_\rho^{N(s)}(s)$ be defined as the value:

$$V_\rho^{N(s)}(s) = \alpha \log\left[\sum_{a' \in \mathcal{A}} \exp\left(\frac{1}{\alpha}\left(\hat{Q}^{N(s,a')}(s,a') + \beta(N(s))\mathcal{H}_Q^{N(s,a')}(s,a')\right)\right)\right], \tag{83}$$

and notice that the value of $\exp(V_\rho^{N(s)}(s)/\alpha)$ is equal to the denominator in Equation (82), and so by rearranging we can write ρ^n_{DENTS} as

$$\rho^n_{\text{DENTS}}(a|s) = \exp\left(\frac{1}{\alpha}\left(\hat{Q}^{N(s,a)}(s,a) + \beta(N(s))\mathcal{H}_Q^{N(s,a)}(s,a) - V_\rho^{N(s)}(s)\right)\right), \tag{84}$$

and subsequently, we can write

$$\pi^n_{\text{DENTS}}(a|s) = (1-\lambda_s)\exp\left(\frac{1}{\alpha}\left(\hat{Q}^{N(s,a)}(s,a) + \beta(N(s))\mathcal{H}_Q^{N(s,a)}(s,a) - V_\rho^{N(s)}(s)\right)\right) + \frac{\lambda_s}{|\mathcal{A}|}. \tag{85}$$

The BTS process. The BTS process is a special case of the DENTS process when $\beta(m) = 0$ for all $m \in \mathbb{N}$.

E.2 Preliminaries

Now we will provide lemmas are useful to avoid having to repeat the same argument in multiple proofs. In the following it will be useful to know that any action at any node in any MCTS process, for any number of trials, there is a minimum positive probability that it is chosen.

Lemma E.1. *Consider any Boltzmann MCTS process. There exists some $\pi^{\min} > 0$, such that for any state $s_t \in \mathcal{S}$, for all $a_t \in \mathcal{A}$ and any number of trials $n \in \mathbb{N}$ we have $\pi^n(a_t|s_t) > \pi^{\min}$.*

Proof outline. Firstly, we consider the case of the MENTS process. Define the $Q^{\min}$ function as follows:

$$Q^{\min}(s_t,a_t) = \min_{s_{t+1},a_{t+1},...,s_H,a_H} \sum_{i=t}^{H} \min(0, Q^{\text{init}}_{\text{sft}}(s_i), R(s_i,a_i)). \tag{86}$$

And define the $V^{\max}$ function as:

$$V^{\max}(s_t) = \alpha \log \sum_{a \in \mathcal{A}} \exp(Q^{\max}(s_t, a)/\alpha), \tag{87}$$

$$Q^{\max}(s_t, a_t) = R(s_t, a_t) + \max_{s_{t+1} \in \text{Succ}(s_t, a_t)} V^{\max}(s_{t+1}). \tag{88}$$

Via induction, it is possible to show that $Q^{\min}(s_t, a_t) \leq \hat{Q}_{\text{sft}}^{N(s_t,a_t)}(s_t, a_t)$ and $V^{\max}(s_t) \geq \hat{V}_{\text{sft}}^{N(s_t)}(s_t)$. Now, define $\pi^{\min}$ as:

$$\pi^{\min} = \inf_{\lambda \in [0,1]} \min_{(s,a) \in \mathcal{S} \times \mathcal{A}} (1-\lambda) \exp\left(\left(Q^{\min}(s,a) - V^{\max}(s)\right)/\alpha\right) + \frac{\lambda}{|\mathcal{A}|}. \tag{89}$$

Because the value of an exponential is positive, as is $1/|\mathcal{A}|$, it follows that $\pi^{\min} > 0$. Recall the MENTS policy (Equation (73)). By the monotonicity of the exponential function, it follows that for any $s_t \in \mathcal{S}, a_t \in \mathcal{A}, n \in \mathbb{N}$:

$$\pi^n(a_t|s_t) = (1-\lambda_{s_t}) \exp\left(\left(\hat{Q}_{\text{sft}}^{N(s_t,a_t)}(s_t, a_t) - \hat{V}_{\text{sft}}^{N(s_t)}(s_t)\right)/\alpha\right) + \frac{\lambda_{s_t}}{|\mathcal{A}|} \tag{90}$$

$$\geq (1-\lambda_{s_t}) \exp\left(\left(Q^{\min}(s_t, a_t) - V^{\max}(s_t)\right)/\alpha\right) + \frac{\lambda_{s_t}}{|\mathcal{A}|} \tag{91}$$

$$\geq \pi^{\min}. \tag{92}$$

Now we consider the DENTS process.

For the DENTS process we can use similar reasoning, but need to update the definition of $V^{\max}$ from Equation (87) to:

$$V^{\max}(s_t) = \alpha \log \sum_{a \in \mathcal{A}} \exp\left(\left(Q^{\max}(s_t, a) + H\beta_{\max} \log |\mathcal{A}|\right)/\alpha\right), \tag{93}$$

where $\beta_{\max} = \sup_{x \in \mathbb{R}} \beta(x)$. Similarly to the MENTS process case, we can show that $Q^{\min}(s_t, a_t) \leq \hat{Q}^{N(s_t,a_t)}(s_t, a_t)$ and $V^{\max}(s_t) \geq V_\rho^{N(s_t)}$, which implicitly uses that $0 \leq \mathcal{H}_Q^{N(s_t,a_t)}(s_t, a_t) \leq (H-t) \log |\mathcal{A}| \leq H \log |\mathcal{A}|$, which can be shown with an inductive argument, and using well-known properties of entropy. Defining $\pi^{\min}$ in the same way as Equation (89), with the updated definition of $V^{\max}$. Recalling the DENTS policy (Equation (85)), and using similar reasoning to before, we have:

$$\pi^n(a_t|s_t) = (1-\lambda_s) \exp\left(\frac{1}{\alpha}\left(\hat{Q}^{N(s,a)}(s,a) + \beta(N(s))\mathcal{H}_Q^{N(s,a)}(s,a) - V_\rho^{N(s)}(s)\right)\right) + \frac{\lambda_s}{|\mathcal{A}|} \tag{94}$$

$$\geq (1-\lambda_s) \exp\left(\frac{1}{\alpha}\left(\hat{Q}^{N(s,a)}(s,a) - V_\rho^{N(s)}(s)\right)\right) + \frac{\lambda_s}{|\mathcal{A}|} \tag{95}$$

$$\geq (1-\lambda_{s_t}) \exp\left(\left(Q^{\min}(s_t, a_t) - V^{\max}(s_t)\right)/\alpha\right) + \frac{\lambda_{s_t}}{|\mathcal{A}|} \tag{96}$$

$$\geq \pi^{\min}. \tag{97}$$

□

It will also be useful in the following that the union of exponentially unlikely events is also exponentially unlikely, and that the intersection of exponentially likely events is exponentially likely:

Lemma E.2. *Let $A_1, ..., A_\ell$ be some events that satisfy for $1 \leq i \leq \ell$ the inequality $\Pr(\neg A_i) \leq C_i \exp(-k_i)$ then:*

$$\Pr\left(\bigcup_{i=1}^{\ell} \neg A_i\right) \leq C \exp(-k), \tag{98}$$

$$\Pr\left(\bigcap_{i=1}^{\ell} A_i\right) = 1 - \Pr\left(\bigcup_{i=1}^{\ell} \neg A_i\right) \geq 1 - C \exp(-k), \tag{99}$$

where $C = \sum_{i=1}^{\ell} C_i$ and $k = \min_i k_i$.

Proof outline. Lemma E.2 is a consequence of the union bound, rearranging and simplifying. Inequality (99) is a consequence from Inequality (98) by negating the events. □

Additionally, Hoeffding's inequality will be useful to bound the difference between a sum of indicator random variables and its expectation.

Theorem E.3. *Let $\{X_i\}_{i=1}^m$ be indicator random variables (i.e. $X_i \in \{0,1\}$), and $S_m = \sum_{i=1}^m X_i$. Then Hoeffding's inequality for indicator random variables states for any $\varepsilon > 0$ that:*

$$\Pr(|S_m - \mathbb{E}S_m| > \varepsilon) \leq 2\exp\left(-\frac{2\varepsilon^2}{m}\right). \tag{100}$$

Proof. This is a specific case of Hoeffding's inequality. See [43] for proof. □

It will also be convenient to be able to 'translate' bounds that depend on some $N(s,a)$ to a corresponding bound on $N(s)$:

Lemma E.4. *Consider any Boltzmann MCTS process. Suppose that every action a_t has some minimum probability η of being chosen from some state s_t (irrespective of the number of trials), i.e. $\Pr(a_t^i = a_t | s_t^i = s_t) > \eta$. And suppose for some $C', k' > 0$ that some event E admits a bound:*

$$\Pr(E) \leq C'\exp(-k'N(s_t,a_t)). \tag{101}$$

Then, there exists $C, k > 0$ such that:

$$\Pr(E) \leq C\exp(-kN(s_t)). \tag{102}$$

Proof. Recall from Equation (67) that $N(s_t,a_t) = \sum_{i\in T(s_t)} \mathbb{1}[a_t^i = a_t] = \sum_{i\in T(s_t)} \mathbb{1}[a_t^i = a_t | s_t^i = s_t]$. By taking expectations and using the assumed $\Pr(a_t^i = a_t | s_t^i = s_t) \geq \eta$ we have $\mathbb{E}N(s_t,a_t) \geq \eta N(s_t)$ (and more specifically as a consequence $\mathbb{E}N(s_t,a_t) - \eta N(s_t)/2 \geq \eta N(s_t)/2$). The probability of $N(s_t,a_t)$ being below a multiplicative ratio of $N(s_t)$ is bounded as follows:

$$\begin{aligned}
\Pr\left(N(s_t,a_t) < \frac{1}{2}\eta N(s_t)\right) &\leq \Pr\left(N(s_t,a_t) < \mathbb{E}N(s_t,a_T) - \frac{1}{2}\eta N(s_t)\right) && (103)\\
&= \Pr\left(\mathbb{E}N(s_t,a_t) - N(s_t,a_t) > \frac{1}{2}\eta N(s_t)\right) && (104)\\
&\leq \Pr\left(|\mathbb{E}N(s_t,a_t) - N(s_t,a_t)| > \frac{1}{2}\eta N(s)\right) && (105)\\
&\leq 2\exp\left(-\frac{1}{2}\eta^2 N(s_t)\right). && (106)
\end{aligned}$$

The first line follows from $\mathbb{E}N(s_t,a_t) - \eta N(s_t)/2 \geq \eta N(s_t)/2$, the second line is a rearrangement, the third line comes from the inequality in (104) implying the inequality in (105), and the final line uses Theorem E.3, a Hoeffding bound for the sum of indicator random variables.

Finally, the bound using $N(s_t,a_t)$ can be converted into one depending on $N(s_t)$ as follows:

$$\begin{aligned}
\Pr(E) =& \Pr\left(E \middle| N(s_t,a_t) \geq \frac{1}{2}\eta N(s_t)\right)\Pr\left(N(s_t,a_t) \geq \frac{1}{2}\eta N(s_t)\right) && (107)\\
&+ \Pr\left(E \middle| N(s_t,a_t) < \frac{1}{2}\eta N(s_t)\right)\Pr\left(N(s_t,a_t) < \frac{1}{2}\eta N(s_t)\right) && (108)\\
\leq& \Pr\left(E \middle| N(s_t,a_t) \geq \frac{1}{2}\eta N(s_t)\right)\cdot 1 && (109)\\
&+ 1\cdot\Pr\left(N(s_t,a_t) < \frac{1}{2}\eta N(s_t)\right) && (110)\\
\leq& C'\exp\left(-\frac{1}{2}k'\eta N(s_t)\right) + 2\exp\left(-\frac{1}{2}\eta^2 N(s_t)\right) && (111)\\
\leq& C\exp(-kN(s_t)), && (112)
\end{aligned}$$

where (111) uses the assumed Inequality (101) with the condition $N(s_t, a_t) \geq \eta N(s_t)/2$, and also uses the bound from (106). On the last line there is an appropriate $C, k > 0$, similar to Lemma E.2. □

Similar to Lemma E.4, if we have some $s' \in \text{Succ}(s, a)$, it will also be convenient to be able to translate bounds that depend on $N(s')$ into bounds that depend on $N(s, a)$:

Lemma E.5. *Consider any Boltzmann MCTS process. For some state action pair (s_t, a_t), and for some $s'_{t+1} \in \text{Succ}(s_t, a_t)$, suppose for some $C', k' > 0$ that some event E admits a bound:*

$$\Pr(E) \leq C' \exp(-k' N(s'_t)). \tag{113}$$

Then, there exists $C, k > 0$ such that:

$$\Pr(E) \leq C \exp(-k N(s_t, a_t)). \tag{114}$$

Proof outline. Proof is similar to Lemma E.4. Instead of having a minimum probability of selecting an action η, replace it with the corresponding probability from the transition distribution $p(s_{t+1}|s_t, a_t)$. Then swapping any $N(s_t)$ with $N(s_t, a_t)$, any $N(s_t, a_t)$ with $N(s'_t)$, and using $N(s_{t+1}) = \sum_{i \in T(s_t, a_t)} \mathbb{1}[s^i_{t+1} = s_{t+1}]$ (Equation (68)) as the sum of indicator random variables will give the result. □

E.3 Soft learning

For this subsection we will temporarily reintroduce the time-step parameter into our value functions to simplify other notation. We will show two results about soft Q-values: that the optimal standard Q-value is less than the optimal soft Q-value, and, that given a sufficiently small temperature, the optimal soft Q-values will preserve any *strict* ordering over actions given by the optimal standard Q-values.

For some policy π, the definition of V^π_{sft} (Equation (3)) can be re-arranged, to give a relation between the soft Q-value, the standard Q-value and the entropy of the policy:

$$Q^\pi_{\text{sft}}(s, a, t) = Q^\pi(s, a, t) + \alpha \mathbb{E}_\pi \left[\sum_{i=t+1}^{H} \mathcal{H}\left(\pi(\cdot|s_i)\right) \middle| s_t = s, a_t = a \right], \tag{115}$$

$$= Q^\pi(s, a, t) + \alpha \mathcal{H}_{t+1}(\pi | s_t = s, a_t = a), \tag{116}$$

where $\mathcal{H}_{t+1}$ is used as a shorthand for the entropy term. By using this relation, it can be shown that the optimal soft Q-value will always be at least as large as the optimal standard Q-value:

Lemma E.6. $Q^*(s, a, t) \leq Q^*_{\text{sft}}(s, a, t)$.

Proof. Taking a maximum over policies in Equation (115), and considering that π^*, the optimal standard policy, is one of the possible policies considered in the maximisation, gives the result:

$$Q^*_{\text{sft}}(s, a, t) = \max_\pi \left(Q^\pi(s, a, t) + \alpha \mathcal{H}_{t+1}(\pi | s_t = s, a_t = a) \right) \tag{117}$$

$$\geq Q^{\pi^*}(s, a, t) + \alpha \mathcal{H}_{t+1}(\pi^* | s_t = s, a_t = a) \tag{118}$$

$$\geq Q^*(s, a, t). \tag{119}$$

Noting that the entropy function is non-negative function. □

The optimal soft and standard values can be 'tied together' by picking a very low temperature. Let $\delta(s, t)$ be the set of actions that have different optimal standard Q-values, that is $\delta(s, t) = \{(a, a') | Q^*(s, a, t) \neq Q^*(s, a', t)\}$. Now define $\Delta_\mathcal{M}$ as follows:

$$\Delta_{s,t} = \min_{(a,a') \in \delta(s,t)} |Q^*(s, a, t) - Q^*(s, a', t)|, \tag{120}$$

$$\Delta_\mathcal{M} = \min_{s,t} \Delta_{s,t}. \tag{121}$$

Note in particular, for some $(a, a') \in \delta(s, t)$ that the definition of $\Delta_\mathcal{M}$ implies that if $Q^*(s, a, t) < Q^*(s, a', t)$ then

$$Q^*(s, a, t) + \Delta_\mathcal{M} \leq Q^*(s, a', t). \tag{122}$$

Lemma E.7. *If $\alpha < \Delta_{\mathcal{M}}/H \log |\mathcal{A}|$, then for all $t = 1, ..., H$, for all $s \in \mathcal{S}$ and for all $(a, a') \in \delta(s, t)$ we have $Q^*_{\text{sft}}(s, a, t) < Q^*_{\text{sft}}(s, a', t)$ iff $Q^*(s, a, t) < Q^*(s, a', t)$.*

Proof. ($\Leftarrow$) First consider that the optimal soft Q-value is less than or equal to the optimal standard Q-value and maximum possible entropy:

$$\begin{aligned} Q^*_{\text{sft}}(s, a, t) &= \max_{\pi} \left(Q^{\pi}(s, a, t) + \alpha \mathcal{H}_{t+1}(\pi)\right) && (123)\\ &\leq \max_{\pi} Q^{\pi}(s, a, t) + \max_{\pi} \alpha \mathcal{H}_{t+1}(\pi) && (124)\\ &= Q^*(s, a, t) + \alpha(H - t) \log |\mathcal{A}| && (125)\\ &\leq Q^*(s, a, t) + \alpha H \log |\mathcal{A}|. && (126) \end{aligned}$$

By using the assumed $\alpha < \Delta_{\mathcal{M}}/H \log |\mathcal{A}|$, using $Q^*(s, a, t) + \Delta_{\mathcal{M}} \leq Q^*(s, a', t)$ and that $Q^*(s, a', t) \leq Q^*_{\text{sft}}(s, a', t)$ from Lemma E.6 we get:

$$\begin{aligned} Q^*_{\text{sft}}(s, a, t) &\leq Q^*(s, a, t) + \alpha H \log |\mathcal{A}| && (127)\\ &< Q^*(s, a, t) + \Delta_{\mathcal{M}} && (128)\\ &\leq Q^*(s, a', t) && (129)\\ &\leq Q^*_{\text{sft}}(s, a', t). && (130) \end{aligned}$$

($\Rightarrow$) To show that given the assumptions that $Q^*_{\text{sft}}(s, a, t) < Q^*_{\text{sft}}(s, a', t) \Rightarrow Q^*(s, a, t) < Q^*(s, a', t)$ we will show the contrapostive instead, which is $Q^*(s, a, t) \geq Q^*(s, a', t) \Rightarrow Q^*_{\text{sft}}(s, a, t) \geq Q^*_{\text{sft}}(s, a', t)$. Given that it is assumed that $(a, a') \in \delta(s, t)$, the following implications hold:

$$\begin{aligned} & Q^*(s, a, t) \geq Q^*(s, a', t) && (131)\\ \Rightarrow\ & Q^*(s, a, t) > Q^*(s, a', t) && (132)\\ \Rightarrow\ & Q^*_{\text{sft}}(s, a, t) > Q^*_{\text{sft}}(s, a', t) && (133)\\ \Rightarrow\ & Q^*_{\text{sft}}(s, a, t) \geq Q^*_{\text{sft}}(s, a', t), && (134) \end{aligned}$$

where the first implication uses that $(a, a') \in \delta(s)$, the second reuses the ($\Leftarrow$) proof, and the final implication holds generally. □

E.4 Simple regret

In this section we will revisit *simple regret* in more detail. Recall that our definition of simple regret is:

$$\text{reg}(s_t, \psi^n) = V^*(s_t) - V^{\psi^n}(s_t), \tag{135}$$

where ψ^n is the policy recommended by a forecaster after n rounds or trials. This definition is different to what has been considered by the tree search literature so far [33, 13]. However, we feel that this definition is a more natural extension to the simple regret considered for multi-armed bandit problems [7, 8], as it stems from a more general MDP planning problem that does not necessarily have to be solved using a tree search (for example, consider if we want to reason about a non-tree search algorithm's simple regret, that only outputs full policies). Moreover, we can reconcile the difference by defining the *simple regret of an action* (or *immediate simple regret*) as:

$$\text{reg}_I(s_t, \psi^n) = V^*(s) - \mathbb{E}_{a_t \sim \psi^n(s_t)}[Q^*(s_t, a_t)], \tag{136}$$

and we show that any asymptotic upper bounds provided on the two definitions are equivalent up to a multiplicative factor.

Lemma E.8. *$\mathbb{E}\text{reg}(s_t, \psi^n) = O(f(n))$ for all $s_t \in \mathcal{S}$ iff $\mathbb{E}\text{reg}_I(s_t, \psi^n) = O(f(n))$ for all $s_t \in \mathcal{S}$.*

Proof. Firstly, notice that the simple regret can be written recursively in terms of the immediate simple regret:

$$\begin{aligned}
\text{reg}(s_t, \psi^n) =& V^*(s_t) - V^{\psi^n}(s_t) && (137)\\
=& V^*(s_t) - \mathbb{E}_{a_t \sim \psi^n(s)}[Q^{\psi^n}(s_t, a_t)] && (138)\\
=& V^*(s_t) - \mathbb{E}_{a_t \sim \psi^n(s)}[R(s_t, a_t) + \mathbb{E}_{s_{t+1} \sim \Pr(\cdot|s_t, a_t)}[V^{\psi^n}(s_{t+1})]] && (139)\\
=& V^*(s_t) - \mathbb{E}_{a_t \sim \psi^n(s)}\big[Q^*(s, a) - \mathbb{E}_{s_{t+1} \sim \Pr(\cdot|s_t, a_t)}[V^*(s_{t+1})] \\
& + \mathbb{E}_{s_{t+1} \sim \Pr(\cdot|s_t, a_t)}[V^{\psi^n}(s_{t+1})]\big] && (140)\\
=& V^*(s_t) - \mathbb{E}_{a_t \sim \psi^n(s_t)}[Q^*(s_t, a_t)] \\
& + \mathbb{E}_{a_t \sim \psi_n(s_t), s_{t+1} \sim \Pr(\cdot|s_t, a_t)}[V^*(s_{t+1}) - V^{\psi^n}(s_{t+1})] && (141)\\
=& \text{reg}_I(s, \psi^n) + \mathbb{E}_{a_t \sim \psi^n(s_t), s' \sim \Pr(\cdot|s_t, a_t)}[\text{reg}(s_{t+1}, \psi^n)], && (142)
\end{aligned}$$

where in line (140) we used $R(s, a) = Q^*(s, a) - \mathbb{E}_{s' \sim \Pr(\cdot|s,a)}[V^*(s')]$, a rearrangement of the Bellman optimality equation for $Q^*(s, a)$.

This shows that if $\mathbb{E}\text{reg}(s_t, \psi^n) = O(f(n))$ then $\mathbb{E}\text{reg}_I(s_t, \psi^n) \leq \mathbb{E}\text{reg}(s_t, \psi^n) = O(f(n))$. Now suppose that $\mathbb{E}\text{reg}_I(s_t, \psi^n) = O(f(n))$ and assume an inductive hypothesis that $\mathbb{E}\text{reg}(s_{t+1}, \psi^n) = O(f(n))$ for all $s_{t+1} \in \cup_{a \in \mathcal{A}} \text{Succ}(s_t, a)$, then:

$$\mathbb{E}\text{reg}(s_t, \psi^n) = \mathbb{E}\left[O(f(n)) + \mathbb{E}_{a_t \sim \psi^n(s_t), s_{t+1} \sim \Pr(\cdot|s_t, a_t)}[O(f(n))]\right] = O(f(n)), \tag{143}$$

where the outer expectation is with respect ψ^n (as the relevant MENTS process is a stochastic one). □

It is useful to specialise Lemma E.8 specifically for the form of bounds used later. I.e. the simple regret at s_t admits a regret bound exponential in $N(s_t)$, if and only if, the immediate simple regret admits a bound exponential in $N(s_t)$.

Corollary E.8.1. *Consider any Boltzmann MCTS process. There exists $C_1, k_1 > 0$ such that $\mathbb{E}\text{reg}(s_t, \psi^n) \leq C_1 \exp(-k_1 N(s_t))$ iff there exists $C_2, k_2 > 0$ such that $\mathbb{E}\text{reg}_I(s_t, \psi^n) \leq C_2 \exp(-k_2 N(s_t))$.*

Proof outline. The proof follows similarly to Lemma E.8. The additional nuance is that we need to apply Lemmas E.4 and E.5. Note that the assumption of a minimum action probability in Lemma E.4 is satisfied, because there is a minimum positive probability of selecting an action in any Boltzmann MCTS process (Lemma E.1). The inductive hypothesis for $s_{t+1} \in \cup_{a \in \mathcal{A}} \text{Succ}(s_t, a)$ would give a bound with respect to $N(s_{t+1})$, and the lemmas are required to 'translate' the bound into one with respect to $N(s_t)$. □

E.5 General Q-value convergence result

Recall the (soft) Q-value backups used by Boltzmann MCTS processes (Equations (75) and (78)). Considering that the backups for MENTS and DENTS processes are of similar form, we will show that generally, backups of that form converge (exponentially), given that the values at any child nodes also converge (exponentially). However, towards showing this, we first need to consider the concentration of the empirical transition distribution around the true transition distribution.

Theorem E.9. *Let $\{X_i\}_{i=1}^m$ be random variables drawn from a probability distribution with a cumulative distribution function of F. Let the empirical cumulative distribution function be $F_m(x) = \frac{1}{m} \sum_{i=1}^m \mathbb{1}[X_i < x]$. Then the Dvoretzky-Kiefer-Wolfowitz inequality is:*

$$\Pr\left(\sup_x |F_m(x) - F(x)| > \varepsilon\right) \leq 2 \exp\left(-2m\varepsilon^2\right). \tag{144}$$

Proof. See [41]. □

The Dvoretzky-Kiefer-Wolfowitz inequality is of interest because it allows us to tightly bound the empirical transition probability $N(s_{t+1})/N(s_t, a_t)$ with the true transition probability $p(s_{t+1}|s_t, a_t)$.

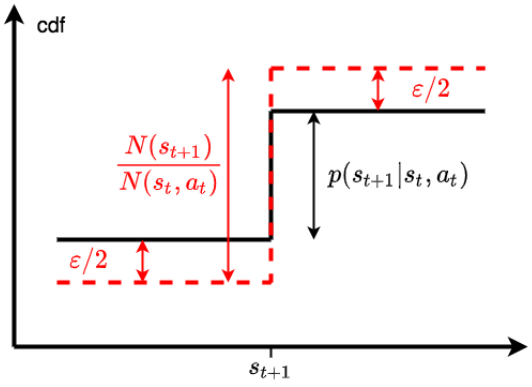


Figure 31: Example cdfs for the transition distribution (black solid line) and empirical transition distribution (red dashed line) around the successor state s_{t+1}. An error of $\varepsilon/2$ between the empirical and true cdfs is shown, and the probability mass for s_{t+1} for the empirical and true transition distributions are shown to indicate how the constructed distribution gives Corollary E.9.1.

Corollary E.9.1. *Consider any Boltzmann MCTS process. For all $(s_t, a_t) \in \mathcal{S} \times \mathcal{A}$ and for all $\varepsilon > 0$ we have:*

$$\Pr\left(\max_{s_{t+1} \in \text{Succ}(s_t,a_t)} \left| \frac{N(s_{t+1})}{N(s_t,a_t)} - p(s_{t+1}|s_t,a_t) \right| > \varepsilon \right) \leq 2\exp\left(-\frac{1}{2}\varepsilon^2 N(s_t,a_t)\right). \tag{145}$$

Proof outline. Recall Equation (68) that allows $N(s_{t+1})$ to be written as a sum of $N(s_t, a_t)$ indicator random variables: $N(s_{t+1}) = \sum_{i \in T(s_t,a_t)} \mathbb{1}[s^i_{t+1} = s_{t+1}]$. Consider mapping the empirical transition distribution, with $N(s_t, a_t) = |T(s_t, a_t)|$ samples, to an appropriate catagorical distribution. Applying Theorem E.9 with $\varepsilon/2$ gives the result. In the constructed cumulative distribution function we need to account for an error of $\varepsilon/2$ before any s_{t+1}, and another error of $\varepsilon/2$ after the same s_{t+1}, to account for an error of ε in the probability mass function (see Figure 31). □

Now that Corollary E.9.1 can be used to bound the empirical transition distribution to the true transition distribution, we can provide a general purpose concentration inequality for Q-values to use in later proofs.

Lemma E.10. *Consider any Boltzmann MCTS process, and some state action pair $(s_t, a_t) \in \mathcal{S} \times \mathcal{A}$. Let $\dot{V}^{N(s)}(s) : \mathcal{S} \to \mathbb{R}$, $\dot{V}^*(s) : \mathcal{S} \to \mathbb{R}$ be some estimated and optimal value functions respectively and suppose that for all $s_{t+1} \in \text{Succ}(s_t, a_t)$ that:*

$$\Pr\left(\left|\dot{V}^{N(s_{t+1})}(s_{t+1}) - \dot{V}^*(s_{t+1})\right| > \varepsilon_1\right) \leq C_{s_{t+1}} \exp(-k_{s_{t+1}}\varepsilon_1^2 N(s')). \tag{146}$$

If we have optimal and estimated Q-values defined as follows:

$$\dot{Q}^*(s,a) = R(s,a) + \mathbb{E}_{s' \sim p(\cdot|s,a)}\left[\dot{V}^*(s')\right], \tag{147}$$

$$\dot{Q}^{N(s,a)}(s,a) = R(s,a) + \sum_{s' \in \text{Succ}(s,a)} \left[\frac{N(s')}{N(s,a)} \dot{V}^{N(s')}(s')\right]. \tag{148}$$

Then for some $C, k > 0$, and for all $\varepsilon > 0$:

$$\Pr\left(\left|\dot{Q}^{N(s,a)}(s,a) - \dot{Q}^*(s,a)\right| > \varepsilon\right) \leq C\exp(-k\varepsilon^2 N(s,a)). \tag{149}$$

Proof. By the assumed bounds, Lemma E.2 and Lemma E.5, there is some $C_1, k_1 > 0$, such that for any $\varepsilon_1 > 0$:

$$\Pr\left(\forall s_{t+1}. \left|\dot{V}^{N(s_{t+1})}(s_{t+1}) - \dot{V}^*(s_{t+1})\right| \leq \varepsilon_1\right) > 1 - C_1\exp(-k_1\varepsilon_1^2 N(s_t,a_t)), \tag{150}$$

and recall using Corollary E.9.1 that for any $p(s_{t+1}|s_t, a_t) > \varepsilon_2 > 0$ we must have:

$$\Pr\left(\max_{s_{t+1}\in\text{Succ}(s_t,a_t)}\left|\frac{N(s_{t+1})}{N(s_t,a_t)} - p(s_t,a_t,s_{t+1})\right| \le \varepsilon_2\right) > 1 - 2\exp\left(-\frac{1}{2}\varepsilon_2^2 N(s_t,a_t)\right). \quad (151)$$

If the events in Inequalities (150) and (151) hold, then $\dot{V}^*(s_{t+1}) - \varepsilon_1 \le \dot{V}^{N_n(s_{t+1})}(s_{t+1}) \le \dot{V}^*(s_{t+1}) + \varepsilon_1$ and $p(s_{t+1}|s_t,a_t) - \varepsilon_2 \le N(s_{t+1})/N(s_t,a_t) \le p(s_{t+1}|s_t,a_t) + \varepsilon_2$. The upper bounds on $\dot{V}^{N_n(s_{t+1})}(s_{t+1})$ and $N(s_{t+1})/N(s_t,a_t)$ can be used to obtain an upper bound on $\dot{Q}^{N(s_t,a_t)}$:

$$\begin{aligned}
&\dot{Q}^{N(s_t,a_t)}(s_t,a_t) && (152)\\
=&R(s_t,a_t) + \sum_{s_{t+1}\in\text{Succ}(s_t,a_t)} \frac{N_n(s_{t+1})}{N_n(s_t,a_t)}\dot{V}^{N_n(s_{t+1})}(s_{t+1}) && (153)\\
\le&R(s_t,a_t) + \sum_{s_{t+1}\in\text{Succ}(s_t,a_t)} (p(s_{t+1}|s_t,a_t)+\varepsilon_2)(\dot{V}^*(s')+\varepsilon_1) && (154)\\
=&R(s_t,a_t) + \mathbb{E}_{s_{t+1}\sim p(\cdot|s_t,a_t)}[\dot{V}^*(s_{t+1})] + \varepsilon_1 + \varepsilon_2 \sum_{s_{t+1}\in\text{Succ}(s_t,a_t)} \dot{V}^*(s_{t+1}) + \varepsilon_1\varepsilon_2 && (155)\\
\le&R(s_t,a_t) + \mathbb{E}_{s_{t+1}\sim p(\cdot|s_t,a_t)}[\dot{V}^*(s_{t+1})] + \varepsilon_1\left|\sum_{s_{t+1}\in\text{Succ}(s_t,a_t)} \dot{V}^*(s_{t+1})\right| + \varepsilon_2 + \varepsilon_1\varepsilon_2 && (156)\\
=&\dot{Q}^*(s_t,a_t) + \varepsilon_1\left|\sum_{s_{t+1}\in\text{Succ}(s_t,a_t)} \dot{V}^*(s_{t+1})\right| + \varepsilon_2 + \varepsilon_1\varepsilon_2. && (157)
\end{aligned}$$

Following the same reasoning but using the lower bounds on $\dot{V}^{N_n(s_{t+1})}(s_{t+1})$ and $N(s_{t+1})/N(s_t,a_t)$ gives:

$$\dot{Q}^{N(s_t,a_t)}(s_t,a_t) \ge \dot{Q}^*(s_t,a_t) - \varepsilon_1\left|\sum_{s_{t+1}\in\text{Succ}(s_t,a_t)} \dot{V}^*(s_{t+1})\right| - \varepsilon_2 + \varepsilon_1\varepsilon_2. \quad (158)$$

For any $\varepsilon > 0$, setting $\varepsilon_1 = \min(\varepsilon/3, \varepsilon/3|\sum_{s_{t+1}} \dot{V}^*(s_{t+1})|)$ and $\varepsilon_2 = \min(\varepsilon/3, p(s_{t+1}|s_t,a_t))$ then gives:

$$\dot{Q}^*(s_t,a_t) - \varepsilon \le \dot{Q}^{N(s_t,a_t)}(s_t,a_t) \le \dot{Q}^*(s_t,a_t) + \varepsilon. \quad (159)$$

Using Lemma E.2 liberally, there is some $C_2, C_3, k_2, k_3 > 0$ such that:

$$\begin{aligned}
&\Pr\left(\left|\dot{Q}^{N(s_t,a_t)}(s_t,a_t) - \dot{Q}^*(s_t,a_t)\right| \le \varepsilon\right) && (160)\\
>&\left(1 - C_1\exp(-k_1\varepsilon_1^2 N(s_t,a_t))\right)\left(1 - 2\exp\left(-\frac{1}{2}\varepsilon_2^2 N(s_t,a_t)\right)\right) && (161)\\
=&\left(1 - C_2\exp(-k_2\varepsilon^2 N(s_t,a_t)\right)\cdot\left(1 - C_3\exp(-k_3\varepsilon^2 N(s_t,a_t)\right) && (162)\\
=&1 - C_2\exp(-k_2\varepsilon^2 N(s_t,a_t)) - C_3\exp(-k_3\varepsilon^2 N(s_t,a_t))\\
&+ C_2C_3\exp(-(k_2+k_3)\varepsilon^2 N(s_t,a_t)) && (163)\\
>&1 - C_2\exp(-k_2\varepsilon^2 N(s_t,a_t)) - C_3\exp(-k_3\varepsilon^2 N(s_t,a_t)), && (164)
\end{aligned}$$

and then by negating, with $C = C_1 + C_2$ and $k = \min(k_1, k_2)$ the result follows:

$$\Pr\left(\left|\dot{Q}^{N(s_t,a_t)}(s_t,a_t) - \dot{Q}^*(s_t,a_t)\right| > \varepsilon\right) \le C\exp(-k\varepsilon^2 N(s_t,a_t)). \quad (165)$$

□

E.6 MENTS results

In this subsection we provide results related to MENTS. We begin by providing concentration inequalities for the estimated soft values at each node, around the optimal soft value. Note that this result is similar to some of the results provided in [37], however to keep this paper more self contained we still provide a proof in our particular flavour. After, the concentration inequalities are then used to show bounds on the simple regret of MENTS, given constraints on the temperature.

To prove the concentration inequality around the optimal soft values, start by showing an inductive step.

Lemma E.11. *Consider a MENTS process. Let $s_t \in \mathcal{S}$, with $1 \leq t \leq H$. If for all $s_{t+1} \in \cup_{a\in\mathcal{A}}\text{Succ}(s_t, a)$ there is some $C_{s_{t+1}}, k_{s_{t+1}} > 0$ for any $\varepsilon_{s_{t+1}} > 0$:*

$$\Pr\left(\left|\hat{V}_{\text{sft}}^{N(s_{t+1})}(s_{t+1}) - V_{\text{sft}}^{*}(s_{t+1})\right| > \varepsilon_{s_{t+1}}\right) \leq C_{s_{t+1}} \exp\left(-k_{s_{t+1}}\varepsilon_{s_{t+1}}^2 N(s_{t+1})\right), \tag{166}$$

then there is some $C, k > 0$, for any $\varepsilon > 0$:

$$\Pr\left(\left|\hat{V}_{\text{sft}}^{N(s_t)}(s_t) - V_{\text{sft}}^{*}(s_t)\right| > \varepsilon\right) \leq C \exp\left(-k\varepsilon^2 N(s_t)\right). \tag{167}$$

Proof. Given the assumptions and by Lemmas E.10, E.4 and E.2, there is some $C, k > 0$ such that for any $\varepsilon > 0$:

$$\Pr\left(\forall a_t \in \mathcal{A}. \left|\hat{Q}_{\text{sft}}^{N(s_t,a_t)}(s_t, a_t) - Q_{\text{sft}}^{*}(s_t, a_t)\right| \leq \varepsilon\right) > 1 - C\exp(-k\varepsilon^2 N(s_t)). \tag{168}$$

So with probability at least $1 - C\exp(-k\varepsilon^2 N(s_t))$, for any a_t, the following holds:

$$Q_{\text{sft}}^{*}(s_t, a_t) - \varepsilon \leq \hat{Q}_{\text{sft}}^{N(s_t,a_t)}(s_t, a_t) \leq Q_{\text{sft}}^{*}(s_t, a_t) + \varepsilon. \tag{169}$$

Using the upper bound on $\hat{Q}_{\text{sft}}^{N(s_t,a_t)}(s_t, a_t)$ in the soft backup equation for $\hat{V}_{\text{sft}}^{N(s_t)}(s_t)$ (Equation (74) gives:

$$\hat{V}_{\text{sft}}^{N(s_t)}(s_t) = \alpha \log \sum_{a\in\mathcal{A}} \exp\left(\frac{\hat{Q}_{\text{sft}}^{N(s_t,a_t)}(s_t, a)}{\alpha}\right) \tag{170}$$

$$\leq \alpha \log \sum_{a\in\mathcal{A}} \exp\left(\frac{Q_{\text{sft}}^{*}(s_t, a) + \varepsilon}{\alpha}\right) \tag{171}$$

$$= \alpha \log \sum_{a\in\mathcal{A}} \exp\left(\frac{Q_{\text{sft}}^{*}(s_t, a)}{\alpha}\right) + \varepsilon \tag{172}$$

$$= V_{\text{sft}}^{*}(s_t) + \varepsilon, \tag{173}$$

noting that the *softmax* function monotonically increases in its arguments. Then with similar reasoning using the lower bound on $\hat{Q}_{\text{sft}}^{N(s_t,a_t)}(s_t, a_t)$ gives:

$$\hat{V}_{\text{sft}}^{N(s_t)}(s_t) \geq V_{\text{sft}}^{*}(s_t) - \varepsilon, \tag{174}$$

and hence:

$$|\hat{V}_{\text{sft}}^{N(s_t)}(s_t) - V_{\text{sft}}^{*}(s_t)| \leq \varepsilon. \tag{175}$$

This therefore shows:

$$\Pr\left(\left|\hat{V}_{\text{sft}}^{N(s_t)}(s_t) - V_{\text{sft}}^{*}(s_t)\right| \leq \varepsilon_1\right) > 1 - C\exp(-k\varepsilon^2 N(s_t)), \tag{176}$$

and negating probabilities gives the result. □

Then completing the induction gives the concentration inequalities desired for any state that MENTS might visit.

Theorem E.12. *Consider a MENTS process, let $s_t \in \mathcal{S}$ then there is some $C, k > 0$ for any $\varepsilon > 0$:*

$$\Pr\left(\left|\hat{V}_{\text{sft}}^{N(s_t)}(s_t) - V_{\text{sft}}^*(s_t)\right| > \varepsilon\right) \leq C\exp\left(-k\varepsilon^2 N(s_t)\right). \tag{177}$$

Moreover, at the root node s_0 we have:

$$\Pr\left(\left|\hat{V}_{\text{sft}}^{N(s_0)}(s_0) - V_{\text{sft}}^*(s_0)\right| > \varepsilon\right) \leq C\exp\left(-k\varepsilon^2 n\right). \tag{178}$$

Proof. Consider that the result holds for $t = H+1$, because $\hat{V}_{\text{sft}}^{N(s_{H+1})}(s_{H+1}) = V_{\text{sft}}^*(s_{H+1}) = 0$. Therefore the result holds for any $t = 1, ..., H+1$ by induction using Lemma E.11. Noting that $N(s_0) = n$ gives (178). □

Note that all of our concentration inequalities imply a convergence in probability. For example, we explicitly demonstrate this in Corollary E.12.1.

Corollary E.12.1. $\hat{Q}_{\text{sft}}^{N(s_t,a_t)}(s_t, a_t) \xrightarrow{p} Q_{\text{sft}}^*(s_t, a_t)$

Proof. This follows from the concentration inequalities shown in Lemma E.10 and Theorem E.12. As the RHS of the bound tend to zero as $n \to \infty$, these concentration inequalities follow by taking the limit $n \to \infty$:

$$\lim_{n\to\infty} \Pr\left(\left|\hat{Q}_{\text{sft}}^{N(s_t,a_t)}(s_t, a_t) - Q_{\text{sft}}^*(s_t, a_t)\right| > \varepsilon\right) \leq \lim_{n\to\infty} C\exp\left(-k\varepsilon^2 N(s_t)\right) \tag{179}$$

$$= 0. \tag{180}$$

□

Now we use the concentration inequalities to show that we are exponentially unlikely to recommend a suboptimal action with respect to the standard values (i.e. the simple regret tends to zero exponentially), provided the temperature parameter α is small enough.

Lemma E.13. *Consider a MENTS process with $\alpha < \Delta_{\mathcal{M}}/3H\log|\mathcal{A}|$. Let $s_t \in \mathcal{S}$, with $1 \leq t \leq H$ then there is some $C', k' > 0$ such that:*

$$\mathbb{E}\text{reg}_I(s_t, \psi_{\text{MENTS}}^n) \leq C'\exp\left(-k'N(s_t)\right). \tag{181}$$

Proof. Let a^* be the optimal action with respect to the soft values, so $a^* = \arg\max_{a\in\mathcal{A}} Q_{\text{sft}}^*(s_t, a)$. Then by Lemma E.7 a^* must also be the optimal action for the standard Q-value function $a^* = \arg\max_{a\in\mathcal{A}} Q^*(s_t, a)$. By Theorem E.12 and Lemmas E.10 and E.4 there exists $C_1, k_1 > 0$ such that for all $\varepsilon_1 > 0$:

$$\Pr\left(\forall a_t \in \mathcal{A}\left|\hat{Q}_{\text{sft}}^{N(s_t,a_t)}(s_t, a_t) - Q_{\text{sft}}^*(s_t, a_t)\right| \leq \varepsilon_1\right) > 1 - C_1\exp(-k_1\varepsilon_1^2 N(s_t)). \tag{182}$$

Setting $\varepsilon_1 = \Delta_{\mathcal{M}}/3H\log|\mathcal{A}|$ then gives with probability at least $1 - C_1\exp(-k_2 N(s_t))$ (where $k_2 = k_1(\Delta_{\mathcal{M}}/3H\log|\mathcal{A}|)^2$) that for all actions $a_t \in \mathcal{A}$:

$$Q_{\text{sft}}^*(s_t, a_t) - \Delta_{\mathcal{M}}/3 \leq \hat{Q}_{\text{sft}}^{N(s_t,a_t)}(s_t, a_t) \leq Q_{\text{sft}}^*(s_t, a_t) + \Delta_{\mathcal{M}}/3. \tag{183}$$

And hence, with probability at least $1 - C_1\exp(-k_2 N(s_t))$, for all $a \in \mathcal{A} - \{a^*\}$ we have:

$$\hat{Q}_{\text{sft}}^{N(s_t,a)}(s_t, a) \leq Q_{\text{sft}}^*(s_t, a) + \Delta_{\mathcal{M}}/3 \tag{184}$$

$$\leq Q^*(s_t, a) + \alpha H\log|\mathcal{A}| + \Delta_{\mathcal{M}}/3 \tag{185}$$

$$\leq Q^*(s_t, a) + 2\Delta_{\mathcal{M}}/3 \tag{186}$$

$$\leq Q^*(s_t, a^*) - \Delta_{\mathcal{M}}/3 \tag{187}$$

$$\leq Q_{\text{sft}}^*(s_t, a^*) - \Delta_{\mathcal{M}}/3 \tag{188}$$

$$\leq \hat{Q}_{\text{sft}}^{N(s_t,a^*)}(s_t, a^*). \tag{189}$$

Where in the above, the first line holds from the upper bound on $\hat{Q}_{\text{sft}}^{N(s_t,a_t)}(s_t, a_t)$; The second holds from maximising the standard return and entropy portions of the soft value seperately (recall

Inequality (124) in Lemma E.7); The third holds from the assumption on α; The fourth holds from the definition of $\Delta_{\mathcal{M}}$ (also see Inequality (122)); The fifth holds from the optimal soft value being greater than the optimal standard value (Lemma E.6); And the final line holds by using the lower bound on $\hat{Q}_{\text{sft}}^{N(s_t,a_t)}(s_t,a_t)$ given above with $a_t = a^*$.

Negating the probability that (189) holds gives:

$$\Pr\left(\exists a_t \in \mathcal{A} - \{a^*\}.\left(\hat{Q}_{\text{sft}}^{N(s_t,a)}(s_t,a) > \hat{Q}_{\text{sft}}^{N(s_t,a^*)}(s_t,a^*)\right)\right) \leq C_1 \exp(-k_2 N(s_t)). \tag{190}$$

Finally, we can bound our expected immediate regret as follows:

$$\begin{aligned}
&\mathbb{E}\text{reg}_I(s_t, \psi_{\text{MENTS}}^n) && (191)\\
&= \sum_{a \in \mathcal{A}-\{a^*\}} (V^*(s_t) - Q^*(s_t,a)) \Pr\left(\psi_{\text{MENTS}}^n(s_t) = a\right) && (192)\\
&= \sum_{a \in \mathcal{A}-\{a^*\}} (V^*(s_t) - Q^*(s_t,a)) \Pr\left(a = \arg\max_{a'} \hat{Q}_{\text{sft}}^{N(s_t,a)}(s_t,a')\right) && (193)\\
&\leq \sum_{a \in \mathcal{A}-\{a^*\}} (V^*(s_t) - Q^*(s_t,a)) \Pr\left(\hat{Q}_{\text{sft}}^{N(s_t,a)}(s_t,a) > \hat{Q}_{\text{sft}}^{N(s_t,a^*)}(s_t,a^*)\right) && (194)\\
&\leq \sum_{a \in \mathcal{A}-\{a^*\}} (V^*(s_t) - Q^*(s_t,a)) C_1 \exp(-k_2 N(s_t)), && (195)
\end{aligned}$$

and setting $k' = k_2$ and $C' = C_1 \sum_{a \in \mathcal{A}-\{a^*\}} (V^*(s_t) - Q^*(s_t,a))$ gives the result. □

And finally, by using Lemma E.13, we can convert the bound on the immediate simple regret to a bound on the simple regret.

Theorem E.14. *Consider a MENTS process with $\alpha < \Delta_{\mathcal{M}}/3H\log|\mathcal{A}|$. Let $s_t \in \mathcal{S}$, with $1 \leq t \leq H$ then there is some $C', k' > 0$ such that:*

$$\mathbb{E}\text{reg}(s_t, \psi_{\text{MENTS}}^n) \leq C' \exp\left(-k' N(s_t)\right). \tag{196}$$

Moreover, at the root node s_0 we have:

$$\mathbb{E}\text{reg}(s_0, \psi_{\text{MENTS}}^n) \leq C' \exp\left(-k' n)\right). \tag{197}$$

Proof. This theorem holds as a consequence of Corollary E.8.1 and Lemma E.13, and noting that at the root node $N(s_0) = n$. □

We have now shown Theorem 3.2:

Theorem 3.2. *For any MDP $\mathcal{M}$, after running n trials of the MENTS algorithm with $\alpha \leq \Delta_{\mathcal{M}}/3H\log|\mathcal{A}|$, there exists constants $C, k > 0$ such that: $\mathbb{E}[\text{reg}(s_0, \psi_{\text{MENTS}}^n)] \leq C\exp(-kn)$, where $\Delta_{\mathcal{M}} = \min\{Q^*(s,a,t) - Q^*(s,a',t) | Q^*(s,a,t) \neq Q^*(s,a',t), s \in \mathcal{S}, a, a' \in \mathcal{A}, t \in \mathbb{N}\}$.*

Proof. This is part of Theorem E.14. □

And to show Proposition 3.1 we will utilise Theorem E.12 and return to the modified 10-chain problem, with illustration repeated in Figure 32 for reference.

Proposition 3.1. *There exists an MDP $\mathcal{M}$ and temperature α such that $\mathbb{E}[\text{reg}(s_0, \psi_{\text{MENTS}}^n)] \not\to 0$ as $n \to \infty$. That is, MENTS is not consistent.*

Proof. We give a proof by construction. Recall the modified 10-chain problem, with $R_f = 1/2$ in Figure 32, and consider a MENTS process (i.e. running MENTS) with a temperature $\alpha = 1$ for n trials. By considering the optimal soft Bellman equations (4) and (5), one can verify that $Q_{\text{sft}}^*(1,2) = 0.9$ and $Q_{\text{sft}}^*(1,1) = \log\left(\exp(1/2) + \sum_{i=0}^{8}\exp(i/10)\right) \approx 2.74$.

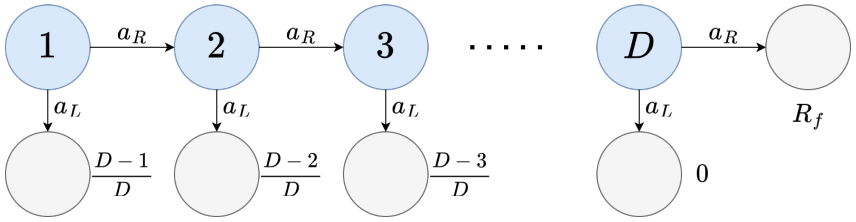


Figure 32: An illustration of the *(modified) D-chain problem*, where 1 is the starting state, all transitions are deterministic and values next to states represents the reward for arriving in that state.

Theorem E.12, Lemma E.10 and Lemma E.4 implies that there is some $C, k > 0$ for any $\varepsilon > 0$:

$$\Pr\left(\left|Q_{\text{sft}}^{N(1,1)}(1,1) - Q_{\text{sft}}^{*}(1,1)\right| > \varepsilon\right) \leq C\exp(-k\varepsilon^2 N(1)) = C\exp(-k\varepsilon^2 n). \tag{198}$$

Letting $\varepsilon = 1$ and using $Q_{\text{sft}}^{*}(1,1) > 5/2$ gives:

$$\begin{aligned}\Pr\left(Q_{\text{sft}}^{N(1,1)}(1,1) < 3/2\right) \leq& \Pr\left(\left|Q_{\text{sft}}^{N(1,1)}(1,1) - Q_{\text{sft}}^{*}(1,1)\right| > 1\right) &(199)\\ \leq& C\exp(-kn). &(200)\end{aligned}$$

And hence:

$$\Pr(\psi_{\text{MENTS}}^{n}(1) = 1) > 1 - C\exp(-kn) \tag{201}$$

Consider that the best simple regret an agent can achieve after selecting action 1 from the starting state is $1/10$. Let $M = \log(2C)/k$, so that $C\exp(-kM) = 1/2$. Then, for all $n > M$ we have $\Pr(\psi_{\text{MENTS}}^{n}(1) = 1) > 1/2$, and hence:

$$\begin{aligned}\mathbb{E}\text{reg}(1, \psi_{\text{MENTS}}^{n}) >& \frac{1}{10}\cdot\Pr(\psi_{\text{MENTS}}^{n}(1) = 1) &(202)\\ >& \frac{1}{20}. &(203)\end{aligned}$$

Thus $\mathbb{E}r(1, \psi_{\text{MENTS}}^{n}) \not\to 0$. □

Additionally, we can show that there is an MDP such that for any α MENTS will either not be consistent, or will take exponentially long in the size of the state space of the MDP. Below we state this formally and provide a proof outline in Theorem E.15, which uses the MDP defined in Figure 33.

Theorem E.15. *Consider a MENTS process with arbitrary temperature α. There exists and MDP such that for any α that MENTS is either not consistent, or requires an exponential number of trials in the size of the state space. More precisely, either $\mathbb{E}\text{reg}(s_0, \psi_{MENTS}^{n}) \not\to 0$ or $\mathbb{E}\text{reg}(1, \psi_{MENTS}^{n}) \geq c(1 - \frac{n}{k^{|S|}})$, which implies that $\mathbb{E}\text{reg}(s_0, \psi_{MENTS}^{n}) > 0$ for $n < k^{|S|}$.*

Proof outline. Proof by construction. Consider the adapted-chain MDP defined in Figure 33, which is parameterised by D the length of the UCT gauntlet, and K half the number of states in the entropy trap. To show our claim, we split the analysis into two cases.

Case 1: $\alpha > \frac{1}{\log(2)K}$

If $\alpha > \frac{1}{\log(2)K}$, the soft value of E is greater than one for any policy over the actions, and ϕ is a uniform policy (note that because there are no MDP rewards after E, ϕ is both the initial policy for MENTS and the optimal soft policy):

$$\begin{aligned}V_{\text{sft}}^{*}(E) =& 0 + \alpha\cdot\mathcal{H}(\phi) &(204)\\ =& \alpha\cdot\log(2)K &(205)\\ >& 1 &(206)\\ =& V_{\text{sft}}^{*}(F). &(207)\end{aligned}$$

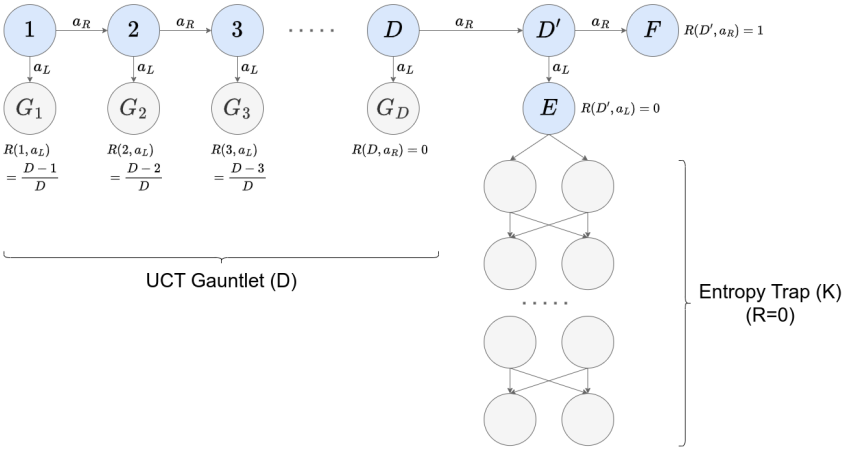


Figure 33: An illustration of the *adapted-chain problem*, where 1 is the starting state, all transitions are deterministic and values next to states represents the reward for arriving in that state. The MDP is split into two sections, firstly, the UCT gauntlet refers to the D-chain part of the MDP, which is difficult for UCT algorithms, and secondly, the entropy trap, where the agent has to chose between states E and F. The entropy trap consists of two chains of K states, all of which give a reward of 0 for visiting, but allows an agent to follow a policy with up to $\log(2)K$ entropy. So the optimal values for E and F are $V^*_{\text{sft}}(E) = \log(2)K$ and $V^*_{\text{sft}}(F) = 1$ respectively.

Hence the optimal soft poilicy (which MENTS converges to) will recommend going to state E and gathering 0 reward. Hence in this case the simple regret will converge to 1 as the optimal value is $V^*(1) = 1$. That is $\mathbb{E}\text{reg}(1, \psi^n_{\text{MENTS}}) \to 1 > 0$.

Case 2: $\alpha \leq \frac{1}{\log(2)K}$

In this case, we argue that with a low α that MENTS will only have a low probability of hitting state D, that is, it requires a lot of trials to garuntee that at least one trial has reached state D, which is necessary to get the reward of 1 (and simple regret of 0).

First consider the composed and simplified soft backup on the adapted-chain problem for any $0 < i < D$ to get:

$$\hat{V}^{N(i-1)}_{\text{sft}}(i-1) = \alpha \log\left(\frac{1}{\alpha}\left(\frac{D-i+1}{D} + \hat{V}^{N(i+1)}_{\text{sft}}(i+1)\right)\right) \tag{208}$$

$$\leq \max\left(\frac{D-i+1}{D}, \hat{V}^{N(i)}_{\text{sft}}(i)\right) + \alpha\log(2) \tag{209}$$

$$\leq \max\left(\frac{D-i+1}{D}, \hat{V}^{N(i)}_{\text{sft}}(i)\right) + \frac{1}{K} \tag{210}$$

where we used the property of log-sum-exp that $\alpha \log \sum_{i=1}^{\ell} \exp(x_i/\alpha) \leq \max_i(x_i) + \alpha\log(\ell)$. Assume that $K \geq D$ and $\hat{V}^{N(D)}_{\text{sft}}(D) = 0$ (we will check these assumptions are valid later). Then using an induction hypothesis of $\hat{V}^{N(i)}_{\text{sft}}(i) \leq \frac{D-i+1}{D}$ we have:

$$\hat{V}^{N(i-1)}_{\text{sft}}(i-1) \leq \max(\frac{D-i+1}{D}, \hat{V}^{N(i)}_{\text{sft}}(i)) + \frac{\log(2)}{\log(K)} \tag{211}$$

$$= \frac{D-i+1}{D} + \frac{1}{K} \tag{212}$$

$$\leq \frac{D-i+1}{D} + \frac{1}{D} \tag{213}$$

$$= \frac{D-i+2}{D} = \frac{D-(i-1)+1}{D} \tag{214}$$

We can then show that the probability the event $Y(n)$ that MENTS visits state D in its nth trial is less than 2^{-D}, given $\hat{V}_{\text{sft}}^{N(D)}(D) = 0$. Because we have just shown that $\hat{V}_{\text{sft}}^{N(i)}(i) \leq \frac{D-i+1}{D} = \hat{V}_{\text{sft}}^{N(G_{i-1})}(G_{i-1})$, using a symmetry argument we must have $\psi_{\text{MENTS}}^{n}(a_R|i) < \frac{1}{2}$, and as such, we must have $\Pr\left(Y(n)\middle|\hat{V}_{\text{sft}}^{N(D)}(D) = 0\right) < \frac{1}{2^D}$.

Then let $Z(n) = \neg \bigcup_{j=1}^{n} Y(j)$ be the event that no trial of MENTS has visited state D in any of the first n trials. And note that $Z(n)$ implies that $\hat{V}_{\text{sft}}^{N(D)}(D) = 0$:

$$\begin{aligned}
\Pr(Z(n)) &= \Pr\left(Z(n) \cap \hat{V}_{\text{sft}}^{N(D)}(D) = 0\right) && (215)\\
&= \Pr\left(\neg Y(n) \cap Z(n-1) \cap \hat{V}_{\text{sft}}^{N(D)}(D) = 0\right) && (216)\\
&= \Pr\left(\neg Y(n)\middle|Z(n-1) \cap \hat{V}_{\text{sft}}^{N(D)}(D) = 0\right) \Pr\left(Z(n-1) \cap \hat{V}_{\text{sft}}^{N(D)}(D) = 0\right) && (217)\\
&\geq (1 - 2^{-D}) \Pr\left(Z(n-1) \cap \hat{V}_{\text{sft}}^{N(D)}(D) = 0\right) && (218)\\
&= ... && (219)\\
&\geq (1 - 2^{-D})^n \Pr\left(Z(0) \cap \hat{V}_{\text{sft}}^{N(D)}(D) = 0\right) && (220)\\
&= (1 - 2^{-D})^n && (221)\\
&\geq 1 - n2^{-D}, && (222)
\end{aligned}$$

where in the penultimate line we used $\Pr\left(Z(0) \cap \left(\hat{V}_{\text{sft}}^{N(D)}(D) = 0\right)\right) = 1$, as $Z(0)$ and $\hat{V}_{\text{sft}}^{N(D)}(D) = 0$ are vacuously true at the start of running the algorithm, and in the final line we used Bernoulli's inequality.

Informally, $Z(n)$ implies that $V^{\psi_{\text{MENTS}}^{n}}(1) \leq \frac{9}{10}$, as no trial has even reached D to be able to reach the reward of 1 from F. And hence the expected simple regret in this environment of MENTS can be bounded below as follows:

$$\begin{aligned}
\mathbb{E}\text{reg}(1, \psi_{\text{MENTS}}^{n}) &\geq \left(1 - \frac{9}{10}\right) \Pr(Z(n)) && (223)\\
&\geq \frac{1}{10}\left(1 - n2^{-D}\right). && (224)
\end{aligned}$$

Finally, setting $K = D$, we must have that $D > \frac{|S|}{3}$ and so $\mathbb{E}\text{reg}(1, \psi_{\text{MENTS}}^{n}) \geq 1 - \frac{n}{\sqrt[3]{2}^{|S|}}$, which is greater than 0 for $n < \sqrt[3]{2}^{|S|}$. That is $c = 0.1$ and $k = \sqrt[3]{2}$.

□

E.7 DENTS results

In this subsection we provide results related to DENTS. Similarly to the MENTS results, we begin by providing concentration inequalities for the estimated soft values at each node, around the optimal soft value. And after, the concentration inequalities are then used to show bounds on the simple regret of DB-MENTS, however, this time no constraints on the temperature are required.

Lemma E.16. *Consider a DENTS process. Let $s_t \in \mathcal{S}$, with $1 \leq t \leq H$. If for all $s_{t+1} \in \cup_{a\in\mathcal{A}}\text{Succ}(s_t, a)$ we have some $C_{s_{t+1}}, k_{s_{t+1}} > 0$ for any $\varepsilon_{s_{t+1}} > 0$:*

$$\Pr\left(\left|\hat{V}^{N(s_{t+1})}(s_{t+1}) - V^*(s_{t+1})\right| > \varepsilon_{s_{t+1}}\right) \leq C_{s_{t+1}} \exp\left(-k_{s_{t+1}}\varepsilon_{s_{t+1}}^2 N(s_{t+1})\right), \quad (225)$$

then there is some $C, k > 0$, for any $\varepsilon > 0$:

$$\Pr\left(\left|\hat{V}^{N(s_t)}(s_t) - V^*(s_t)\right| > \varepsilon\right) \leq C \exp\left(-k\varepsilon^2 N(s_t)\right). \quad (226)$$

Proof. Given the assumptions and by Lemmas E.10, E.2 and E.4, for some $C, k > 0$ we have for any $\varepsilon_1 > 0$:

$$\Pr\left(\forall a_t \in \mathcal{A}. \left|\hat{Q}^{N(s_t,a_t)}(s_t,a_t) - Q^*(s_t,a_t)\right| \leq \varepsilon_1\right) > 1 - C\exp(-k\varepsilon_1^2 N(s_t)). \tag{227}$$

Let $\varepsilon > 0$, and set $\varepsilon_1 = \min(\varepsilon, \Delta_{\mathcal{M}}/2)$. So with probability at least $1 - C_1\exp(-k_1\varepsilon_1^2 N(s_t))$ we have for any a_t that:

$$Q^*(s_t,a_t) - \varepsilon_1 \leq \hat{Q}^{N(s_t,a_t)}(s_t,a_t) \leq Q^*(s_t,a_t) + \varepsilon_1. \tag{228}$$

Let $a^* = \max_{a\in\mathcal{A}} Q^*(s_t,a)$. Using $\varepsilon_1 \leq \Delta_{\mathcal{M}}/2$ in (228), then for any $a \in \mathcal{A} - \{a^*\}$:

$$\begin{aligned}
\hat{Q}^{N(s_t,a)}(s_t,a) \leq& Q^*(s_t,a) + \Delta_{\mathcal{M}}/2 && (229)\\
\leq& Q^*(s_t,a^*) - \Delta_{\mathcal{M}}/2 && (230)\\
\leq& \hat{Q}^{N(s_t,a^*)}(s_t,a^*), && (231)
\end{aligned}$$

and hence $\arg\max_{a\in\mathcal{A}} \hat{Q}^{N(s_t,a)}(s_t,a) = a^*$. As a consequence:

$$\hat{V}^{N(s_t)}(s_t) = \max_a \hat{Q}^{N(s_t,a)}(s_t,a) = \hat{Q}^{N(s_t,a^*)}(s_t,a^*). \tag{232}$$

Then using (228) with $a_t = a^*$ (noting $V^*(s_t) = Q^*(s_t,a^*)$), using (232) and using $\varepsilon_1 \leq \varepsilon$ gives:

$$\begin{aligned}
V^*(s_t) - \varepsilon \leq& V^*(s_t) - \varepsilon_1 && (233)\\
\leq& \hat{V}^{N(s_t)}(s_t) && (234)\\
\leq& V^*(s_t) + \varepsilon_1 && (235)\\
\leq& V^*(s_t) + \varepsilon. && (236)
\end{aligned}$$

Hence:

$$\Pr\left(\left|\hat{V}^{N(s_t)}(s_t) - V^*(s_t)\right| > \varepsilon\right) \leq C\exp(-k\varepsilon_1^2 N(s_t)) \leq C\exp(-k\varepsilon^2 N(s_t)), \tag{237}$$

which is the result. □

Similarly to the MENTS section, Lemma E.16 provides an inductive step, which is used in Theorem E.17 to show concentration inequalities at all states that DENTS visits.

Theorem E.17. *Consider a DENTS process, let $s_t \in \mathcal{S}$ then there is some $C, k > 0$ for any $\varepsilon > 0$:*

$$\Pr\left(\left|\hat{V}^{N(s_t)}(s_t) - V^*(s_t)\right| > \varepsilon\right) \leq C\exp\left(-k\varepsilon^2 N(s_t)\right). \tag{238}$$

Moreover, at the root node s_0 we have:

$$\Pr\left(\left|\hat{V}^{N(s_0)}(s_0) - V^*(s_0)\right| > \varepsilon\right) \leq C\exp\left(-k\varepsilon^2 n\right). \tag{239}$$

Proof. The result holds for $t = H+1$ because $\hat{V}^{N(s_{H+1})}(s_{H+1}) = V^*(s_{H+1}) = 0$. Hence the result holds for all $t = 1, ..., H+1$ by induction using Lemma E.16. Noting that $N(s_0) = n$ gives (239). □

Again, the concentration inequalities are used to show that the simple regret tends to zero exponentially, and therefore that DENTS will be exponentially likely in the number of visits to recommend the optimal standard action at every node.

Lemma E.18. *Consider a DENTS process. Let $s_t \in \mathcal{S}$, with $1 \leq t \leq H$ then there is some $C', k' > 0$ such that:*

$$\mathbb{E}\text{reg}_I(s_t, \psi^n_{\text{DENTS}}) \leq C'\exp\left(-k' N(s_t)\right). \tag{240}$$

Proof. Let a^* be the locally optimal standard action, so $a^* = \arg\max_{a \in \mathcal{A}} Q^*(s_t, a)$. By Theorem E.17 and Lemmas E.10 and E.4 there exists $C_1, k_1 > 0$ such that for all $\varepsilon_1 > 0$:

$$\Pr\left(\forall a_t \in \mathcal{A}. \left|\hat{Q}^{N(s_t,a_t)}(s_t, a_t) - Q^*(s_t, a_t)\right| \leq \varepsilon_1\right) > 1 - C_1 \exp(-k_1 \varepsilon_1^2 N(s_t)). \tag{241}$$

Setting $\varepsilon_1 = \Delta_{\mathcal{M}}/2$ then gives with probability $1 - C_1 \exp(-k_2 N(s_t))$ (where $k_2 = k_1 \Delta_{\mathcal{M}}/2$) for all actions $a \in \mathcal{A}$ that:

$$Q^*(s_t, a) - \Delta_{\mathcal{M}}/2 \leq \hat{Q}^{N(s_t,a_t)}(s_t, a) \leq Q^*(s_t, a) + \Delta_{\mathcal{M}}/2. \tag{242}$$

And hence, with probability $1 - C_1 \exp(-k_2 N(s_t))$, for all $a_t \in \mathcal{A} - \{a^*\}$ we have:

$$\begin{aligned}
\hat{Q}^{N(s_t,a)}(s_t, a_t) \leq & Q^*(s_t, a_t) + \Delta_{\mathcal{M}}/2 && (243)\\
\leq & Q^*(s_t, a^*) - \Delta_{\mathcal{M}}/2 && (244)\\
\leq & \hat{Q}^{N(s_t,a^*)}(s_t, a^*). && (245)
\end{aligned}$$

Negating the probability of (245) then gives:

$$\Pr\left(\hat{Q}_{\text{sft}}^{N(s_t,a)}(s_t, a) > \hat{Q}_{\text{sft}}^{N(s_t,a^*)}(s_t, a^*)\right) \leq C_1 \exp(-k_2 N(s_t)) \tag{246}$$

Finally, the bound on the expected immediate regret follows:

$$\begin{aligned}
& \mathbb{E}\text{reg}_I(s_t, \psi^n_{\text{DENTS}}) && (247)\\
= & \sum_{a \in \mathcal{A} - \{a^*\}} (V^*(s_t) - Q^*(s_t, a)) \Pr\left(\psi^n_{\text{DENTS}}(s_t) = a\right) && (248)\\
= & \sum_{a \in \mathcal{A} - \{a^*\}} (V^*(s_t) - Q^*(s_t, a)) \Pr\left(a = \arg\max_{a'} \hat{Q}^{N(s_t,a)}(s_t, a')\right) && (249)\\
\leq & \sum_{a \in \mathcal{A} - \{a^*\}} (V^*(s_t) - Q^*(s_t, a)) \Pr\left(\hat{Q}^{N(s_t,a)}(s_t, a) > \hat{Q}^{N(s_t,a^*)}(s_t, a^*)\right) && (250)\\
\leq & \sum_{a \in \mathcal{A} - \{a^*\}} (V^*(s_t) - Q^*(s_t, a)) C_1 \exp(-k_2 N(s_t)), && (251)
\end{aligned}$$

and setting $k' = k_2$ and $C' = C_1 \sum_{a \in \mathcal{A} - \{a^*\}} (V^*(s_t) - Q^*(s_t, a))$ gives the result. □

Again similarly to before, by using Lemma E.18, we can convert the bound on the immediate simple regret to a bound on the simple regret.

Theorem E.19. *Consider a DENTS process. Let $s_t \in \mathcal{S}$, with $1 \leq t \leq H$ then there is some $C', k' > 0$ such that:*

$$\mathbb{E}\text{reg}(s_t, \psi^n_{\text{DENTS}}) \leq C' \exp\left(-k' N(s_t)\right). \tag{252}$$

Moreover, at the root node s_0:

$$\mathbb{E}\text{reg}(s_0, \psi^n_{\text{DENTS}}) \leq C' \exp\left(-k' n)\right). \tag{253}$$

Proof. Theorem holds as a consequence of Corollary E.8.1 and Lemma E.18. To arrive at (253) note that $N(s_0) = n$. □

Finally, Theorem 4.2 follows from the results that we have just shown.

Finally, we have shown Theorem 4.2, as it is a subset of what we have already shown.

Theorem 4.2. *For any MDP $\mathcal{M}$, after running n trials of the DENTS algorithm with a root node of s_0, if β is bounded above and $\beta(m) \geq 0$ for all $m \in \mathbb{N}$, then there exists constants $C, k > 0$ such that for all $\varepsilon > 0$ we have $\mathbb{E}[\text{reg}(s_0, \psi^n_{\text{DENTS}})] \leq C \exp(-kn)$, and also $\hat{V}^{N(s_0)}(s_0) \xrightarrow{p} V^*(s_0)$ as $n \to \infty$.*

Proof. The simple regret bound follows immediately from Theorem E.19. Using Theorem E.17 we must have $\Pr\left(\left|\hat{V}^{N(s_0)}(s_0) - V^*(s_0)\right| > \varepsilon\right) \leq C \exp\left(-k\varepsilon^2 n\right) \to 0$ as $n \to \infty$, and hence $\hat{V}^{N(s_0)}(s_0)$ converges in probability to $V^*(s_0)$. □

E.8 BTS results

Recall that the BTS process is a special case of the DENTS process, where the decay function is set to $\beta(m) = 0$. As such, all of the results for DENTS processes also hold for BTS processes, and specifically Theorem 4.1 must hold.

Theorem 4.1. *For any MDP $\mathcal{M}$, after running n trials of the BTS algorithm with a root node of s_0, there exists constants $C, k > 0$ such that for all $\varepsilon > 0$ we have $\mathbb{E}[\text{reg}(s_0, \psi^n_{\text{BTS}})] \leq C \exp(-kn)$, and also $\hat{V}^{N(s_0)}(s_0) \xrightarrow{p} V^*(s_0)$ as $n \to \infty$.*

Proof. Follows from setting $\beta(m) = 0$ and using Theorem 4.2. □

E.9 Results for using average returns in Boltzmann MCTS processes

In this section we give informal proof outlines of our results for AR-BTS and AR-DENTS. To begin with we define the average return $\bar{V}^{N(s)}(s)$ for a decision node at s, and recall the definition of $\bar{Q}^{N(s,a)}(s, a)$:

$$\bar{V}^{N(s_t)+1}(s_t) = \bar{V}^{N(s_t)}(s_t) + \frac{\bar{R}(s_t) - \bar{V}^{N(s_t)}(s_t)}{N(s_t) + 1}, \tag{254}$$

$$\bar{Q}^{N(s_t,a_t)+1}(s_t, a_t) = \bar{Q}^{N(s_t,a_t)}(s_t, a_t) + \frac{\bar{R}(s_t, a_t) - \bar{Q}^{N(s_t,a_t)}(s_t, a_t)}{N(s_t, a_t) + 1}, \tag{255}$$

where $\bar{R}(s_t) = \sum_{i=t}^{H} R(s_i, a_i)$ and $\bar{R}(s_t, a_t) = \sum_{i=t}^{H} R(s_i, a_i)$. Note that these average return values also satisfy the equations:

$$\bar{V}^{N(s_t)}(s_t) = \sum_{a \in \mathcal{A}} \frac{N(s_t, a)}{N(s_t)} \bar{Q}^{N(s_t,a_t)}(s_t, a_t), \tag{256}$$

$$\bar{Q}^{N(s_t,a_t)}(s_t, a_t) = R(s_t, a_t) + \sum_{a \in \mathcal{A}} \frac{N(s')}{N(s_t, a_t)} \bar{V}^{N(s')}(s'). \tag{257}$$

Firstly, we show that using a non-decaying search temperature with AR-BTS is not guaranteed to recommend the optimal policy.

Proposition B.1. *For any $\alpha_{\text{fix}} > 0$, there is an MDP $\mathcal{M}$ such that AR-BTS with $\alpha(m) = \alpha_{\text{fix}}$ is not consistent: $\mathbb{E}[\text{reg}(s_0, \psi^n_{\text{AR-BTS}})] \not\to 0$ as $n \to \infty$.*

Proof outline. Consider the MDP given in Figure 34. We can show inductively that (as $n \to \infty$) the value of $\mathbb{E}\bar{V}^{N(k)}(k) \leq 2E^{D-k+1} < 2$, where $E = e^2/(1 + e^2)$ for $2 \leq k \leq D$. The inductive step is as follows:

$$\mathbb{E}\bar{V}^{N(k)}(k) = \frac{\exp\left(\bar{V}^{N(k+1)}(k+1)\right)}{1 + \exp\left(\bar{V}^{N(k+1)}(k+1)\right)} \mathbb{E}\bar{V}^{N(k+1)}(k+1) + \frac{1}{1 + \exp\left(\bar{V}^{N(k+1)}(k+1)\right)} \cdot 0 \tag{258}$$

$$\leq E\mathbb{E} \cdot \bar{V}^{N(k+1)}(k+1) \tag{259}$$

$$\leq E \cdot 2E^{D-k} \tag{260}$$

$$= 2E^{D-k+1}. \tag{261}$$

where we know that $\exp\left(\bar{V}^{N(k+1)}(k+1)\right) / \left(1 + \exp\left(\bar{V}^{N(k+1)}(k+1)\right)\right) < E$, because the function $e^x/(1 + e^x)$ is monotonically increasing in x, and $\bar{V}^{N(k+1)}(k+1) < 2$. Hence, by choosing an integer D such that $D - 1 \geq \log(1/3)/\log(E)$, we have $\mathbb{E}\bar{V}^{N(2)}(2) = 2E^{D-1} \leq 2/3 < 1 = \mathbb{E}\bar{Q}^{N(1,a_2)}(1, a_2)$.

A full proof should show that AR-BTS does indeed converge to these expected values, possibly through concentration bounds.

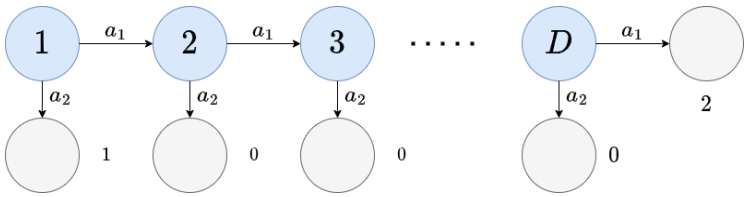


Figure 34: An MDP that AR-BTS will not converge to recommending the optimal policy on, for a large enough value of D.

Hence, AR-BTS does not converge to a simple regret of zero, because the expected Q-values as $n \to \infty$ are $\mathbb{E}\bar{Q}^{N(1,a_2)}(1,a_2) = 1$ and $\mathbb{E}\bar{Q}^{N(1,a_1)}(1,a_1) < 2/3$, so AR-BTS would incorrectly recommend action a_2 from the root node. □

Because the search temperature is now decayed, we need to show that the exploration term in the search policies leads to actions being sampled infinitely often.

Lemma E.20. *Let ρ^k be an arbitrary policy, and let a search policy be $\pi^k(a) = (1-\lambda(k))\rho^k(a) + \lambda(k)/|\mathcal{A}|$, with $\lambda(k) = \min(1, \epsilon/\log(e+k))$, where $\epsilon \in (0,\infty)$. Let $a^k \sim \pi^k$, and let $M(a)$ be the number of times action a was sampled out of m samples, i.e. $M(a) = \sum_{i=1}^m \mathbb{1}[a^i = a]$. Then for all $a \in \mathcal{A}$ we have $M(a) \to \infty$ as $m \to \infty$.*

Proof outline. This lemma restated means that our search policies will select all actions infinitely often. To show this, we argue by contradiction, and suppose that there is some $b \in \mathcal{A}$ such that after ℓ samples that b is never sampled again. The probability of this happening at the mth sample is then:

$$\Pr\left(\bigcap_{i=\ell}^{m} a^i \neq b\right) = \prod_{i=\ell}^{m} \Pr(a^i \neq b) \tag{262}$$

$$\leq \prod_{i=\ell}^{m} \left(1 - \frac{\epsilon}{|\mathcal{A}|\log(e+i)}\right) \tag{263}$$

$$\leq \left(1 - \frac{\epsilon}{|\mathcal{A}|\log(e+m)}\right)^{m-\ell} \tag{264}$$

$$\to 0, \tag{265}$$

as $m \to \infty$. Which is a contradiction. □

Theorem B.2. *For any MDP $\mathcal{M}$, if $\alpha(m) \to 0$ as $m \to \infty$ then $\mathbb{E}[\text{reg}(s_0, \psi^n_{\text{AR-BTS}})] \to 0$ as $n \to \infty$, where n is the number of trials.*

Proof outline. We can argue that average returns converge in probability by induction similar to the previous proofs. Suppose that $\bar{Q}^{N(s_t,a_t)}(s_t,a_t) \xrightarrow{p} Q^*(s_t,a_t)$ as $N(s_t,a_t) \to \infty$. From Lemma E.20, we know that if $N(s_t) \to \infty$, then $N(s_t,a_t) \to \infty$.

All that remains to show that $\bar{V}^{N(s_t)}(s_t) \xrightarrow{p} V^*(s_t)$ as $N(s_t) \to \infty$ is that $\frac{N(s_t,a)}{N(s_t)} \to \mathbb{1}[a = a^*]$. If that is true, then by considering Equation (256), we can see that in the limit $\bar{V}^{N(s_t)}(s_t)$ tends to $\bar{Q}^{N(s_t,a*)}(s_t,a*)$.

Intuitively we can see that $\frac{N(s_t,a)}{N(s_t)} \to \mathbb{1}[a = a^*]$ as the AR-BTS search policy converges to a greedy policy as $\alpha(m) \to 0$. □

Theorem B.3. *For any MDP $\mathcal{M}$, if $\alpha(m) \to 0$ and $\beta(m) \to 0$ as $m \to \infty$ then $\mathbb{E}[\text{reg}(s_0, \psi^n_{\text{AR-DENTS}})] \to 0$ as $n \to \infty$, where n is the number of trials.*

Proof outline. Proof is similar to the proof for Theorem B.2. □

As a final note, using BTS or DENTS with a decaying search temperature $\alpha(m)$ will lead to a consistent algorithm, although they would not admit an exponential regret bound. Proofs would be nearly identical to Theorems B.2 and B.3.

References

[39] Sébastien Bubeck, Rémi Munos, and Gilles Stoltz. Pure exploration in multi-armed bandits problems. In *International conference on Algorithmic learning theory*, pages 23–37. Springer, 2009.

[40] Sébastien Bubeck, Rémi Munos, and Gilles Stoltz. Pure exploration in finitely-armed and continuous-armed bandits. *Theoretical Computer Science*, 412(19):1832–1852, 2011.

[41] Aryeh Dvoretzky, Jack Kiefer, and Jacob Wolfowitz. Asymptotic minimax character of the sample distribution function and of the classical multinomial estimator. *The Annals of Mathematical Statistics*, pages 642–669, 1956.

[42] Zohar Feldman and Carmel Domshlak. Simple regret optimization in online planning for markov decision processes. *Journal of Artificial Intelligence Research*, 51:165–205, 2014.

[43] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. In *The collected works of Wassily Hoeffding*, pages 409–426. Springer, 1994.

[44] Thomas Keller and Patrick Eyerich. Prost: Probabilistic planning based on uct. In *Twenty-Second International Conference on Automated Planning and Scheduling*, 2012.

[45] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[46] David Tolpin and Solomon Shimony. Mcts based on simple regret. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, pages 570–576, 2012.

[47] David J Wu. Accelerating self-play learning in go. *arXiv preprint arXiv:1902.10565*, 2019.

[48] Chenjun Xiao, Ruitong Huang, Jincheng Mei, Dale Schuurmans, and Martin Müller. Maximum entropy monte-carlo planning. *Advances in Neural Information Processing Systems*, 32, 2019.